Data Converters

Monte Mar

Version 1, 12 November 2010 Describes how to model data converters in Verilog-AMS.

Contents

1 Introduction

So far in this book, the modeling techniques have focused on blocks where the major choices are transistor-level modeling or behavioral modeling of a single block. When several blocks are combined, the choices for modeling techniques increase dramatically. The main objectives for modeling need to be defined in order to guide which modeling techniques will be most effective.

In this paper, four types of data converters are examined:

- Flash Converters
- Pipelined Converters
- Oversampling/Noise Shaping Converters
- Current Steering D/A converters

An A/D converter can be thought of as a complex mixed-signal system. Analog and digital blocks of varying complexity are combined to perform a function that is often timing critical. Since it is a system it can be modeled at different levels of abstraction. The correct modeling technique to use is defined by the results that need to be obtained. In early design stages, a very simple behavioral model may be needed by system level designers to investigate trade-offs in sample rate against the number of bits required. During the detailed design stage, designers may need a model more closely based on structure that captures the effects of circuit non-idealities to investigate effects like non-linearity on the converter or sample rate limitations. Different levels of abstraction can be used in each of the components to capture the relevant behavior. For full-chip validation, a model that correctly captures performance and system timing may be needed. Finally, a very specific model replicating a certain type of failure might need to be developed.

For each of the applications, there are trade-offs in speed of evaluation and accuracy of the model. At one end of the spectrum, SPICE simulation can be considered an accurate behavioral model. However for some converters, especially oversampling converters, the numerical noise of the simulator can provide artifacts that mask the true resolution of the converter. At the other end of the spectrum, simple behavioral models capture the function of the converter, but may not be able to adequately model second order effects like non-linearity. This paper illustrates how these types of problems can be attacked with behavioral modeling in Verilog AMS.

As a general rule, the modeling techniques used in this paper focus on using the highest level of abstraction. This approach usually gives the fastest model, providing the designer with the capability of exploring parametric variations in the model. Balanced with the level of abstraction is an approach that can be extended to capture relevant second order effects as needed.

2 A Simple Functional A/D Model

The function of the A/D converter is well known and independent of the conversion algorithm used. Under the command of a request-for-conversion signal, the A/D block will sample the input voltage and provide a digital representation of the analog voltage with respect to some reference voltage. For most system-level simulation applications, a very simple model can be used regardless of the implementation.

Consider the model shown in Listing 1, which was modified from the code found at the examples section at the VerilogAMS website on *www.eda.org*. The website model is based on Verilog-A where the Voltage nature provides support for continuous-valued wires. Within Verilog AMS, there are three alternatives for continuous time wires: Voltage, electrical, and wreal wires. Wires that are declared electrical use the electrical discipline, which forces the through (voltage) and across (current) variables to conform to

LISTING 1 *A behavioral model of an A/D converter.*

```
'include "discipline.h"
module adc_8bit_ideal(dout, in, clk);
voltage in;
output dout;
reg [7:0] dout;
input clk;
parameter real vref     = 1.0 from [0:10];
parameter real propDel = 2.0 from (0:100]; // Delay is determined by 'timescale
real unconverted, halfref;
integer i;

initial begin
    for (i = 0; i < 8; i = i + 1) begin
        dout[i] = 0;
    end
end

always @(posedge(clk)) begin
    #propDel;
    halfref = vref / 2;
    unconverted = V(in);
    for (i = 7; i >= 0; i = i – 1) begin
        if (unconverted > halfref) begin
            dout[i] = 1;
            unconverted = unconverted - halfref;
        end else begin
            dout[i] = 0;
        end
        unconverted = unconverted * 2.0;
    end
end
endmodule
```

conservation laws (Kirchhoff's voltage and current laws). They can only be assigned values within an analog context. The voltage nature is similar, but there is no associated current context. The wreal construct allows real valued wires.

Models based on Voltage or electrical variables must use the analog solver. These models will be much slower than digital models since convergence is needed at every time-step. For many models in this paper, only voltages are being sampled and processed. The amount of current does not matter. This means that wires can be declared using the Voltage nature. If a much faster digital event-driven formulation can be found, *wreal* wires can be used and the time-step solver will be avoided. As a general rule of thumb, replacing a SPICE transistor level model with a behavioral model that uses the analog solver provides speed-ups in the 5-10X region. Replacing a analog solver model with an event-driven wreal-based model can provide another 10-20X speedup. An event-driven model can achieve a 100-200X speedup over a transistor level simulation [1].

The model shown in Listing 1 avoids the analog context. It implements a modified successive approximation algorithm. The conversion is controlled by the rising edge of the *clk* signal. The input is sampled and converted to a real value. Since there is no need for conservation of voltage and current, there is no need to use electrical variables. A loop is used to make a decision as to whether the current value (unconverted) is above 1/2 of the full scale reference or below. The loop then calculates the residue and amplifies it, repeating the binary search to generate the bits of the conversion word.

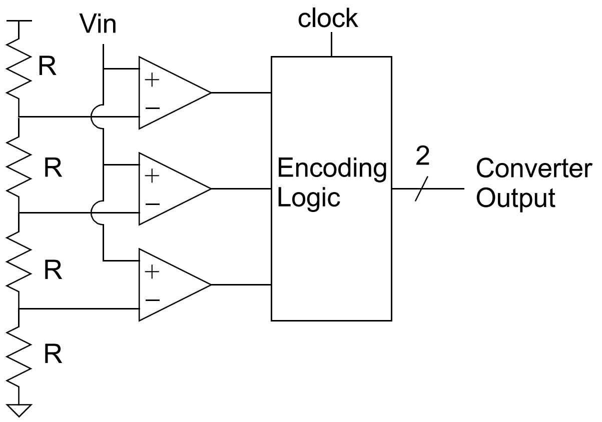
While the routine is compact, efficient, and functionally correct at a high level, it does not model some key features in converters that may affect the overall performance of the system. An inherent sample and hold is coded into the routine. The conversion time is fixed and controlled by a fixed delay (2.1 ns in this case). The model assumes equally spaced sizes for quantization steps, implying perfect linearity. The output is coded as an unsigned binary signal. The following sections detail various A/D converters and present methods for modeling a select set of problems.

3 Flash Converters

3.1 Introduction

Flash converters will be used to introduce behavioral models for A/D converters. A flash converter makes use of a resistor string, comparators, and digital decoding logic to achieve an A/D conversion. A basic flash 2-bit flash converter is shown in Figure 1.

FIGURE 1 *A two-bit flash converter.*



In a simple flash A/D converter, *n*-bits of resolution are obtained by using a resistor string with 2^{n-1} taps and 2^{n-1} comparators. One of the key features of the flash architecture is the open loop nature of the circuits. There are no feedback paths nor amplifiers with feedback providing precision gains. Another key feature is the parallelism of the architecture. A single input is simultaneously compared to 2^{n-1} reference values. The result from the comparators is a thermometer code indicating the conversion value. A thermometer code is a word with only one transition between 0 and 1 values, so the boundary appears much like the mercury in a thermometer. Digital logic is used to encode the thermometer coded output to a binary coded word. This configuration provides the highest speed possible in a given technology. The conversion time is simply a comparator evaluation time, plus the time to encode the thermometer code output.

The large input capacitance of these converters provides problems in systems. Note that each of the 2^{n-1} comparators will have one input tied to the resistor string and the other input tied to the input. This usually means that some type of buffer amplifier is needed to drive the comparator inputs.

Most A/D converters require a clock or a begin conversion signal. A flash converter can be designed to be run without a clock but this requires that the propagation delays of all the comparator paths be equal so that the thermometer code output moves monotonically without glitching. More commonly, the outputs of the comparators are latched under the control of a clock signal.

Performance of these converters is limited by several factors. The overall resolution is limited by the precision of the resistor string and by comparator offsets.The size (in area) is limited by the need to integrate the large number of comparators. Typically, resolutions of 8-bits are obtained with an untrimmed resistor and careful layout. Speed is limited by the power dissipation of the resistor string and comparators. More current can provide faster comparators, but incremental additions in current are magnified by the need to have 2^{n-1} comparators.

A wide variety of problems in this type of converter can be studied using behavioral modeling. A few problems will be illustrated with examples, followed by a brief discussion of current directions of research in flash converters.

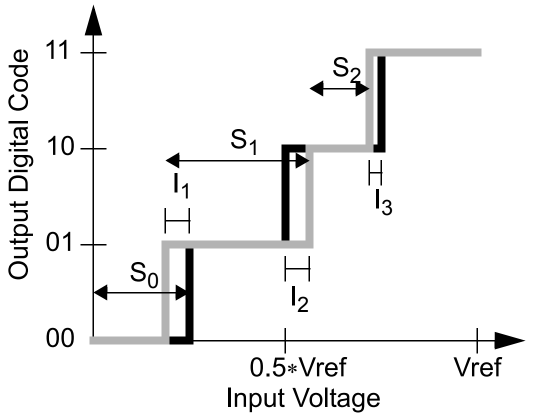
3.2 Modeling Resistor Mismatch

For a well designed flash A/D converter, the resistor mismatch should dominate the overall resolution of the converter. Resistor mismatch can be studied statistically to predict yields. Specific types of mismatch may need to be simulated to verify the effects on overall system performance.

There are several approaches to modeling the resistor mismatch. A simple approach would be to mimic the physical structure by writing a resistor model, a comparator model, and instantiating them. This might be the way it would be done from a schematic representation or if a spice netlist were translated. A faster and more compact way would be to describe the problem algorithmically, providing more compact code and an implementation that is easier to understand.

The previous functional A/D model made use of successive approximation, and can be modified to easily support resistor mismatches. The key idea is that the differences in quantization steps caused by resistor mismatch accumulate along the resistor string giving rise to non-linearities. Figure 2 illustrates this with a plot of the A/D transfer function. The dark bold line is an ideal transfer function, while the shaded line is a non-ideal transfer function. From this plot, the two basic measures of non-linearity, differential and integral, can be determined. The differences between two adjacent resistor taps gives rise to different step sizes, shown as S_0, S_1, and S_2. Differential non-linearity is given by calculating the difference in step-sizes, and finding the maximum value over the whole range. The fact that the differences of all the steps sum up to equal the reference voltage means that it is possible for some taps voltages to be closer to their ideal values even though all the steps do not match. At a given step, the difference between the ideal value and the current value is a measure of integral non-linearity, as shown by I_1, I_2, and I_3. When specifying the quality of an A/D converter, the maximum value over

FIGURE 2 *An illustration of non-linearities in A/D transfer curves.*



all steps is taken as the integral non-linearity. Plots of both DNL and INL vs. code number are typically given to show the quality of a converter.

A portion of a model for resistor mismatches is shown in Listing 2.

LISTING 2 *A modified A/D model with resistor mismatch.*

```
initial begin
    for (i = 0; i < 8; i = i+1) begin
        dout[i] = 0;
    end
    for (i = 1; i < 255; i= i+1) begin
        taps[i] = (1.0+(($random % 65536)/6536.0)∗0.4)∗vref/256;
    end
    taps[255] = vref;
end

always @(posedge(clk)) begin
    unconverted = V(vin);
    refval = 0;
    count = 0;
    for (i = 0; i <= 255; i = i+1) begin
        if (overdrive < unconverted) begin
            refval = refval + taps[i];
            count = i;
        end
    end
    thresh = 128;
    for (i = 7; i >= 0; i = i-1) begin
        if (count > thresh) begin
            dout[i] = 1;
            count = count - thresh;
        end else
            dout[i] = 0;
        thresh = thresh/2;
    end
end
```

This model works, but several improvements could be added. Note that the comparator reference values are stored in an array called result. It was assumed that the 255 values for the array were generated using the **$random** function in an initial block to generate the values, as might be done for statistical analysis. However, it may be desirable to use a fixed set of comparator reference values to model some systematic perturbations in the resistor chain. It is possible to declare the values by hand and include them in the model. A second improvement could be made in computation efficiency. For the 8-bit model, 255 comparisons are performed per conversion. Since the resistor values are calculated prior to execution, they could also be encoded as the 255 comparison levels. In this case, a binary search algorithm could then be implemented using the reference values, provided that the overall reference values remain monotonic with array index. Note that the model calculates the conversion output as an unsigned integer, avoiding the need for thermometer to binary code conversion.

3.3 Modeling Bubble Errors

To obtain the best speed in the converter, a simple but fast comparator is often used. The offsets in the comparators need to be made less than the resolution of a step-size. An offset larger than a step-size will cause a missing code error. If the resistor string is driven by 3.3V and the converter is designed for 8-bit accuracy, the comparator offset must be held to less than 3.3V/256 = 12.9 mV. For CMOS processes, this can be difficult to do, so some form of offset cancellation is often used.

The comparator offset has two effects. If the comparator offset is less than the size of the quantization step, the effect looks much like a mismatch in resistance and leads to linearity problems. If the effect is larger than the quantization step, it is possible that the thermometer code output can become non-monotonic.

The non-monotonicity of the code can be caused by other problems. For very high speed flash converters, mismatches in comparator delay can also cause the problem. The delay differences do not have to be large and code errors only show up when the input is moving very fast relative to the sampling rate. The errors typically show up as a non-monotonic thermometer code. This type of error is called a bubble error. The thermometer code will have a pattern like 111010000. The extra 0 looks like a bubble in the mercury of a thermometer. The encoding of the error can lead to codes that are distant from the expected code, and this may appear like a brief anomaly or sparkle in the time series of codes. Hence they are called sparkle errors.

Bubble errors can also be caused by metastability in the latched comparators. Metastability is the condition where the output of a latch does not resolve to either logic level within a clock period. The latch instead remains near an unstable equilibrium point. Comparator metastability is related to the amount of time allowed for signal evolution and the gain of the comparator. Proper design is needed to ensure adequate time for resolution and high enough gain to limit the probability of metastability to a low enough value. It is also possible to detect metastability and force an output state.

There are two reasons for modeling bubble errors. The first reason may be to test hypotheses on why a converter creates the errors. The second reason may be to see how a system responds to bubble errors and if they can be tolerated. In the first case, more information is needed to determine what a good model is. In the second case, bubble errors can be introduced by injecting them into the model.

In order to model bubble errors, the model must be extended from the simple model of Listing 2. An integer output cannot contain information about the bubble error. An array or vector can be used to hold the thermometer code. Additional digital logic would be needed for thermometer code to binary or gray code conversion. Logic implementations of the encode logic are described in the literature [2], [3].

One way to model the comparator errors is to incorporate a metastability model for the comparator error. The chance of metastability is modeled as a function of the input overdrive to the comparator and a probability that the comparator does not evaluate within the allotted time. To model the problem of comparator delay, small delay differences can be assigned to each comparator much like small differences in resistance were assigned to the values in the resistor ladder. Listing 3 shows a portion of a model with

LISTING 3 *A code fragment adding delays and a metastability model.*

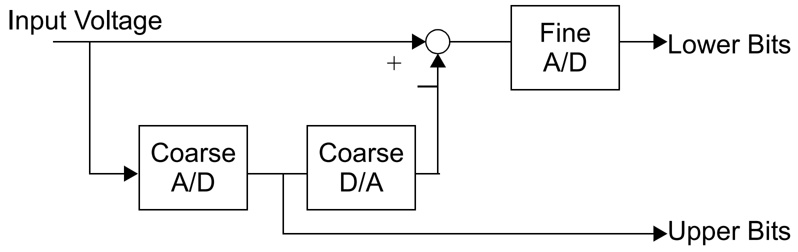
```
initial begin
    for (i = 0; i < 8; i = i+1) begin
        dout[i] = 0;
    end
    refs[0] = (1.0+(($random % 65536)/65536.0)∗0.4)∗vref/256;
    for (i = 1; i < 255; i= i+1) begin
        refs[i] = (1.0+(($random % 65536)/65536.0)∗0.4)∗vref/256 + refs[i–1];
    end
    refs[255] = vref;
    for (i = 0; i < 255; i= i+1) begin
        // Delay is in ns, if all modules use ‘timescale 1ns / 1ps
        del[i] = (1.0)∗10.0 + ($random % 65536)/65536.0)∗10;
        if (del[i] < 0.0) del[i] = -del[i];
    end
end

always @(posedge(clk)) begin
    unconverted = V(vin);
    refval = 0;
    count = 0;
    for (i = 0; i <= 255; i = i+1) begin
        overdrive = unconverted – refs[i];
        if (overdrive < 0.0) begin
            overdrive = –overdrive;
        end
    if (overdrive < vref/1024) begin
        // very simple metastability model
        if ($random > 0) dout[i] <= #(del[i]) 1;
        else dout[i] <= #(del[i]) 0;
    end else if (unconverted > refs[i]) begin
        dout[i] <= #(del[i]) 1;
    end else begin
        dout[i] <= #(del[i]) 0;
    end
end
```

the delays assigned and a very simple metastability model incorporated. The delays are generated in the initial block. Metastability is modeled by looking for comparator input voltages that are less than 0.25 of a step, and then assigning the output randomly.

3.4 Two-Stage A/Ds

Modeling of the flash A/D converter focused basically on mismatch failures and dynamic effects. Flash A/D converters are also used as building blocks for Two-Stage or subranging A/D converters. Behavioral modeling of a two-stage A/D converter is more challenging and can be used to study potential problems. A two-stage A/D converter makes use of 2 flash A/Ds, a D/A, and a differencing amplifier. A basic block diagram is shown in Figure 3.

FIGURE 3 *A block diagram showing a two-stage flash A/D converter.*



The key advantage of the two-stage architecture is the reduction in overall hardware. An 8-bit flash required 255 comparators. A two-stage might have 4 bit flash A/D converters in each stage leading to 30 comparators. However, there are now new problems to study. The linearity of the first 4-bit A/D and the D/A needs to be as good as the overall converter. Note that if a Flash A/D converter is used, the resistor string can be used for the D/A function. If the linearity in the first A/D converter is not as good as the overall converter, then the residue created for the second stage could lead to errors. A second problem for the two-stage design is the need for synchronization between the first conversion and the second sub-conversion. Both stages need to convert the same input signal. If the input signal is moving during conversion, it is possible for the voltage to change before it reaches the differencing amplifier causing a skew in time, leading to an error in voltage. This problem can be remedied by using a sample and hold or an analog delay line.

3.5 Folding and Averaging

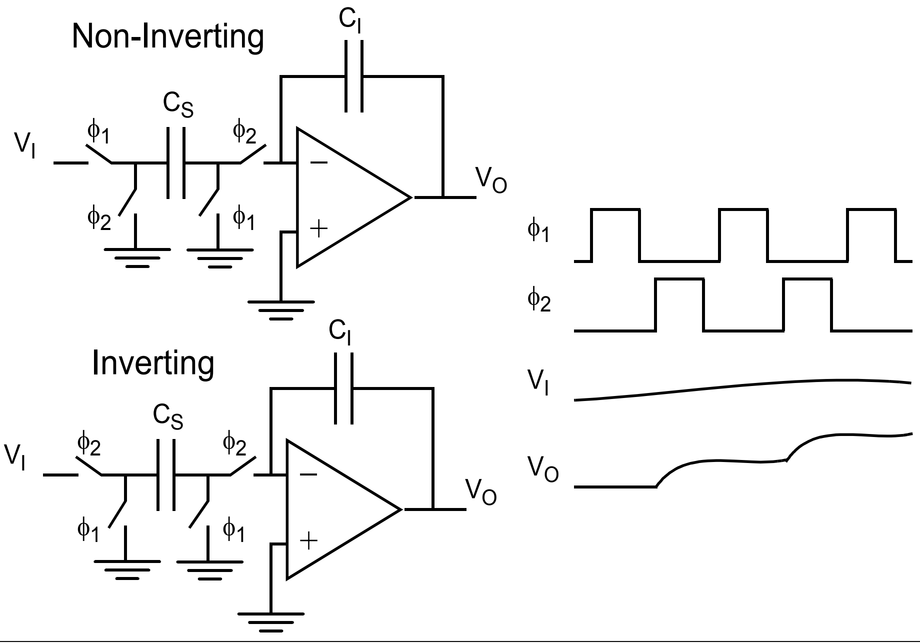
A recent technique for improving the design of flash type converters is the use of folding and averaging. This is beyond the scope of this paper. A future version of this paper may add models for these types of converters.

4 Modeling Switched-Capacitor Integrators and Gain Stages

Switched-capacitor (SC) circuits can provide lower cost implementations for A/D converters in standard digital CMOS technologies. Accurate modeling of the SC integrator is important for prediction of overall A/D behavior. This section will provide the background on SC circuit modeling necessary for studying SC implementations of pipelined and oversampling A/D converters. In addition, the SC modeling approach is flexible enough to be used for modeling SC filters built from integrator blocks. The techniques described in this section provide efficient modeling capabilities that can also incorporate sampled-data noise effects.

A switched capacitor integrator consists of an op-amp, switches, and capacitors as shown in Figure 4. Some different ways to model an SC integrator will be discussed in

FIGURE 4 *Two switched-capacitor blocks and a timing diagram.*



this section. The models will start out at a high-level and move down to more detailed models.

4.1 Switched Capacitor Integrator Analysis.

This section develops a high level model for the integrator that can be quite accurate for design exploration work, while still maintaining very fast simulation times. A switched-capacitor integrator can be analyzed using charge conservation equations that can be coded using an event-driven formulation. This provides one of the fastest methods for simulating switched-capacitor circuits with the potential for very high accuracy, depending on which second-order effects dominate. An added advantage of this model is the relative ease for including sampled noise effects.

4.2 Difference Equation Analysis

Consider the circuits shown in Figure 4. Both inverting and non-inverting SC integrators are shown. Both circuits have the same timing diagram. It is assumed that the overall clock period is defined as *T*. Consider first the non-inverting integrator. When ϕ_1 is asserted, the input is sampled and a charge is placed on C_s. During ϕ_2, the charge is dumped on to the integrating capacitor. Ideally, this is modeled using the following equation.

$$C_I V_O(nT) = C_I V_O(nT - T) + C_S V_I\left(nT - \frac{T}{2}\right) \tag{1}$$

By using a discrete time representation, (1) can be rewritten as a difference equation. When using a difference equation, it is assumed that the sampling is uniform (i.e. there is a fixed sampling rate). Rather than using the continuous-time argument *nT*, a single variable n is used. The use of brackets rather than parentheses also indicates that a sampled-date difference equation is being used.

$$V_o[n] = V_o[n-1] + \frac{C_S}{C_I} V_I\left[n - \frac{1}{2}\right] \tag{2}$$

Recalling that *Q=CV*, the equation can then be coded as shown in Listing 4. In this

LISTING 4 *A model for a non-inverting SC integrator.*

```
module noninvert_integ(out, in, phi1, phi2);
input in, phi1, phi2;
voltage in;
output out;
wreal out;
wire phi1, phi2;

parameter real Cs = 1p from (0:inf);
parameter real Ci = 1p from (0:inf);
parameter real agnd = 1.65 from (0:inf);
real state, qs;

initial begin
    qs = 0.0;
    state = 0.0;
end

always @(negedge(phi1))
    qs = Cs*(V(in)–agnd);

always @(negedge(phi2))
    state = (qs + Ci*(out–agnd))/Ci;

assign out = state + agnd;
endmodule
```

model, the input *in* is defined as a voltage, but the output is defined as *wreal*. For SC circuits, it is not necessary to use electrical or voltage variables since the capacitors force charge conservation. Thus *wreal* nets can be used. The equation was broken up along the two phases and charge variables are defined. The model uses two physical clock inputs to fully model the effects of half-cycle delays in sampling. As with the real circuit, the final value of the input signal is sampled on the falling edge of ϕ_1. Note that unlike the real circuit, the output goes valid on the falling edge of ϕ_2 and remains valid until the next falling edge of ϕ_2. In order for a subsequent stage to correctly sample the output of this stage, the two clocks need to be non-overlapping, just as in the real circuit.

Also note that in this case, the signal is processed around a value of analog ground that is supplied as a parameter.

For the inverting integrator, the input is directly coupled to the output during ϕ_2. The difference equation can be derived as shown in (3).

$$V_o[n] = V_o[n-1] - \frac{C_s}{C_I} V_i[n] \tag{3}$$

The equation can be implemented as a model as shown in the listing below.

LISTING 5 *A model for an inverting SC integrator.*

```
module noninvert_integ(out, in, phi1, phi2);
input vin, phi1, phi2;
voltage in;
output out;
wreal out;
wire phi1, phi2;

parameter real Cs = 1p from (0:inf);
parameter real Ci = 1p from (0:inf);
parameter real agnd = 1.65 from (0:inf);
real state, qs;
initial begin
      qs = 0.0;
      state = 0.0;
end

always @(negedge(phi2)) begin
      qs = Cs∗(V(in)–agnd);
      state = (qs + Ci∗(out–agnd))/Ci;
end

assign out = state + agnd;
endmodule
```

Note that in the model, the processing only occurs on the falling edge of ϕ_2. Additional information on deriving difference equations and implementing them as models can be found elsewhere [4].

4.3 Modeling the Effects of Finite Op-Amp Gain

The non-inverting integrator model can be easily extended to include second order effects. To model finite gain in the op-amp, the difference equations must be modified. Finite gain will create a residual voltage on the sampling capacitor C_S at the end (falling edge) of *$\phi 2$*. If the open loop gain of the op-amp is A_{OL}, the magnitude of the residual voltage can be approximated by V_O/A_{OL}. Incorporating this change in the charge conservation equation gives:

$$G = 1 + \frac{1}{A_{ol}} \tag{4}$$

$$GC_IV_o[n] = GC_IV_o[n-1] + C_S\left(V_I\left[n-\frac{1}{2}\right] - \frac{V_o[n]}{A_{ol}}\right) \tag{5}$$

Solving for $V_O[n]$ gives:

$$V_o[n] = \frac{A_{ol}+1}{A_{ol}+\frac{C_S}{C_I}+1}V_o[n-1] + \frac{C_S}{C_I}\frac{A_{ol}}{A_{ol}+\frac{C_S}{C_I}+1}V_I\left[n-\frac{1}{2}\right] \tag{6}$$

The core of the model to implement (6) appears as Listing 6.

LISTING 6 *Code implementing an integrator with finite gain.*

```
parameter real Cs = 1u from (0:inf);
parameter real Ci = 1u from (0:inf);
parameter real agnd = 1.65 from (0:inf);
parameter real Aol = 1e6 from (0:inf);
real state, qs, qi;

initial begin
    qs = 0.0;
    state = 0.0;
end

always @(negedge(phi1)) begin
    qs = Cs*(Aol/(Aol+(Cs/Ci)+1.0))*(V(vin)–agnd);
end

always @(negedge(phi2)) begin
    qi = state*((Aol+1.0)/(Aol+(Cs/Ci)+1.0))*Ci;
    state = (qs + qi)/Ci;
    qs = 0;
end

assign vout = state + agnd;
```

4.4 Modeling the Effects of Non-linearities

Nonlinearity can be modeled in a similar fashion. The three major sources of nonlinearity in a switched capacitor integrator are signal-dependent charge injection from the switches, nonlinearity in the capacitors, and nonlinearity in the op-amp gain function. To illustrate techniques for modelling nonlinearity, capacitor nonlinearity will be considered and can be modeled by using a Taylor Series Expansion about a nominal operating point.

$$C(v) = C_0(1 + \alpha_1 V + \alpha_2 V^2 + \ldots) \tag{7}$$

For simplicity, assume only the first term of the Taylor Series is retained and that the nominal operating point is V=0. The charge conservation equation can now be written in two equations as:

$$g(V[n]) = 1 + \alpha_1 V[n] \tag{8}$$

$$C_I g(V_o[n])V_o[n] = C_I g(V_o[n-1])V_o[n-1] + C_S g\left(V_i\left[n-\frac{1}{2}\right]\right)V_i\left[n-\frac{1}{2}\right] \tag{9}$$

Solving for $V_O[n]$ gives:

$$V_o[n] + \alpha_1(V_o[n])^2 = V_o[n-1] + \alpha_1(V_o[n-1])^2 + \frac{C_s}{C_I}\left(V_i\left[n-\frac{1}{2}\right] + \alpha_1\left(V_i\left[n-\frac{1}{2}\right]\right)^2\right) \tag{10}$$

The difference equation is no longer linear and could be solved by iteratively choosing a value for $V_O[n]$ until the solution converges. Since the system is oversampled, efforts must be made to simplify the evaluation so that compute times are not excessive. The initial choice in the iteration is obtained by assuming that $\alpha_1(V_O[n])^2$ is 0. Since α_1 is usually much less than 1, this term will be close to 0. A first-order simplification is to calculate $V_O[n]$ under these assumptions, then subtract the $\alpha_1(V_O[n])^2$ term out to obtain an estimate for $V_O[n]$. This will allow the integration loop to consist of in-line code without the need for extra iterations. Experience has shown that this simplification does not adversely affect the simulation results if the linearity is small. A code listing that implements the equation is shown in Listing 7.

LISTING 7 *An implementation of an integrator with capacitor non-linearity.*

```
parameter real nlci = 1e–6 from (0:inf);
parameter real nlcs = 1e–6 from (0:inf);
real state, qs, qi;

initial begin
    qs = 0.0;
    state = 0.0;
end

always @(negedge(phi1)) begin
    qs = cap_in*(1.0+nlcs*(V(vin)–agnd))*(V(vin)–agnd);
end

always @(negedge(phi2)) begin
    qi = state*(1.0+nlci*state)*cap_fb;
    state = (qs + qi)/cap_fb
    state = state – nlci*state*state;
    qs = 0;
end

assign vout = state + agnd;
```

Note that different non-linearity coefficients are used for the sampling and feedback capacitors.

It is possible to use piecewise linear approximations for an op-amp transfer function to implement non-linearity. This allows for stronger non-linearities than given by a Taylor Series approximation. The application of this modeling technique is left to the reader. The author’s experience has shown this can be very effective for modeling harmonic distortion.

4.5 Combining Finite Gain and Capacitor Nonlinearity

The effects of finite gain and nonlinearity can be combined. A model can be derived that combines both finite gain and capacitor nonlinearity. The model will not be presented here due to space limitations.

4.6 Parasitic Input Capacitance of the Op-amp

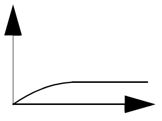
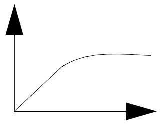
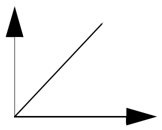
The parasitic capacitance at the input of the op-amp can be quite large. In some op-amps, the parasitic can be of the same magnitude as C_S or C_I. The effect of the parasitic capacitance can be analyzed using difference equations. Under the assumption of finite op-amp gain, charge redistribution will leave charge on both the sampling capacitor and the parasitic at the input of the op-amp. This diminishes the overall apparent gain of a switched capacitor integrator stage.

4.7 Integrator Settling

Settling time effects of the op-amps should also be carefully modeled since they can provide a signal dependent offset when the settling is slew limited. A simple settling model can be implemented by determining the ideal output, and then applying rules to determine how well the op-amp settles. A simple single-pole op-amp model is characterized by a time constant.

For a more advanced model, the settling period can be divided into a slewing region and a linear settling region. A settling model with this concept has been used in simulating oversampling A/D converters [5]. This simplified model does not account for the initial capacitive feed-through in the integrator, but does provide a model for simulation. In this model, there are three regions of operation, as shown in Figure 5.

FIGURE 5 *Definitions of regions for settling time models.*

Time Domain View	Region of Operation	Final Simulation Output Value
	linear settling $V_O \le \tau S_R$	$V_O\left(1 - exp\left(-\frac{T_S}{\tau}\right)\right)$
	Combination slewing with some linear settling $\tau S_R < V_O \le (T_S + \tau)S_R$	$V_O - \tau S_R exp\left(\frac{V_O}{\tau S_R} - 1 - \frac{T_S}{\tau}\right)$
	Slew limited $V_O > T_S + \tau$	$T_S S_R$

If the integrator were ideal, the output would change by an amount V_o. It is assumed that the time constant τ of the single-pole system is less than the sampling period T_S and the integrator has a maximum slew rate S_R. When the output step is small, the integrator should settle with a one pole response. When slew limiting occurs, there will be a period of slewing until the output is close to the final value, and then a time of linear settling. In this model, the cutoff point between these regions is chosen to be at the time τ. When there is slewing in the model, the output slews to $1/(\tau S_R)$ of the final value and then settles linearly in the remaining time. If the output is fully slew limited, then the output can only change by the amount $T_S S_R$ regardless of V_o.

This model can be implemented together with the other models for nonlinearity and finite op-amp gain. The quantity V_o is calculated based on the equations developed previously for the integrator. The region of operation is then determined based on the inequalities given in Figure 5, and the integrator output is calculated based on the output value, also given in the figure. The same definitions could be made for a two-pole model of settling.

4.8 Integrator Clipping

Switched-capacitor integrators are limited in swing by the compliance of the op-amp. Implementing this in simulation requires only a minor addition. The output of the integrator must be monitored and a hard limit applied when the output goes beyond a fixed range. This should be done after the correction for settling effects. Extra features could be added to model the soft saturation effects normally found in op-amps.

4.9 Continuous Time Models of the SC Integrator

The difficulty in modeling the settling time of the integrator suggests that a continuous time model would provide advantages in studying the effects of settling time and nonlinear settling. However, there will be a significant trade-off in speed for accuracy. A continuous time model would require the use of the analog solver. The previous models only had an overhead of solving an equation and provided the voltage which is valid at the falling edge of the clock. The analog solver would require the calculation of the entire trajectory of the op-amp settling. For simulation of a circuit like an Δ–Σ modulator, the discrete time models are favored to provide faster simulation.

5 Modeling Noise in Switched-Capacitor Circuits

During the sampling operation in switched-capacitor circuits, the resistive noise of the sampling switch is captured on the sampling capacitor. The noise power is inversely proportional to the size of the capacitance. Thus in A/D converters, there typically is a size vs. noise trade-off. The effect of sampled data noise can easily be incorporated in the simulation model. This section outlines how thermal and $1/f$ noise can be added to the basic SC integrator model.

5.1 Generating White Noise

White noise has a flat power spectral density (PSD). The total noise power of the signal is determined by integrating the PSD over the frequency range of interest. Since the

PSD is the Fourier transform of the autocorrelation, a flat PSD indicates that the noise samples are uncorrelated. Generation of a time-domain series representing white noise is equivalent to generating a time sequence of uncorrelated random numbers. Gaussian distributions are used since they are fully characterized by first and second order statistics.

For switched-capacitor integrators, white-noise generators are used to model the noise in the switches. As an example, consider the noise in a sampling switch connected to a capacitor. The noise power associated with the switch resistance and the sampling capacitor is kT/C with units of V^2/Hz, where k is Boltzmann's Constant, T is the absolute temperature, and C is the size of the capacitor in Farads. The value of the voltage noise power is independent of sampling frequency due to aliasing during the sampling process.

To add noise to a Verilog model, the $random system task can be used. The $random system task must be initialized with a seed value prior to the first call. It returns a 32-bit integer value. The range can be set by using a modulo integer. For example, ($random % *b*) will create a random integer in the range [$-b+1$, $b-1$]. The modified code for the input sampling portion of the non-inverting integrator is shown in Listing 8.

LISTING 8 *Modeling capacitor sampling with additive kT/C noise.*

```
// Boltzmann's constant is defined as P_K in constants.h
'include "constants.h"
parameter integer ival = 65536 from (0:inf);
parameter real fval = 65536.0 from (0:inf);
parameter T = 300.0 from (0:inf);

always @(negedge(phi1)) begin
    qs = cap_in*((V(vin)–agnd)+ sqrt(P_K*T/cap_in)*($random%ival)/fval);
end
```

The input voltage is modified by the standard deviation of the noise multiplied by a random variable normalized to be in the range (-1, 1).

5.2 Generating 1/f Noise

Although $1/f$ noise is found throughout the natural world, algorithms for $1/f$ noise generation are not as pervasive. Modeling the noise near DC for A/D converters can be important since this noise can dominate the overall baseband noise floor if not well controlled. A promising approach for $1/f$ noise generation is to use the summation of Lorentzian spectra. A Lorentzian spectrum is created by filtering white noise with a one-pole filter. This approach has been used in instrumentation to generate continuous time $1/f$ noise over a specified range of frequencies. It has been shown that a constant distribution of 1.4 poles per decade gives a $1/f$ spectrum with less than 1% error [6]. By changing the pole distribution, $1/f^{\gamma}$ noise can be modelled. This model allows simulation of $1/f$ noise in over a frequency region which can be defined to include the areas near DC.

For discrete time, the spectral density can be approximated using sampling concepts. (11) gives the power spectral density for summation of Lorentzian spectra. S(f) only has an approximate $1/f$ spectra over a range defined by the values of ϕ_n.

$$S(f) = \kappa \sum_{n=1}^{N} \frac{\varphi_n}{f^2 + \varphi_n^2} \approx \frac{\kappa}{f} \text{ for } \varphi_1 < f < \varphi_N \tag{11}$$

The magnitude response *H* for a Lorentzian spectrum and the power spectral density for white noise filtered by *H* are given in (12).

$$H(j2\pi f) = \frac{\sqrt{p}}{j2\pi f + p} \qquad S(f) = |H(jf)|^2\sigma^2 = \frac{p\sigma^2}{f^2 + p^2} \tag{12}$$

The DC gain for the filter must be adjusted to give the Lorentzian spectrum.

In order to incorporate 1/*f* noise in a discrete time simulation, noise samples need to be generated. Since 1/*f* noise has emphasized low frequency components, the aliasing effects due to the sampled approximation are minimized. One way to obtain a discrete time approximation by using the *s*-to-*z* mapping of (13) where T_S is the sampling period.

$$s = \frac{1 - z^{-1}}{T_S} \tag{13}$$

This represents backward-Euler integration and provides adequate results at the low frequencies of interest. If higher frequencies need to be considered, the bilinear s-to-z mapping could be used. The approximate magnitude response H' for the filter H is given in (14).

$$H'(z) = \frac{\dfrac{T_S\sqrt{p}}{1 + T_S p}}{1 - \dfrac{1}{1 + T_S p}z^{-1}} = \frac{\dfrac{\sqrt{T_S}\sqrt{x}}{1 + x}}{1 - \dfrac{1}{1 + x}z^{-1}} \text{ where } x = T_S p \tag{14}$$

The corresponding power spectral density when white noise is filtered is given in (15).

$$S'(f) = \frac{T_S\left(\dfrac{\sqrt{x}}{1 + x}\right)^2\sigma^2}{1 - 2\left(\dfrac{1}{1 + x}\right)\cos\left(\dfrac{2\pi f}{f_S}\right) + \left(\dfrac{1}{1 + x}\right)^2} \tag{15}$$

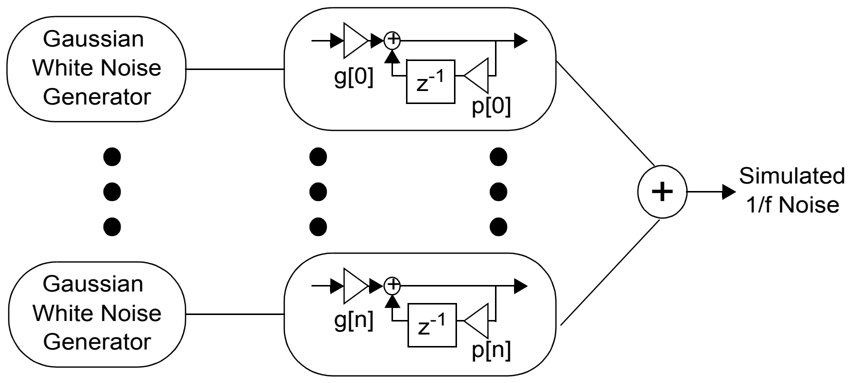
Substituting this result into (11) gives (16) which is the power spectral density of the proposed 1/*f* noise generator.

$$S'(f) = \sum_{n=1}^{N} \frac{\left(\dfrac{x}{1 + x}\right)^2}{1 - 2\left(\dfrac{1}{1 + x}\right)\cos\left(\dfrac{2\pi f}{f_S}\right) + \left(\dfrac{1}{1 + x}\right)^2}\sigma^2 T_S \tag{16}$$

In order to match the magnitude of the spectral density *S'(f)* with the continuous time counterpart, the input white noise generators must have the same noise power. For this to occur, the variance or noise power in a certain bandwidth must be set equal. In this case, the input white noise generators in the continuous time model of (11) have spectral density κ. In the bandwidth f_S, they will have noise power κf_S. Thus in the discrete time simulation, the proper value to use for κT_S is the noise power of the continuous time white noise generators, which in this case is $\kappa f_S T_S = \kappa$, a quantity that is independent of frequency.

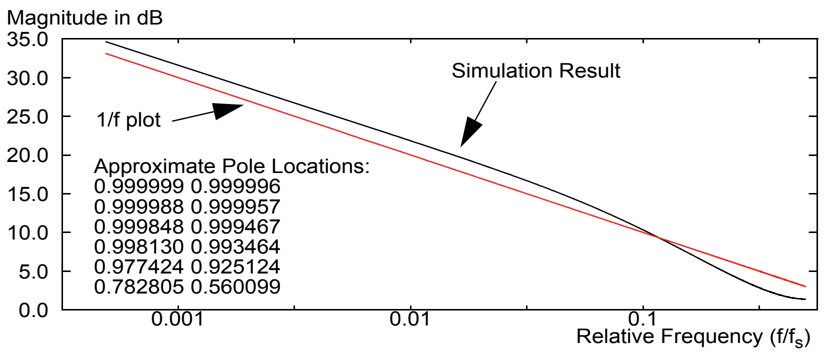
The noise generator can be implemented as shown in Figure 6. The frequency response

FIGURE 6 *Summation of approximate Lorentzian spectra to obtain a 1/f noise model.*


Gaussian White Noise Generator
g[0]
z^{-1}
p[0]
Gaussian White Noise Generator
g[n]
z^{-1}
p[n]
Simulated 1/f Noise

of one implementation is shown in Figure 7 along with the 12 approximate pole values

FIGURE 7 *Frequency response of a 1/f noise generator.*



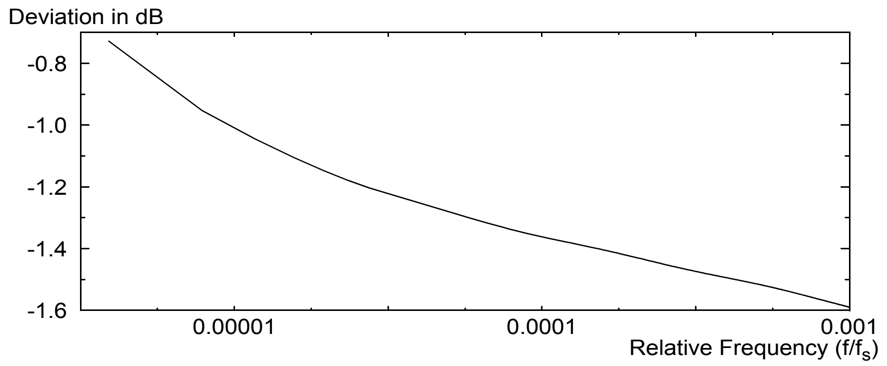
used in the generator model. In addition, a plot of the deviation from a true $1/f$ response shown in Figure 8. It shows little variation over the low frequency range of interest. The downward trend indicates that the simulation model provides noise with a $1/f^{\nu}$ characteristic where ν is slightly less than 1. There is little ripple since a high pole density was used across the interval.

To set the value of variance for the input white noise generator, consider the example of a MOS transistor. The equivalent input-referred noise can be modelled as in (17).

$$v_{eq}^2 = \frac{K_f}{WLC_{ox}f} \tag{17}$$

The parameter $\sigma^2 T_S$ should be set equivalent to the factor $K_f/(WLC_{ox})$ in magnitude. For the simulation, the output of the $1/f$ noise generator is placed so that it adds to the

FIGURE 8 *Deviation between the true 1/f response and the simulation result.*



input signal of the MOS transistor. It is interesting to note that a set of samples from the 1/*f* noise generator does not give any information about the sampling rate. In fact, a single set of samples can be used as 1/*f* noise regardless of the sample rate. This scalability is a characteristic of 1/*f* noise. It will look the same no matter what resolution is used. This phenomenon has been exploited in the generation of fractal images.

A listing for code to generate 1/f noise is shown in Listing 9. Both flicker and white noise is added to create the output sample. An output sample is created every time the signal *clk* has a rising edge. The pole locations for the Lorentzian spectra are calculated in the initial block. This could have been hard-coded as numerical values just as easily as calculating them in the initial block. Samples from this code could be added to voltage samples in an actual circuit model.

Several theories have been advanced for the cause of 1/*f* noise in MOSFETs. A widely accepted view attributes it to generation-recombination noise at the silicon-oxide interface where the traps are characterized by a distribution of time constants. The model simulates this physical mechanism since a distribution of time constants gives rise to something similar to a summation of Lorentzian spectra. Analysis using these assumptions gives rise to (17) for a MOSFET which shows that 1/*f* noise is inversely proportional to gate capacitance.

With the basic understanding of switched-capacitor circuits, it is now time to focus on the actual converter architectures.

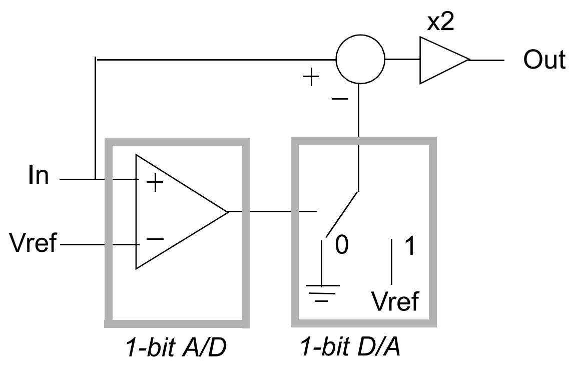
6 Algorithmic and Pipelined A/D Converters

Flash converters use 2^{n-1}parallel comparisons to obtain a fast *n*-bit A/D conversion. The speed is limited by the comparison time and the encoding time, and conversion is achieved in one clock cycle. Another way to achieve A/D conversion is to use an algorithmic approach. The functional example of Section 2 is an example of an algorithmic approach where it may take many operations and clock cycles to achieve the A/D conversion. The approach used in the functional example is essentially how algorithmic and pipelined converters work. Figure 9 shows a circuit that would implement one pass through the code.

LISTING 9 *Verilog-AMS code for generating sampled 1/f noise and white noise.*

```
module fwnoise(clk, out);
input clk;
wreal out;
real high, low, tlow, interval, fout;
integer numPoles, seed, i, j, number;
reg [11:0] poles, gain, state;
parameter real wnStdDev = 1e-6 from [0:inf);

initial begin
    seed = 32;
    high = 1;
    low = 0.000001;
    number = 12;
    tlow = $log(low);
    interval = ($log(high) - tlow)/(number-1);
    for (i = 0; i < number; i = i + 1) begin
        poles[i] = $atan($pow(10.0, tlow + interval∗i));
        poles[i] = $atan($pow(10.0, tlow + interval∗i));
        gain[i] = $sqrt(poles[i]);
        poles[i] = 1.0/(1.0 + poles[i]);
        // initialize the states
        state[i] = 0.0;
    end
end

always @(posedge(clk)) begin
    out = 0.0;
    // Calculate the 1/f noise sample
    for (j = 0; j < number; j=j+1) begin
        fout = state[j] + $rdist_normal(seed, 0.0, gain[j]);
        state[j] = fout ∗ poles[j];
        out = out + fout;
    end
    // add in white noise
    out = out + $rdist_normal(seed, 0.0, wnStdDev);
end
endmodule
```

FIGURE 9 *One cycle of an algorithmic A/D conversion.*



A flash conversion is first performed on the input. In IC applications, the conversion is typically of low resolution, from 1 to 4 bits. A D/A operation is performed using the result and then subtracted from the input signal. The resulting voltage is called a residue. The residue is then amplified by a fixed amount. In the simplest case, the amplification factor is 2^n, where n is the number of bits for the A/D converter. This circuit produces n bits of the conversion, and restores the signal level so the next stage receives an input that can vary over the full scale of the reference voltages.

There are two ways to use this circuit. If a sample and hold circuit is added, the result can be recirculated through the same circuit. This is what is known as a cyclic or algorithmic A/D converter. The same hardware is used to do successive iterations of the conversion algorithm. If one bit is created per iteration, it takes n iterations to achieve an n-bit conversion. Another option is to create several instances of the stage and operate them in a pipelined fashion. If one bit is created per stage, it still takes n cycles to achieve an n-bit conversion, but the output rate can still be n bits per cycle with n cycles of latency. For the remainder of this section, the discussion will focus on the pipelined version of the converter.

Note that each stage in a pipelined converter creates bits that must be assembled in a time-staggered fashion. The digital logic needed to capture the bits is modest. Digital correction schemes can be implemented to ease design constraints. A digital correction method for comparator offset will be illustrated in Section 6.2.

The pipeline converter has proven to be a power efficient, reliable, and expandable converter for 8 to 10 bit conversion in the 10 to 40 MHz range. These specifications made it suitable for consumer analog video applications. It can be implemented in a standard digital CMOS process providing a low cost solution that can be integrated with other digital circuits. Techniques have been developed to add resolution at the expense of speed or to increase the speed through parallel pipelines.

6.1 Modeling the Pipelined Converter

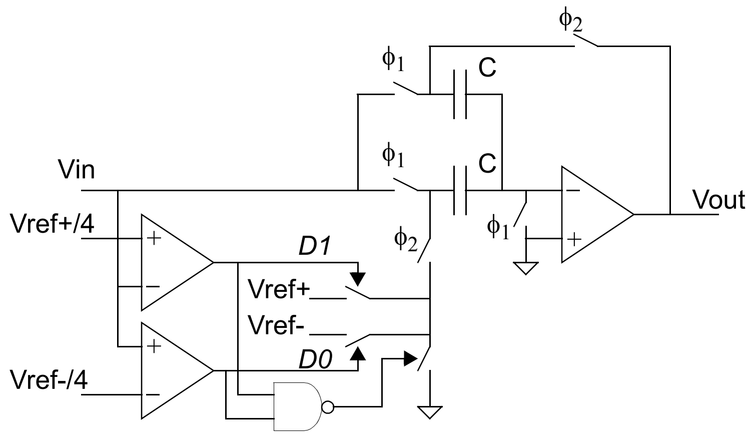
The functional model of Listing 1 can be used to model the high-level behavior of a pipelined converter. In order to model the second order effects of the converter, a more structural approach needs to be taken. The model can be built around the basic building block description of the converter shown in Figure 10. The blocks shown in this figure will be discussed in more depth in the next few sections. The structural approach allows freedom in substituting more detailed models when non-idealities need to be studied. The following sections will focus on behavioral modeling of switched capacitor integrators and on a technique for digital correction for comparators.

6.2 Digital Correction for Comparator Offset

Before creating the full model of a pipelined converter, the issue of digital correction will be discussed. As mentioned previously, each stage of the pipeline is designed to convert a number of bits. The design of the comparators used in the Flash A/D converter can be quite difficult as discussed previously. A missed decision early in the pipeline will destroy the conversion and may lead to large code errors.

As mentioned in Section 3.3, the comparator offset needs to be smaller than the stepsize of the flash converter. However, when used in the pipeline converter, the step-size is not set by the stage A/D resolution but by the overall resolution of the converter down-

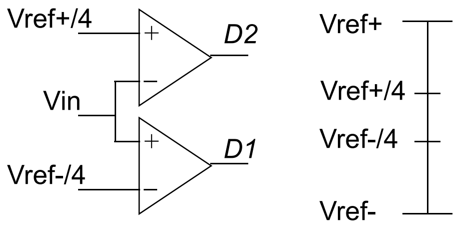
FIGURE 10 *A switched capacitor pipeline stage providing 1.5 bits per stage.*



stream from the current stage. For example, the resolution of the first stage conversion of an 8-bit converter must be accurate to the 8 bit level even though the first stage flash may only provide 2 bits. The comparator gain also must be large enough to cause a full scale output transition at less than a step-size of overdrive. The speed of the comparator is important, because the longer the worst case evaluation time, the less time available to the switched-capacitor integrator for settling.

To overcome this problem, designers have favored the use of a digital correction scheme for comparator offset. It is typically implemented by using a scheme that provides 1.5 bits per stage in the pipeline. More detailed descriptions of this scheme can be found in the literature [7]. Basically, two comparators are used to give three possible output states or 1.5 bits as shown in Figure 11. Two of the three states cause the output or con-

FIGURE 11 *Comparators and states for encoding 1.5 Bits per stage conversion.*



D2D1	Conv. Bit	Correction Action
0 1	1	no correction
1 1	0	no correction
1 0	0	subtract 1

version bit to be set to 1 or 0 for the stage. The third state indicates that an error was made in a previous stage and a correction of subtracting by 1 is needed for the output code. Theoretical analysis shows that under this scheme, the comparators can have fairly large offsets (up to +/−*Vref*/4) while still providing accurate conversion. The net result is that fast and simple latch type comparators can be used in place of high precision comparators. This eases the design constraints and provides a solution that has lower power and area.

The digital correction scheme not only simplifies the analog design task but also the behavioral modeling in most cases. Since the latched comparators are used, the behavioral model can again avoid the use of the analog block. The listing for a comparator implementing the comparison to *Vref*+/4 is shown in Listing 10. Note that in the model,

LISTING 10 *Verilog AMS description of a latched comparator.*

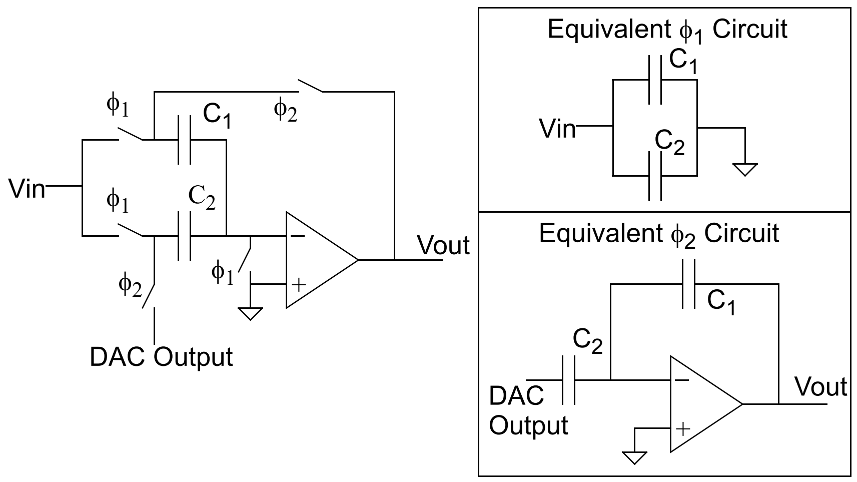
```
module compinp (inp, inm, clk, dout);
input inp,inm,clk;
output dout;
reg dout;
wreal inp, inm;
parameter real vcc = 3.3 from (0:inf);
parameter real tdel = 2.0 from (0:inf);
real vagnd;

initial begin
    dout = 0;
    vagnd = vcc/2.0;
end

always @(negedge(clk)) begin
    if (inp > 0.25*(inm–vagnd) + vagnd) begin
        #tdel
        dout = 1;
    end else begin
        #tdel
        dout = 0;
    end
end

// model precharge reset
always @(posedge(clk))
    dout = 0.0;

endmodule
```

wreal nets are used for inputs and outputs, avoiding the use of electricals since conservation of charge is not needed. Also note that the input *inp* is the input signal, while *inm* is a full scale reference voltage. The input signals are assumed to be positive and referenced to an analog ground, defined to be *Vcc*/2.

6.3 Modeling the SC S/H and Gain Stage

For the 1.5 bits/stage architecture, the interstage amplifier must have a gain of two. The basic gain stage is shown in Figure 12. In the figure, the equivalent circuits for phase 1 and 2 of the clock are shown. It is assumed that the two capacitors are nominally equal in value. In the first phase, the input is sampled on the two capacitors. In the second phase, two things happen. If the DAC output were analog ground, all of the charge on capacitor C_2 would flow onto C_1, and an exact multiply by two would be achieved. By superposition, the DAC output voltage subtracts from the output result because the SC circuit is in an inverting configuration. Thus the circuit as shown achieves multiply by two, sample and hold of the input on the falling edge of ϕ_1, and summation of the DAC voltage.

FIGURE 12 *Block diagrams illustrating an SC interstage gain and a S/H amplifier.*



A code listing for a model of this circuit is shown in Listing 11. Note that as in previous models, wreal nets are used for inputs and outputs. The DAC is decoded within the model based on the two bits that are generated by the comparators. The comparator encoding scheme leaves the possibility of one unused state. In this case, an error message is printed to alert the user.

6.4 Digital Logic

A simple model for the digital logic can be created by storing for each stage the D0 bit and a bit indicating if correction was needed. A staggered scheme is needed to ensure that all the bits are time synchronized. At the end of the pipeline, there should be two *n*-bit registers, one with the D0 bits and one with the correction bits. A simple subtraction suffices to create the final corrected conversion word. The output will be conveniently represented in two's complement arithmetic. A code listing implementing this for an 8-bit converter is shown in Listing 12.

6.5 A Complete Pipeline A/D Model

A complete pipelined A/D model is created by combining the modules shown in the previous sections. A testbench can then be created and the model exercised. Note that in the last stage, the final bit is generated by comparison using a single comparator.

6.6 Modeling Non-Idealities in the Pipelined Converter Stage

The basic 1.5 bits per stage model can be extended to handle various non-idealities. A few of these will be examined in the following sections.

6.6.1 Gain Errors

The architecture relies on a precise multiplication by two to achieve accurate conversion. If the stage gain deviates from exactly two, there will be an error in the conversion. A small amount of error can be tolerated and the error can be larger in later stages of the

LISTING 11 *SC interstage gain amplifier model.*

```
'include "disciplines.h"
// Input sampling on phi1, DAC and integration take place on phi2
module scgain (in, out, refp, refm, d1, d2, phi1, phi2);
input in,phi1,phi2,d1,d2;
output out;
wreal in, out, refp, refm;
wire d1, d2;

parameter real vcc = 3.3 from (0:inf);
parameter real cap_in = 1.0 from (0:inf);
parameter real cap_fb = 1.0 from (0:inf);
parameter real tdel = 1n from (0:inf);
real vagnd, qc1, qc2, qc1p2, state;

initial
    vagnd = vcc/2.0;

always @(negedge phi1) begin
    qc1 = (in–vagnd)∗cap_in;
    qc2 = (in–vagnd)∗cap_fb;
end

always @(negedge phi2) begin
    if (!d1 && d2) begin
         qc1p2 = –(refp–vagnd)∗cap_in;
    end else if (!d1 && !d2) begin
        qc1p2 = 0.0;
    end else if (d1 && !d2) begin
        qc1p2 = –(refm–vagnd)∗cap_in;
    end else begin
        // error state in decoding
        qc1p2 = 10.0;
    end
    state = (qc1 + qc2 + qc1p2)/cap_fb + vagnd;
end

assign out = state;
endmodule
```

pipeline. The exact size of the error is a function of the overall converter resolution [8]. The gain is controlled by the capacitor variations. In the model, the gain of two is achieved by using two equally sized capacitors to sample the input and then combining the charge on one of the capacitors to achieve a gain of two. Capacitor variations depend on the physical size of the capacitor and the precision of etching in the process. A large capacitor will have less variation than a small capacitor. To model the gain error, a deviation can be added to the capacitor values. This can be done during the initial step using **$random**, or it can be hard-coded into the model.

6.6.2 Differential Modeling

The model shown in this example is set up for single ended circuit operation. A differential model could also be used but there are several difficulties with implementing a differential model. A differential signal requires a common mode operating point. One method of implementing this is to use two single ended circuits operating off of a com-

LISTING 12 *Digital logic for an 8-bit pipelined conversion.*

```
module correction(s, c, out, clk);
input [6:0] s, c;
output [7:0] out;
input clk;
reg s1, c1;
reg [1:0] s2, c2;
reg [2:0] s3, c3;
reg [3:0] s4, c4;
reg [4:0] s5, c5;
reg [5:0] s6, c6;
reg [6:0] s7, c7;
reg [7:0] s8, c8, out;
reg tmp;

initial
    tmp = 1'b0;

always @(posedge clk) begin
    s8[7] = tmp; s8[6:0] = s7[6:0];
    c8[7] = tmp; c8[6:0] = c7[6:0];
    s7[6:1] = s6[5:0]; s7[0] = s[0];
    c7[6:1] = c6[5:0]; c7[0] = c[0];
    s6[5:1] = s5[4:0]; s6[0] = s[1];
    c6[5:1] = c5[4:0]; c6[0] = c[1];
    s5[4:1] = s4[3:0]; s5[0] = s[2];
    c5[4:1] = c4[3:0]; c5[0] = c[2];
    s4[3:1] = s3[2:0]; s4[0] = s[3];
    c4[3:1] = c3[2:0]; c4[0] = c[3];
    s3[2:1] = s2[1:0]; s3[0] = s[4];
    c3[2:1] = c2[1:0]; c3[0] = c[4];
    s2[1] = s1; s2[0] = s[5];
    c2[1] = c1; c2[0] = c[5];
    s1 = s[6];
    c1 = c[6];
    // output is in 2's complement format
    out[7:0] = s8[7:0] – c8[7:0];
end
endmodule
```

mon voltage reference. This adds complexity and may not add significant information content for modeling problems in a differential circuit. Three reasons for using differential modeling will be discussed.

6.6.3 Examining Effects of Common-Mode Feedback Errors

One complication of using differential circuits is the need for a common-mode feedback circuit in the amplifiers. This circuit establishes the common-mode operating point for a true differential amplifier. Typically, there are time constants associated with the settling of the common-mode operating point. If the settling is not fast enough, it may be possible to convert some common-mode errors to differential signals. This is an indication of an incorrectly designed circuit. These effects can be modeled, but should be handled in a more detailed amplifier model that can be embedded within the gain stage model.

6.6.4 Noise Immunity Provided by Differential Circuits

Differential circuits are often adopted for their superior power supply noise rejection. Supply noise is injected symmetrically on the signal path nodes, thus the noise appears as a common-mode signal and is rejected. Noise simulations of this magnitude require accurate modeling of the op-amp and the settling process. This type of noise modeling is beyond the scope of this paper.

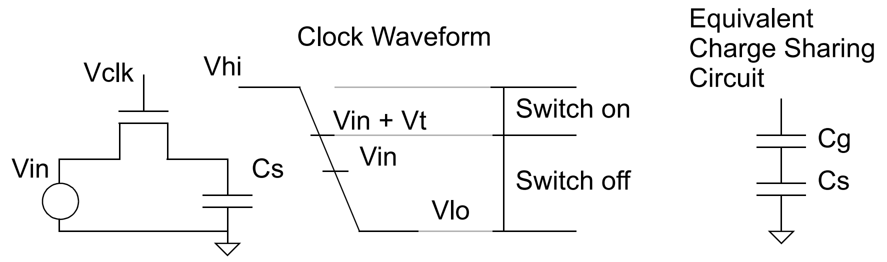
6.6.5 Charge Injection Errors Mitigated by Differential Bottom-Plate Sampling

Another reason for choosing a differential implementation is to incorporate a more accurate sampling circuit. In a single-ended implementation, a single sampling switch is used to acquire the signal on to the sampling capacitor. The switch is typically an MOS transistor. The key problem is that the channel charge in the transistor is a function of the input voltage. During the turn off of the switch, a signal dependent charge can be dumped from the channel on to the sampling capacitor [9]. This introduces a signal dependent charge error that appears as non-linearity. To avoid the charge injection from the switches, a differential bottom plate sampling scheme is often used [10].

In studying this effect, a single-ended architecture can still be used with a behavioral model for charge injection. The necessary modification is to add a small charge to the capacitor that is dependent on the input voltage as referenced to ground. A simple mathematical derivation follows.

Consider the clock edge and sampling switch circuit shown in Figure 13. Assuming the

FIGURE 13 *Illustration of charge injection with an NMOS switch.*



clock edge is slower than the time for the device to turn off, the two sides of the switch will be equalized by the transistor until it reaches the threshold voltage. At this point, the device will turn off. The maximum offset error caused by the sampling operation is shown in (18).

$$V_{os} = \frac{C_g}{C_g + C_S}(V_i + V_T - V_L) = \frac{C_g V_i}{C_g + C_S} + \frac{C_g(V_T - V_L)}{C_g + C_S} \tag{18}$$

Note that this equation contains a signal dependent portion and a fixed offset. If a differential circuit is used, the fixed portion will approximately cancel if the switches are perfectly matched. The signal dependent portion can be calculated and added in the model. Note that this model is only a rough approximation.

6.6.6 Noise Modeling

As discussed previously, the thermal sampling noise of capacitors can be modeled. If the capacitors are not sized correctly, thermal noise will limit the resolution of the converter. Even though more bits may be calculated, they will not have significant information in them. One way to see this would be to add the noise to the model and then convert a single DC value and see how many codes are obtained.

To add noise to the model, two noise contributions need to be added for the two input sampling operations. Temperature also needs to be a consideration since the noise increases with temperature.

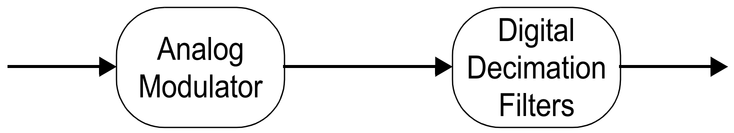
6.6.7 Power vs. Area Trade-offs

As shown in the behavioral model, the pipeline converter is easily implemented by designing a single stage and then replicating it. In this approach, the stage has to be designed for the worst case, which is the first stage. However, the requirements on matching accuracy and noise decline for each of the following stages. To save power and die area, it can prove advantageous to optimize each stage for it's own requirements [11]. This can be done in behavioral modeling by paying attention to capacitor sizing for mismatch and kT/C noise. In addition, models for integrators need to be sized to ensure adequate settling time for the signals.

7 Oversampling Noise Shaping A/D Converters

Oversampling A/D converters are attractive for IC implementation since they can be realized in standard CMOS and have been used to demonstrate greater than 16 bit resolution without the need for trimming or precision analog circuitry. The term oversampling A/D converter is usually applied to the combination of a noise shaping coder or modulator and a set of digital filters as shown in Figure 14.

FIGURE 14 *Block diagram of an oversampling and noise shaping A/D converter.*



7.1 Modulators for Oversampling A/D Conversion

Oversampling modulators are related to Delta Modulators. They were initially proposed as a means of overcoming some of the problems encountered in Delta Modulation. They are referred to as modulators since they were first used to encode telemetry information in a bit stream. Inose and Yasuda, the researchers who developed the method, named it $\Delta - \Sigma$ modulation [12]. Later researchers have also used the name $\Sigma - \Delta$ modulation which has caused some confusion. In this section, the terms modulator and noise shaping coder will be used interchangeably to describe these circuits. This section will provide an overview of oversampling, quantization noise, noise shaping, and the different types of modulators currently in use.

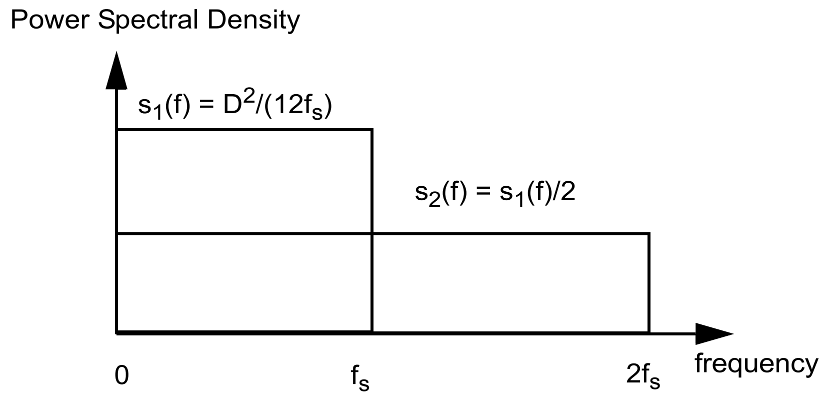
7.2 Quantization Noise and Oversampling

The process of signal quantization consists of sampling a signal and then assigning each sample a digitized representation. The quantization error is defined to be the difference between the actual analog value and the digitized representation that is assigned to the sample. Uniform quantizers use the same step size for each digital level assigned to a value. In a linear 8 bit A/D converter, for example, there are 256 levels equally spaced across the full scale voltage range.

Quantization can be modeled using an additive error signal $e(t)$ to simulate the actual quantization noise. If the quantizer has a resolution larger than 2 bits and if the input signal is active, the error signal $e(t)$ will tend to a uniform distribution on the interval defined by the quantizer step size. Consecutive samples from $e(t)$ appear to be statistically independent and the quantization noise can be modeled using white noise. Assuming that the errors are uniformly distributed and that the step size is Δ, it can be shown that the quantization error has power $\Delta^2/12$. A simple model for an A/D converter can be implemented by adding white noise of with power $\Delta^2/12$ to the original signal.

By using the additive white noise model, the effects of oversampling can be studied. If the signal is oversampled by a factor of two, the total quantization error still has power $\Delta^2/12$, but it is spread over a larger frequency region. Figure 15 illustrates this using

FIGURE 15 *Power spectral density of the quantization noise for two sampled signals.*



power spectral densities *s(f)*. An ideal low pass filter can be used to limit the signal bandwidth to *fs*, eliminating half of the noise power and increasing the signal power to noise power ratio by two. Since the signal is now band limited, it can be resampled at half the rate to provide data at the original desired rate. When using higher oversampling ratios, we find that each octave of oversampling provides a gain of 3 dB in SNR when perfect digital filtering is used. Note it is only 3 dB because SNR is expressed as a voltage ratio and not a power ratio.

For sinusoids, it can be shown that if a full scale signal is quantized using an N-bit quantizer, the SNR will be given by (19).

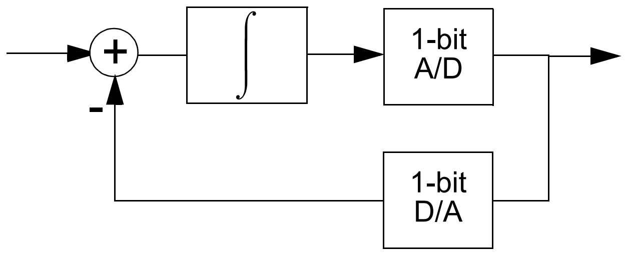
$$SNR = (6.02N + 1.76)dB \tag{19}$$

The 3 dB gain per octave of oversampling translates into 1/2 bit in effective resolution for an A/D converter. This illustrates the fundamental trade-off of resolution for speed in A/D converters. Rather than using better A/D converters which can require costly trim steps, designers can use oversampling and digital decimation filtering which adds costs in on-chip digital filters. The gains through oversampling are limited, since a two bit gain in converter performance requires a signal to be oversampled by a factor of $2^4 = 16$. A better solution is to gain more resolution per octave of oversampling by using noise shaping along with oversampling.

7.3 Noise Shaping and the *Δ–Σ* Modulator Family

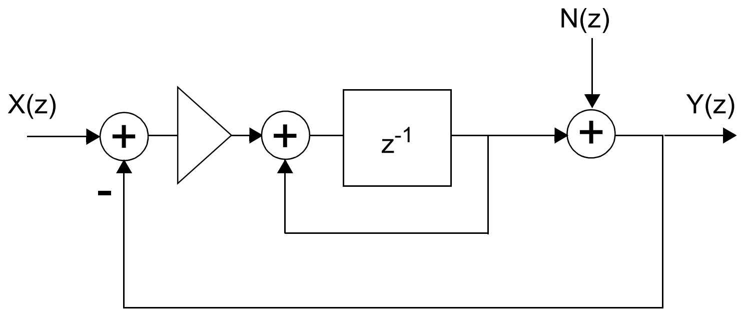
Noise shaping is achieved by pushing quantization noise away from a particular region of interest. It can be implemented using the Δ–Σ modulator shown in Figure 16. The cir-

FIGURE 16 *Block diagram of a first order Δ–Σ modulator.*



cuit consists of an integrator in a feedback loop with an A/D and a D/A converter. In the figure, 1-bit A/D and D/A converters are used, but noise shaping will occur regardless of the number of bits in the converters. To examine how noise is shaped, the A/D converter can be replaced with an additive white noise source to provide the signal flow graph shown in Figure 17.

FIGURE 17 *Linearized model of the first order delta-sigma modulator.*



The transfer functions for the linearized flowgraph of Figure 17 can be easily analyzed using feedback theory. The transfer function from input to output is given in (20)

$$\frac{Y(z)}{X(z)} = z^{-1} \tag{20}$$

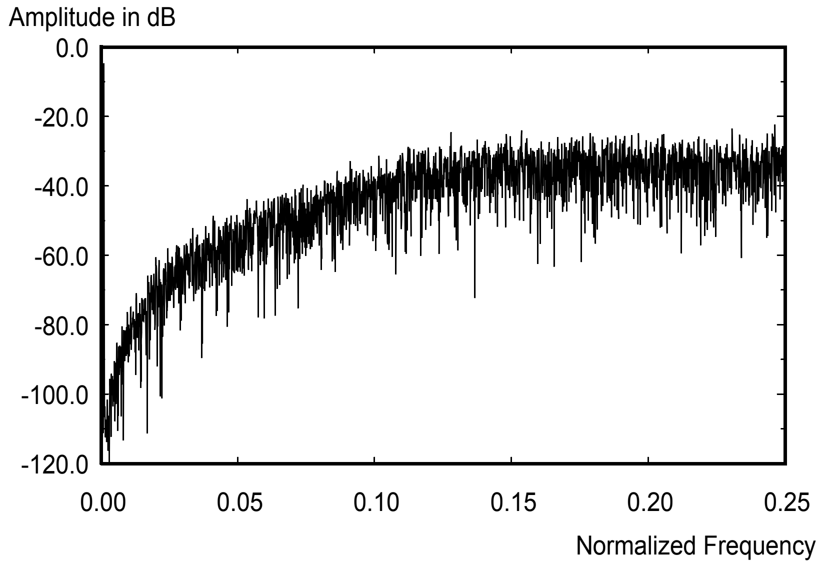
The transfer function from the noise source to the output is given in (21).

$$\frac{Y(z)}{N(z)} = 1 - z^{-1} \tag{21}$$

(20) is a delay, which causes no signal distortion while (21), the noise transfer function, is high pass. The quantization noise has been shaped away from the low frequency baseband region. It must be stressed that this analysis is based on simplified assumptions, but it gives good first-order insight.

These noise shaping results can be verified in simulation. The FFT of the output of a modulator with a sinusoidal input is shown in Figure 18. The input signal passes

FIGURE 18 *FFT output of a second-order $\Delta-\Sigma$ modulator with sinusoidal input.*



through the modulator with no distortion or attenuation but the quantization noise is shaped with a high pass response. If the high pass noise was filtered out, the original sinusoid could be recovered with high resolution since the low quantization noise region would be preserved.

The gain in SNR per octave of oversampling can be calculated by using (21). It is assumed that the 1-bit A/D converter contributes additive white noise with power $e_o^2 = \Delta^2/12$ and the sampling period is τ. The power spectral density of the noise at the output of the modulator can be derived from (21).

$$PSD = 2\tau e_o^2(2 - 2\cos\omega) \tag{22}$$

To find the total in band noise, the power spectral density must be integrated from 0 to ω_o, the highest frequency of interest. After performing the integration, the in-band noise power N can be expressed by (23).

$$N = \frac{2e_o^2}{\pi}(\omega_o - \sin\omega_o) \tag{23}$$

This can be simplified if the sine function is approximated by the first 3 terms of a Taylor's Series expansion, giving (24).

$$N \approx \frac{e_o^2 \pi^2}{3}(2 f_o \tau)^3 \text{ where } f_o = \omega_o / 2\pi \tag{24}$$

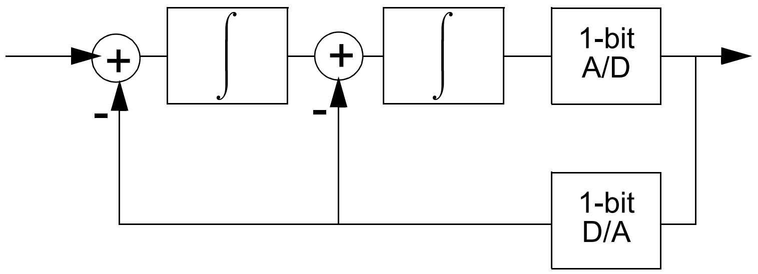
If we assume that the power of the largest possible input sinusoid is Δ^2 and define the oversampling ratio OSR to be $1/(2 f_o \tau)$, the SNR is given by (25).

$$SNR = \frac{6}{\pi}(OSR)^{\frac{3}{2}} \tag{25}$$

Thus each time OSR is doubled, the SNR increases by 9 dB, which translates to 1.5 bits in effective resolution for an A/D converter per octave of oversampling.

More noise shaping can be achieved using a second order Δ–Σ modulator which is shown in Figure 19. Under suitable assumptions, the transfer function for noise and signal can be expressed as in (26).

FIGURE 19 *Block diagram of a second order Δ–Σ modulator.*



$$Y(z) = E(z)(1 - z^{-1})^2 + X(z) z^{-1} \tag{26}$$

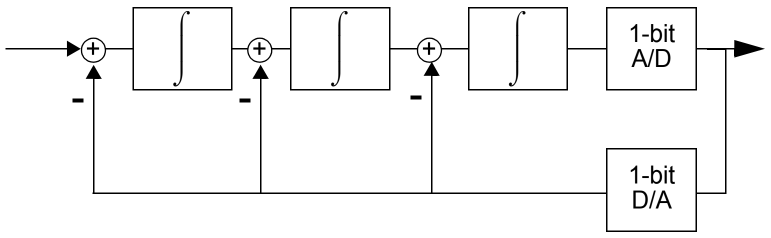
The SNR value can be calculated using the method described previously giving (27). For each octave of oversampling, a 15 dB gain in SNR is achieved.

$$SNR = \frac{\sqrt{60}}{\pi^2}(OSR)^{\frac{5}{2}} \tag{27}$$

These calculations estimate the possible resolution that could be achieved if a perfect brick wall filter were used to eliminate the out of band noise. While the equations are not accurate for design work, they do illustrate the major advantages of oversampling A/D converters. The modulators tend to be simple, requiring a few op-amps, capacitors, and comparators. Since little precision analog circuitry is required, these units can be implemented on-chip with digital signal processing systems.

The additive white noise model ignores the nonlinear behavior of the loop. First and second order Δ–Σ modulators exhibit tones that are created by limit cycles in the loop. However, first and second order Δ–Σ modulators are inherently stable. Higher order modulators based on cascades of integrators, like the one shown in Figure 20, have been studied, but stability problems make them less attractive than the first and second order modulators.

FIGURE 20 *A third order Δ−Σ modulator using triple integration.*



7.4 Developing Accurate High-Level Simulation Models

Behavioral modeling has been used from the beginning in the design of oversampling A/D converters. Circuit simulation approaches were too computationally expensive to provide meaningful results for analysis. The key problem is that oversampling forces many clock cycles to be simulated for each output sample. In 1993, a simulation required 2 weeks of CPU time to create 500 output data points. This simulation would be much faster on today's machines, but still would be in the hours range. The key problem in using a Spice approach to simulate the oversampling converter is the need to simulate all of the transient effects at high accuracy. A typical second order modulator may use an oversampling ratio of 256. To create 1024 output samples, 262,144 clock cycles need to be simulated. For 16-bit resolution, the numerical dynamic range of 96dB. Simulator tolerances are typically in the range of 80 dB when tolerances are set high. Each cycle requires calculation of the settling of the two op-amps. The simulation takes a long time, may not show the full resolution, and does not model noise effects. Behavioral modeling provides the possibility of a large speed up (100-200X), but circuit designers are wary of loss of accuracy or missing the effects of parasitics.

In order to create fast and accurate simulation models of the analog modulators dominant performance limiters must be identified and modeled. Experience has shown that behavioral modeling can be quite effective in predicting the behavior of oversampling A/D converters. For circuit design, previous studies have shown how to identify key problems and quantify the sensitivity of the design to the particular non-ideality.

A reasonable model can be built using the models for switched-capacitor integrators described earlier in this paper. This model can be applied to a variety of circuit implementations, and with care, good agreement can be obtained with actual circuit implementations. In the remainder of this section, a brief description for each of the main modulator types will be given and the key issues for modeling the modulators will be reviewed. Details on the actual design and theory of operation can be found in the literature [15].

7.4.1 Low Order Modulators

The low order modulators include the first and second order modulators. Models for these circuits were pioneered by Hauser [13] and Boser [14]. Both used discrete time models coded in C or C++ to provide analysis of the limits of performance for the circuits. They identified several key circuit issues such as op-amp gain, capacitor non-linearity, and settling time and simulated the performance. Lower order modulators are attractive because they can be easily implemented in standard digital CMOS and are inherently stable and have been used to achieve up to 16 bit resolution. They provide a good trade-off for small, easy-to-design analog blocks that can be combined with fairly small digital filters. The main drawbacks are the need for comparatively large oversampling ratios, and the presence of limit cycle tones due to pattern noise.

The basic model for a first order modulator is shown in Listing 13, once again using a

LISTING 13 *Basic Verilog-AMS model for a second order delta-sigma modulator.*

```
‘include “discipline.h”
module delsig2(dout, in, phi1, phi2);
input in, phi1, phi2;
electrical in;
output dout;
reg dout;
wire phi1, phi2;
parameter real cs1 = 1u from (0:inf);
parameter real ci1 = 1u from (0:inf);
parameter real cs2 = 1u from (0:inf);
parameter real ci2 = 1u from (0:inf);
parameter real agnd = 1.65 from (0:inf);
parameter real vrefm = 1.15 from (0:inf);
parameter real vrefp = 2.15 from (0:inf);
real state1, state2, qs1, qs2, dacval;

initial begin
    qs1 = 0.0; qs2 = 0.0; state1 = 0.0; state2 = 0.0;
    dacval = vrefm – agnd;
end

always @(negedge(phi1)) begin
    qs1 = cs1∗(V(in)–agnd);
    qs2 = cs2∗(state1);
end

always @(negedge(phi2)) begin
    state1 = (qs1 + ci1∗state1 – dacval∗cs1)/ci1;
    state2 = (qs2 + ci2∗state2 – dacval∗cs2)/ci2;
end

always @(posedge(phi2)) begin
    if (state2 > 0.0) begin
        dout = 1;
        dacval = vrefp – agnd;
    end else begin
        dout = 0;
        dacval = vrefm – agnd;
    end
end
endmodule
```

single-ended implementation. An initial block was removed from the listing to save space. Note that the DAC function is implemented in a separate block that evaluates on the rising edge of $\phi 1$. The two integrators are evaluated on the falling edge of $\phi 2$.

7.4.2 Non-Idealities

Note that the model is similar to the switched-capacitor models shown in Section 4. To add the effects of finite gain, capacitor non-linearity, or finite settling time, the model only needs to be modified according to the proper template.

7.4.3 Limit Cycles

A key problem with the low order modulators is the presence of tones in the output spectrum due to pattern noise. Pattern noise arises because the modulator circuit encodes the input signal as a pulse density modulated output. For DC input values that can be expressed as rational fractions of the input reference, it is possible for a repeating pattern to set up. In the first order modulator, the easiest pattern to identify is when the input is one half of the reference. If a positive and negative references are used, the input would need to be at zero. In this case, the limit cycle oscillation is the pattern 10101010...

Behavioral models are particularly suited towards investigating the effects of tones in a system. Many simulations can be run in a short time to isolate patterns or to see if patterns exist. Other random noise sources or additive dither can often provide enough noise to disrupt the patterns. This can be studied using the models. First-order modulators were effectively used in telephony applications because the voice signal provides enough randomness to overcome the tones.

7.4.4 Multistage Noise Shaping

A key benefit of the first and second order modulators is the stability of the modulators. One way to obtain the equivalent high order transfer function is to use the multistage noise shaping approach. In this approach low order modulators are cascaded. The key to the approach is that the noise from the first stage is requantized and recombined in a way that cancels out the noise, providing an approximation to the higher order loop noise transfer function. The approach can be effective because it maintains the fast settling time of a low order loop. However, if first order modulators are used, the limit cycle noise provides an output that is hard to predict. A more important limitation is the need for high gain op-amps. Without these, some quantization noise is not cancelled and leaks through to the later stages. The problems with limit cycle noise can be lessened by using a second order loop as the first stage. Since more noise shaping is performed in the first loop, there is less sensitivity to lower gain and fewer problems with limit cycle noise.

Behavioral models are effective in accessing the effects of finite gain and limit cycles in the performance of these modulators. Note that the output is no longer a single bit. This forces the use of different digital filters than for the first and second order modulators. Behavioral modeling can also be used to study the digital filtering trade-offs in the design of these converters.

7.4.5 High Order Loops

A higher order loop is a more direct approach to obtaining the benefits of the better noise shaping. Feed-forward and feedback paths can be used to help shape the filter for the modulator. The high order loop will only be conditionally stable. Problems can occur with larger inputs, and techniques can be applied to limit the growth of signal size or the integrators can be reset if the instability is encountered. Since the feedback is kept in a single loop, there is less sensitivity to op-amp gain. Depending on the noise shaping desired, there may be a sensitivity to component variation in the loop filter.

Higher order single loop modulators provide significant modeling challenges. These loops are typically applied in high resolution applications. Switched-capacitor circuits are not favored since large capacitors are needed to minimize kT/C noise. Continuous time circuits can be used in an effort to reduce the size and increase the bandwidth.

7.4.6 Multi-Bit Quantizers

Multi-bit quantizers are often used in the loop. The quantization noise is more randomized and the less noise shaping is needed, providing lower oversampling ratios. The key issue for behavioral modeling is capturing the quantizer. The other parts of the modulator are modeled as presented previously.

7.4.7 Combination of Techniques and Recent Trends

Several of these techniques can be combined to provide higher performance converters. Most commonly, multi-bit quantization is combined with higher-order loops to achieve more resolution at a lower oversampling ratio. Behavioral modeling is a key in the design studies.

In recent years, bandpass delta-sigma modulators have become popular for conversion of IF signals in RF applications. Rather than using a high-pass noise shaping for the quantization noise, a bandpass noise shaping is used. The same types of techniques discussed previously can be applied in the study of these converters.

7.5 Digital Decimation Filtering

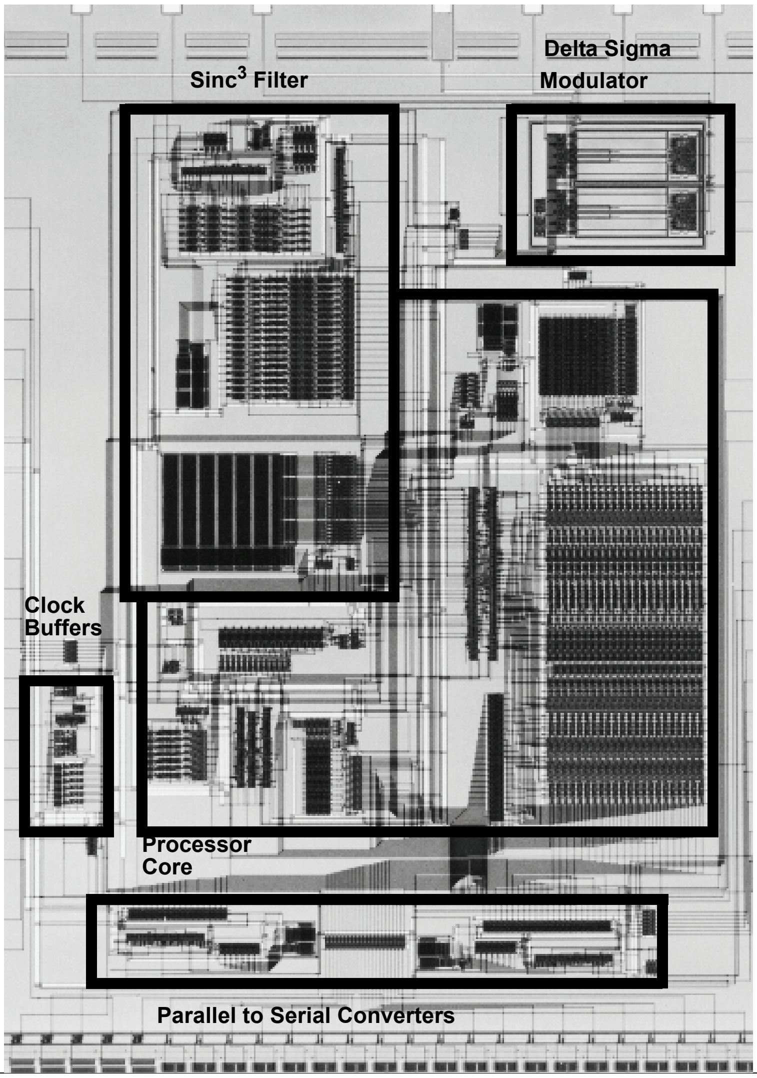
In the design of oversampling A/D converters, the effects of the digital decimation filter are often ignored. The filtering requirements of the digital filter drive the size and power dissipation. The final resolution of the conversion does not necessarily dictate the word size of the digital filters since quantization noise effects often require wider word lengths to preserve resolution. Compilation, synthesis, and digital circuit techniques can be applied to the design and modeling of these filters using standard Verilog.

7.6 A Behavioral Modeling Example

A key question is the accuracy of the behavioral models. In this section, the results from a behavioral model are compared to measured results from a fabricated device shown in Figure 16. The chip integrated a second order modulator and three digital filters.The total chip area required was 5.1 x 6.0 mm^2. The die photo for the signal acquisition module is shown in Figure 21. Performance results are tabulated in Table 1.

Figure 22 shows a comparison of measured and simulation data for the chip. There is excellent agreement between the simulation for the signal acquisition module and the actual measured data. The upper set of lines in the figure represents the resolution coming out of the first digital filter. Finite word lengths used in the final two stages of filtering steal some of the effective SNR, as shown by the lower set of curves. At most, there is less than 3 dB deviation over the entire 87 dB of dynamic range. The entire chip was simulated using a C++ based simulation system using the techniques shown in this

FIGURE 21 *Die photo of the oversampling A/D converter chip.*



paper. Models for 1/*f* and kT/C noise were included in the modulator. Finite word lengths were accounted for in the digital filter simulations. The example shows how a fairly simple behavioral model can be used to obtain good agreement with fabricated chips.

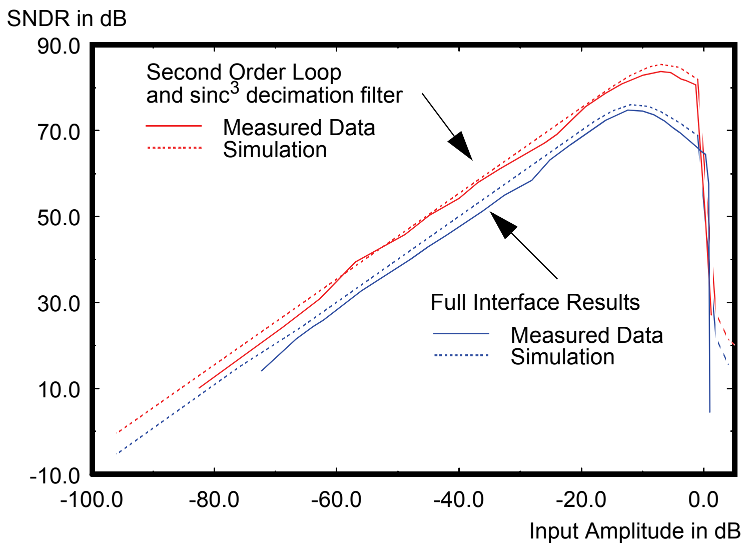
8 D/A Conversion

The majority of this paper focuses on A/D conversion. Many of the techniques already applied to A/D conversion can be used in modeling D/A converters (DACs). There are basically two key difficulties in modeling DACs for design and system models. The first problem is mismatch in the DAC elements. Much like in modeling of the Flash converter, these lead to problems in non-linearity. The second modeling task is capturing

TABLE 1 *Summary of chip performance.*

Parameter	Value
Technology	1.2mm standard CMOS
A/D Interface Area	3.2 x 4.0 mm^2 = 12.9 mm^2
Extrapolated Dynamic Range	87 dB
Measured Peak SNDR	74 dB
Output Sampling Rate	20 kHz
Oversampling Ratio	256
Anti-alias performance	> 65 dB over 0 to 9 kHz
Passband Ripple	< 0.15 dB over 0 to 9 kHz

FIGURE 22 *Comparison of simulated and measured results.*



the glitching of the DAC as it switches. In some applications, the DAC output is filtered by a reconstruction filter, so modeling of the glitch is not important. Other times, the raw DAC output drives a continuous time system and the glitch energy can have a large effect on the performance of the system. A behavioral model for DACs with glitching has been developed and presented [17]. This portion of the paper will focus on modeling the first problem of mismatch in the DAC elements.

8.1 Switched-Capacitor DACs

Trivial examples of 1-bit switched capacitor DACs were shown in the pipeline and oversampling A/D conversion sections. Switched capacitor DACs can be used in A/D conversion when successive approximation is used. Typically, the DAC is comprised of unit capacitors grouped into binary clusters to make up the array. Each bit in the input word controls one piece of the binary array. Listing 14 shows a simple model for a

LISTING 14 *Verilog AMS code for a binary-weighted DAC.*

```
module ideal_dac(in, out, clk);
input [7:0] in;
wire clock;
wreal out;

parameter real vref = 0.5 from [0:inf);
parameter real agnd = 1.65 from [0:inf);
real pow2 [7:0];
real dacout;
integer i;

initial begin
    for (i = 0; i < 8; i = i + 1) pow2[i] = vref*pow(2,i)/256.0;
end

always @(posedge(clk)) begin
    out = 0.0;
    for (i = 0; i < 8; i = i + 1) code = code + (in[i]) ? pow2[i] : 0.0;
end

assert out = dacout + agnd;
endmodule
```

binary weighted DAC. Note that the output only changes once per clock cycle, so there is no need to model the DAC with two clock phases.

A key problem with binary weighted DACs is in nonlinearity. Since unit capacitors are grouped to form the binary weighted portions, the variability between the arrays is larger than if single unit capacitors were added one at a time. The differential nonlinearity will be largest at major carries of the input code. These effects are easily modeled by adding a term in the definition of the pow2 array.

8.2 Unit Element DACs

Unit element DACs are typically made up of resistors or current sources. Much like in the Flash converter, a resistor string can be used as a DAC as shown in Figure 23. A decode tree is needed to select the correct tap to route to the output based on the input word D1D0. This is a voltage output DAC. For absolute voltages, this will be the most accurate since the two ends of the resistor string are tied to known voltages. In contrast, a current steering DAC is made up of an array of current sources connected to differential pairs as shown in Figure 24. This DAC produces a current output. Since it is a sum of currents, the full scale range can vary as the sum of the individual sources. In both cases, for a single LSB change in the input code, a single unit element is added or subtracted to change the DAC output voltage or current.

Listing 15 shows a unit element DAC model with ideal element values generated in an initial statement. Much like the flash A/D, there is mismatch in the unit elements. This was modeled much like the approach for the Flash A/D, by setting a randomness on the units. As with the Flash converter, the units could be generated in an initial statement, or they could be hand coded in an array. The variation of the units may not be totally random and can depend on the layout architecture.

FIGURE 23 *Block diagram of a two-bit resistor DAC.*

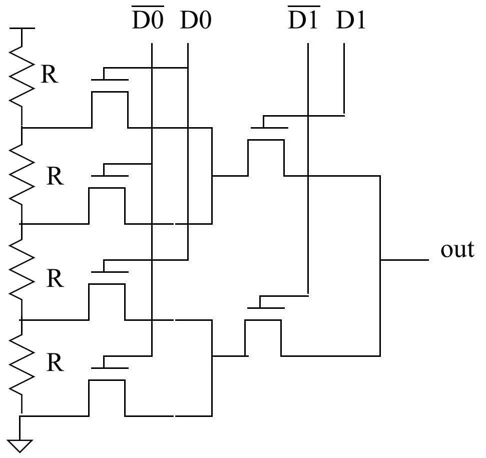
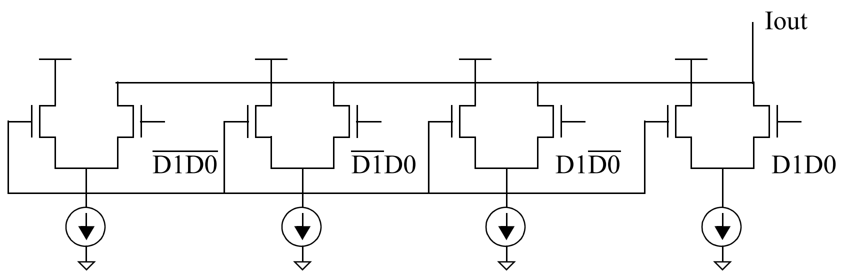


FIGURE 24 *Block diagram of a two-bit current steering DAC.*



In addressing the dynamic behavior of the DACs, the digital decoding logic affect the output. There may be different propagation delays for each of the unit cells causing different kinds of glitching patterns depending on the input data pattern.

8.3 Oversampling D/A Conversion

Oversampling and noise shaping can be used for D/A conversion. Models quite similar to the analog modulator models can be used, but they must be extended to account for finite word lengths. The digital modulator uses digital integrators with registers used in place of the accumulating capacitors.

Similar problems exist in modeling the modulator. For a 1-bit output, the shape of the output pulse stream must be modeled to capture offset and edge rate imbalances if a 1-bit D/A is used. If a multi-bit DAC is used, all the problems with nonlinearity previously mentioned need to be addressed.

The modulator creates an analog output with just a few levels. This signal needs to be filtered with some type of filter, switched-capacitor or continuous time. Accurate modeling of the filter is needed, as the filter can limit both resolution and linearity.

LISTING 15 *Verilog AMS code for a unit element DAC.*

```
module unit_dac(in, out, clk);
input [7:0] in;
wire clock;
wreal out;

parameter real ref = 0.5 from [0:inf);
real ovals[255];
real dacout;
integer i, code;
integer pow2 [7:0];

initial begin
    ovals[0] = ref/256.0;
    for (i = 1; i < 256; i = i + 1) ovals[i] =  ovals[i] + ref/256.0;
    for (i = 0; i < 8; i = i + 1) pow2[i] = pow(2, i);
end

always @(posedge(clk)) begin
    code = 0;
    for (i = 0; i < 8; i = i + 1) code = code + (in[i]) ? pow2[i] : 0.0;
    dacout = ovals[code];
end

assert out = dacout;
endmodule
```

9 Summary

This paper presented an overview of behavioral modeling techniques for A/D and D/A converters that avoided the use of the analog procedural block. These models execute very quickly and have the capability of accurately modeling the major non-ideal effects, including noise, finite op-amp gain, settling time, and non-linearities. An example of an oversampling A/D was presented to show the matching possible between fabricated devices and the behavioral models.

9.1 If You Have Questions

If you have questions about what you have just read, feel free to post them on the *Forum* section of *The Designer's Guide Community* website. Do so by going to *www.designers-guide.org/Forum*.

References

[1] M. Mar and B. Sullam, “Modeling and Verification of a Programmable Mixed-Signal Device Using Verilog”, Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), vol. 3, pp. III-903 - III-905, 2003.

[2] C. W. Mangelsdorf, “A 400-MHz Input Flash Converter with Error Correction”, IEEE Journal of Solid-State Circuits, Vol. 25, No. 1, pp. 184-191, February 1990.

[3] K. Ono, T. Matsuura, E. Imaizumi, H. Okazawa, and R. Shimokawa, “Error Suppressing Encode Logic of FCDL in a 6-b Flash A/D Converter”, IEEE Journal of Solid-State Circuits, Vol. 32, No. 9, pp. 1460-1464, September 1997.

[4] R. Gregorian and G. C. Temes, Analog MOS Integrated Circuits for Signal Processing, John Wiley and Sons, New York, 1986.

[5] K. C.-H. Chao, S. Nadeem, W. L. Lee, and C. G. Sodini, “A Higher Order Topology for Interpolative Modulators for Oversampling A/D Converters”, IEEE Transactions on Circuits and Systems, Vol. 37, No. 3, pp. 309-318, March 1990.

[6] B. Pellegrini, R. Saletti, B. Neri, and P. Terreni, “$1/f^{\nu}$ Noise Generators”, In *Noise in Physical Systems and 1/f Noise*, 1985.

[7] S. H. Lewis, H. S. Fetterman, G. F. Gross, R. Ramachandran, and T. R. Viswanathan, “A 10-b 20-Msample/s Analog-to-Digital Converter”, IEEE Journal of Solid-State Circuits, Vol. 27, No. 3, pp. 351-358, March 1992.

[8] S. H. Lewis and P. R. Gray, “A Pipelined 5-Msample/s 9-bit Analog-to-Digital Converter”, IEEE Journal of Solid-State Circuits, Vol. SC-22, No. 6, pp. 954-961, December 1987.

[9] J.-H. Shieh, M. Patil, and B. J. Sheu, “Measurement and Analysis of Charge Injection in MOS Analog Switches”, IEEE Journal of Solid-State Circuits, Vol. SC-22, No. 2, pp. 277-281, April 1987.

[10] Y.-M. Lin, B. Kim, and P. R. Gray, “A 13-b 2.5-MHz Self-Calibrated Pipelined A/D Converter in 3-μm CMOS”, IEEE Journal of Solid-State Circuits, Vol. 25, No. 4, pp. 628-636, April 1991.

[11] T. Cho and P. R. Gray, “A 10 b, 20 Msamples/s, 35mW Pipeline A/D Converter”, IEEE Journal of Solid-State Circuits, Vol. 30, No. 3, pp. 166-172, March 1995.

[12] H. Inose and Y. Yasuda, “A Unity Bit Coding Method by Negative Feedback”, Proceedings of the IEEE, Vol. 51, pp. 1524-1535, September, 1962.

[13] M. W. Hauser and R. W. Brodersen, “Circuit and Technology Considerations for MOS Delta-Sigma A/D Converters”, Proceedings of the International Symposium on Circuits and Systems, pp. 1310-1315, May 1986.

[14] B. E. Boser and B. A. Wooley, “The Design of Sigma-Delta Modulation Analog-to-Digital Converters”, IEEE Journal of Solid-State Circuits, Vol. 23, No. 6, pp. 1298-1308, December 1988.

[15] J. C. Candy and G. C. Temes, editors, “Oversampling Delta-Sigma Data Converters: Theory, Design and Simulation”, IEEE Press, New York, 1992.

[16] M. F. Mar and R. W. Brodersen, “A Design System for On-Chip Oversampling A/D Interfaces”, IEEE Transactions on VLSI Systems, Vol. 3, No. 3, pp. 345-354, September 1995.

[17] J. Vandenbussche, G. Van der Plas, G. Gielen, And W. Sansen, “Behavioral Models of Reusable D/A Converters”, IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing, Vol. 46, No. 10, pp. 1323-1326, October 1999.