The Designer's Guide Community downloaded from www.designers-guide.org

Simulating Switched-Capacitor Filters with SpectreRF

Ken Kundert
Designer's Guide Consulting, Inc.

Version 6c, 28 July 2006

SpectreRF provides powerful and unique new analyses that offer designers of switched-capacitor filters the ability to predict the performance of their circuits in ways that were not previously possible. It allows designers to quickly and directly predict the transfer and noise characteristics of filters described at the transistor level while including all of the important second-order effects. Previously, designers had to choose between using a traditional circuit simulator such as SPICE and a discrete-time simulator such as Switcap. SPICE simulates from a transistor-level description and so includes all of the desired second-order effects, but is not capable of directly computing the transfer and noise characteristics of circuit. While it may be possible to use Spice to indirectly predict such things, doing so is difficult, slow, and error prone. Switcap does directly compute the transfer and noise characteristics of switched-capacitor filters, but works from a high-level behavioral description. It cannot accept a transistor-level description and cannot include many of the second-order effects that critically important to designers.

This document works through several examples to show how to use SpectreRF to predict most of the quantities of interest with switched-capacitor filters.

Last updated on July 28, 2006. You can find the most recent version at www.designers-guide.org. Contact the author via e-mail at ken@designers-guide.com.

Permission to make copies, either paper or electronic, of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage and that the copies are complete and unmodified. To distribute otherwise, to publish, to post on servers, or to distribute to lists, requires prior written permission.

1 Introduction

When it comes to simulating switched-capacitor filters, designers have had to make a choice. They could simulate at the transistor-level with SPICE, but then they must give up on using AC and noise analyses. SPICE is only capable of performing these small-signal analyses about a DC operating point, but switched-capacitor filters need an active clock signal to operate. Without these analyses it is very expensive or impossible to predict the transfer and noise characteristics of their filters. Alternatively, they could use specially designed discrete-time simulators, such as Switcap [7], but these simulators accept only behavioral-level descriptions of the circuit. With these simulators, designers lose the ability to predict second-order effects such as finite bandwidth effects, nonlinear switch resistance effects, charge redistribution effects, slew-rate limiting effects, and back-gate bias effects.

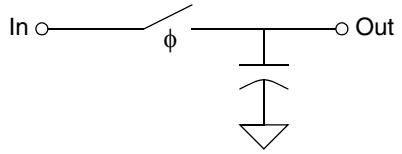
With Spectre®RF† designers now have another choice. SpectreRF is capable of simulating switched-capacitor filters at the transistor level. It is capable of performing small-signal analyses such as AC and noise about a periodic operating point and so can directly predict the transfer and noise characteristics of filters with the clock active. SpectreRF naturally includes the second-order effects mentioned above, and also allows designers to include parasitics back annotated from the layout in the simulation. Finally, because it simulates at the circuit level, SpectreRF can accurately simulate switched-current filters. These circuits cannot be adequately simulated using traditional switched-capacitor filter simulators because the imperfections that are second order effects in switched-capacitor circuits play a much more important role in switched-current filters. Most of these effects can only be included when using transistor-level simulators.

This document starts off by showing how SpectreRF can be applied to predict the performance of a simple track-and-hold, which is nothing more than a periodically clocked switch and a capacitor. The behavior of this circuit has been widely studied, and the achieved results are compared with expected results to confirm their validity. SpectreRF is then applied to a representative switched-capacitor filter as a way of demonstrating how one simulates such circuits.

2 A Simple Track and Hold

A simple track and hold is shown in Figure 1. This circuit will be used to illustrate concepts that are important for all switched-capacitor circuits. It is also used to help demonstrate how one applies SpectreRF to determine common performance metrics for switched-capacitor circuits.

FIGURE 1 *A simple track and hold.*



†. Spectre is a registered trademark of Cadence Design Systems.

The Spectre netlist of the circuit is shown in Listing 1. In normal operation the switch is periodically opened and closed at a fixed frequency with a particular phase, denoted ϕ. The signal used to drive the switch is referred to as the clock.

LISTING 1 *The Spectre netlist for Figure 1, a simple switched-capacitor track-and-hold. The include files needed to run this netlist can be found at www.designers-guide.org/Analysis. To run these examples, you will need to use the –E command line option when running Spectre.*

```
// Simple switched-capacitor track-and-hold
simulator lang=spectre
#include "cmos.mod"
ahdl_include "sh.va"

parameters VDD=5.0_V TCLK=2.5us

// power supplies
Vdd     (vdd 0)          vsource  dc=VDD
Vss     (vss 0)          vsource  dc=–VDD

// Clock
Vphi    (phi 0)          vsource  type=pulse period=TCLK val0=–VDD val1=VDD \
                                  rise=10ns delay=0.25us width=1us \
                                  fundname="clock"
Inv1    (phib 0 phi 0)   vcvs     gain=–1

// Input source
Vin     (in 0)           vsource  type=dc dc=0 pacmag=1 \
                                  freq=10kHz ampl=1 fundname="input" \
                                  freq2=10.1kHz ampl2=0 fundname2="input2"

// CMOS Switch
Mn      (in phi out vss)  nmos    w=30.4u l=7.6u
Mp      (in phib out vdd) pmos    w=30.4u l=7.6u

// Hold capacitor
Chold   (out 0)          capacitor c=10pF

// Ideal sample and hold
SH      (sout 0 out 0)   sh       period=TCLK

// Analyses
clockAlone               pss      period=TCLK maxacfreq=50MHz
directGain               pac      start=0_Hz stop=10/TCLK maxsideband=0
convGain (sout 0)        pxf      start=0_Hz stop=0.5/TCLK maxsideband=25
unsmpldNoise (out 0)     pnoise   start=0_Hz stop=10/TCLK maxsideband=100
smpldNoise (out 0)       pnoise   start=0_Hz stop=0.5/TCLK maxsideband=100 \
                                  noisetype=timedomain noisetimepoints=[0] \
                                  numberofpoints=1
enableTone1              alter    dev=Vin param=type value=sine
harmDisto                qpss     funds=["clock" "input"] maxharms=[0 3]
enableTone2              alter    dev=Vin param=ampl2 value=1
intermodDisto            qpss     funds=["clock" "input" "input2"] maxharms=[0 3 3]
```

This netlist uses a feature that is unique to Spectre. It specifies the parameters for several different waveform shapes on the input source *Vin*, and then uses alter statements during the stream of analyses to change the shape of the waveform produced. In particular, it specifies parameters for a constant-valued waveshape (*dc = 0*), a single sinusoidal tone (*freq = 10kHz ampl = 1 fundname = "input"*), and for a second sinusoidal tone

(*freq2 = 10.1kHz ampl2 = 0 fundname2 = "input2"*). Initially, the waveshape is set to a fixed value by *type = dc*. Later, the *alter* statement named *enableTone1* changes the waveshape type to *sine* to enable the first tone. Finally, *enableTone2* turns on the second tone by setting its amplitude to 1. The circuit in Listing 1 uses an idealized Verilog-A sample-and-hold model, which is given in Listing 2.

2.1 Periodic Operating Point

Generally the first step in characterizing a switch-capacitor filter with SpectreRF is to compute the periodic operating point. This is the steady-state solution of the circuit with only the clock applied (the input signal is turned off). This analysis is primarily needed to compute the periodic operating point that will be used by subsequent small-signal analyses. The periodic operating point is computed by adding a PSS or periodic steady state analysis. First make sure the input signal is disabled, done in Listing 1 by specifying *type=dc* for *Vin*. Then, in this case, a PSS analysis named *clockAlone* calculates the periodic operating point. It is specified with two numerical parameters, the clock period and a parameter *maxacfreq* that specifies the maximum frequency that will be used in any subsequent small-signal analyses. This helps the PSS analysis choose a timestep that assures that the small-signal analyses will be accurate. Its value is affected by both the maximum small-signal analysis frequency (generally the stop frequency) and the maximum sideband of interest, such that $maxacfreq \geq f_{stop} + f_c \times maxsideband$ where f_c is the clock frequency. In the example, the noise analyses are using *maxsideband* = 100, f_{stop} = *stop* = 10/TCLK = 4 MHz, and f_c = 1/TCLK = 400 KHz and so *maxacfreq* must be set to at least 44 MHz. This represents a lower bound for the value of *maxacfreq*. This value generally gives accurate results. Higher values can increase accuracy, but will also result in slower simulations.

2.2 Noise

The noise at the output of a track an hold, shown as v_c in Figure 2, can conceptually be separated into two orthogonal components [1],

$$v_c = v_t + v_h. \tag{1}$$

The first component, v_t, represents the noise in the track mode. It consists of band-limited white noise during the track phase and is zero during the hold phase. The noise is produced by the thermal noise in the switch resistance and is filtered by the time constant caused by the capacitor loading the switch. Assume for the moment that the switch is always closed (that there is no hold phase), the single-sided noise density would be

$$S_{RC}(f) = \frac{4kTR_{on}}{1 + (2\pi f R_{on} C)^2} \tag{2}$$

and the total noise would be

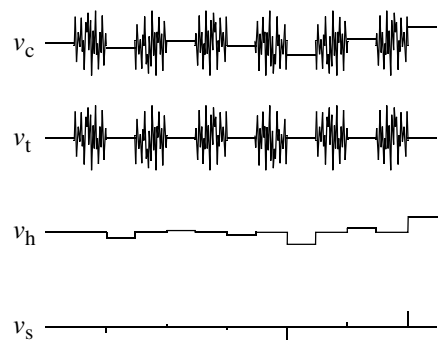
$$\overline{v_{RC}^2} = 4kTR_{on}\int_0^\infty \frac{1}{1 + (2\pi f R_{on} C)^2} df = \frac{4kTR_{on}}{2\pi R_{on} C}\tan^{-1}(2\pi f R_{on} C)\Big|_0^\infty = \frac{kT}{C}. \tag{3}$$

Note that the noise density within the bandwidth of the RC filter is proportional to R_{on}, but that the bandwidth is inversely proportional to R_{on}, so that the total noise is independent of R_{on}.

LISTING 2 *An idealized sample-and-hold model written in Verilog-A that works with SpectreRF.*

```
// Periodic Sample & Hold
// Works with SpectreRF (has no hidden state)
// Almost ideal ...
//      Has buffered input and output (infinite input Z, zero out Z)
//      Exhibits no offset or distortion errors
//      Only nonideality is finite aperture time and very small amount of droop

`include "discipline.h"
`include "constants.h"

module sh (Pout, Nout, Pin, Nin);
input Pin, Nin;
output Pout, Nout;
electrical Pin, Nin, Pout, Nout;
parameter real period=1 from (0:inf);
parameter real tdelay=0 from [0:inf);
parameter real aperture=period/100 from (0:period/2);
parameter real tc=aperture/10 from (0:aperture);

integer n;
real tstart, tstop;
electrical hold;

    analog begin

// Determine the point where the aperture begins
        n = ($abstime - tdelay + aperture) / period + 0.5;
        tstart = n*period + tdelay - aperture;
        @(timer(tstart))
            ;

// Determine the time where the aperture ends
        n = ($abstime - tdelay) / period + 0.0;
        tstop = n*period + tdelay;
        @(timer(tstop))
            ;

// Implement switch with effective series resistence of 1 Ohm
        if (($abstime > tstop - aperture) && ($abstime <= tstop))
            I(hold) <+ V(hold) - V(Pin, Nin);
        else
            I(hold) <+ 1.0e-12 * V(hold);

// Implement capacitor with an effective capacitance of tc
        I(hold) <+ tc * ddt(V(hold));

// Buffer output
        V(Pout, Nout) <+ V(hold);

// Control time step tightly during aperture and loosely otherwise
        if (($abstime >= tstop - aperture) && ($abstime < tstop)) begin
            $bound_step(tc);
        end else begin
            $bound_step(period/5);
        end
    end
endmodule
```

FIGURE 2 *The output of a track-and-hold with noisy input signal (v_c) is conceptually decomposed into a track signal (v_t) and a hold signal (v_h), where $v_c = v_t + v_h$. The signal v_s is a sampled version of v_h.*



Now allow the switch to operate so that it is opens and closes with a period of T_c with the switch being closed for mT_c and open for $(1-m)T_c$. The result will be a cyclostationary noise process whose time-average power spectral density and total power are simply (2) and (3) scaled by the duty cycle m,

$$S_t(f) = \frac{4mkTR_{on}}{1+(2\pi R_{on}Cf)^2}, \tag{4}$$

$$\overline{v_t^2} = \frac{mkT}{C}. \tag{5}$$

The second component, v_h, represents the noise in the hold mode. It is zero during the track phase and has a constant value during the hold phase. Conceptually, v_h is constructed by sampling v_c at the very end of the track phase, and then holding that value for the duration of the hold phase. Consider explicitly separating these two steps. First sample v_c at the very end of the track phase to create v_s. The sampling process results in aliasing, with the noise density of (2) being replicated at intervals of f_c and summed as in Figure 3 to give a noise power density† for v_s of

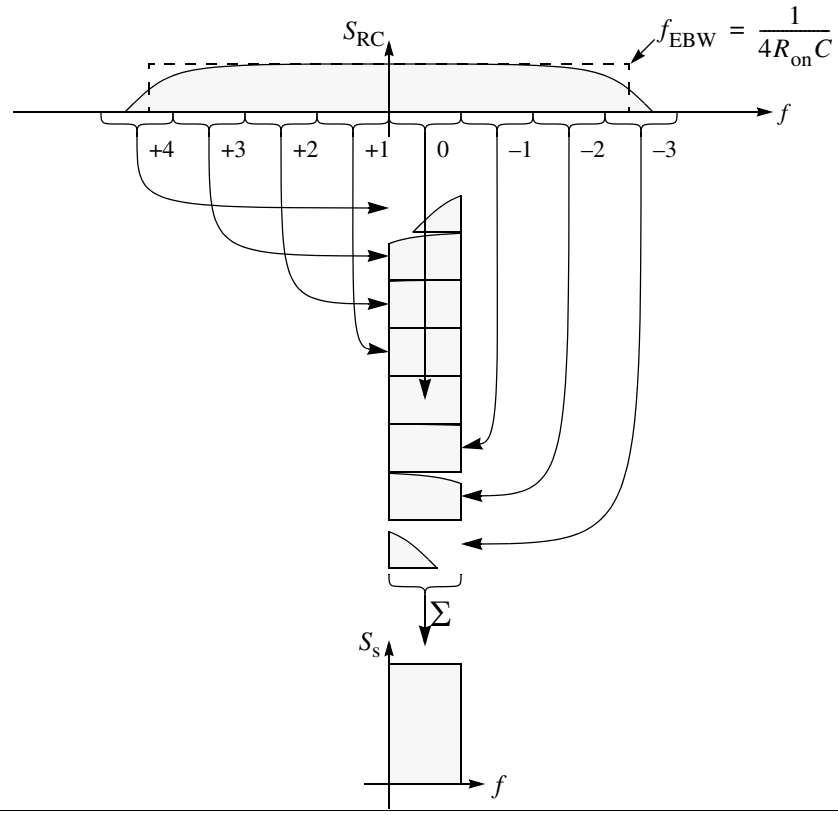
$$\hat{S}_s(f) = \sum_{k=-\infty}^{\infty} \hat{S}_{RC}(f-kf_c). \tag{6}$$

At this point a double-sided representation, $\hat{S}_s$ and $\hat{S}_{RC}$, is used for S_s and S_{RC} to make it easier to properly account for aliasing. v_s is a sequence with rate f_c, and so S_s will be periodic with period f_c. Being a real sequence, it is also conjugate symmetric about 0. When using the single-sided transform, the frequency range of interest becomes 0 to $f_c/2$; everything outside this range is either a repeat, or a repeat of a mirror image. Contributions to this range from large values of k in (6) are negligible because S_{RC} has finite

†. If you were expecting to see a factor of f_c^2 multiplying the right-hand side of (6), you are assuming the use of the Fourier transform, which is appropriate for finite-energy signals. These signals have finite power, and so we base our conversion to the frequency domain on the discrete-time Fourier series.

See Section 2.4 of [1] for a derivation of (6).

FIGURE 3 *An illustration of the accumulation of noise that results in v_s due to aliasing.*



bandwidth. One can make an estimation of the total noise power density in S_s by approximating S_{RC} with a rectangular noise density that has the same total power and the same density at low frequencies. As can be seen from (2) the single-sided power spectral density at low frequencies is $4kTR_{on}$, and from (3) the total power in S_{RC} is kT/C, and so the effective bandwidth of the rectangle, f_{EBW}, is

$$f_{EBW} = \frac{kT}{C}\frac{1}{4kTR_{on}} = \frac{1}{4R_{on}C}. \tag{7}$$

Now the summation in (6) can be approximated by splitting the effective noise bandwidth from $-f_{EBW}$ to f_{EBW} into N rectangles, each with a width of f_c and height $2kTR_{on}$,

$$N = 2\frac{f_{EBW}}{f_c} = \frac{1}{2R_{on}Cf_c}. \tag{8}$$

The noise in S_{RC} is a stationary process and so is uncorrelated over f [5] allowing the N rectangles to be combined by simply summing their noise powers, giving

$$\hat{S}_s(f) \cong \sum_{k=-N/2}^{N/2-1} 2kTR_{on} = (2kTR_{on})N = 2kTR_{on}\frac{1}{2R_{on}Cf_c} = \frac{kT}{Cf_c}. \tag{9}$$

Rewriting using a single-sided representation gives

$$S_s(f) = 2\frac{kT}{Cf_c}. \tag{10}$$

Integrating over the range of 0 to $f_c/2$ gives the total noise power in the sampled waveform v_s,

$$\overline{v_s^2} = \frac{kT}{C}. \tag{11}$$

This shows that the sampling concentrates the full noise power of the switch resistance into the baseband of the sampling process. Notice that once again, the total noise power is independent of the switch resistance, and inversely proportional to the capacitance.

To compute the noise density in v_h, it is only necessary to process S_s in (10) to account for the effect of the zero-order hold. To implement a zero-order hold involves convolving the sequence v_s with a pulse of unit height with a width of $(1–m)T_c$. Convolution in the time domain becomes multiplication in the frequency domain. The Fourier series of the pulse train† is

$$\Pi\left(\frac{t}{(1-m)T_c}, T_c\right) \Leftrightarrow \frac{(1-m)T_c}{T_c}\frac{\sin(\pi f(1-m)T_c)}{\pi f(1-m)T_c} = (1-m)\mathrm{sinc}(f(1-m)T_c). \tag{12}$$

where $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$. Thus,

$$S_h(f) = \left((1-m)\frac{\sin(\pi f(1-m)T_c)}{\pi f(1-m)T_c}\right)^2 S_s(f), \tag{13}$$

$$S_h(f) = [(1-m)\mathrm{sinc}(f(1-m)T_c)]^2 2\frac{kT}{Cf_c}. \tag{14}$$

The function $\mathrm{sinc}(x) \to 1$ as $x \to 0$, and so for low frequencies (when $f \ll f_c$),

$$S_h(f) = 2(1-m)^2\frac{kT}{Cf_c}. \tag{15}$$

Finally, (4) and (14) can be combined to find the noise density of v_c, the composite that includes the noise from both the track and the hold phases,

$$S_c(f) = S_t(f) + S_h(f) = \frac{4mkTR_{on}}{1+(2\pi f R_{on}C)^2} + [(1-m)\mathrm{sinc}(f(1-m)T_c)]^2 2\frac{kT}{Cf_c}. \tag{16}$$

Integrating from $f = 0$ to ∞ gives a total composite noise of

$$\overline{v_c^2} = \frac{mkT}{C} + \frac{(1-m)^2}{2(1-m)T_c}2\frac{kT}{Cf_c} = m\frac{kT}{C} + (1-m)\frac{kT}{C} = \frac{kT}{C}. \tag{17}$$

†. If you were not expecting to see the factor of T_c in the denominator of the transform shown in (12), you are assuming the use of the Fourier transform, which is appropriate for finite-energy signals. These signals have finite power, and so we base our conversion to the frequency domain on the discrete-time Fourier series.

2.2.1 The *maxsideband* Parameter

The formulas given above can be used to estimate the noise performance of a switched capacitor track and hold, as can SpectreRF. However, it is important to understand that SpectreRF does not use these formulas or anything like them. Such would be the approach used by traditional switched-capacitor filter simulators [7]. SpectreRF on the other hand, is a circuit simulator like SPICE. And like SPICE, it uses first principles to predict noise. Innately, SpectreRF knows nothing particular about switched-capacitor circuits. Instead, it combines the SPICE-level device models using KCL and solves the resulting systems of equations. As such, it is not limited to switched-capacitor circuits, or in fact to any particular type of circuit by the assumptions it makes. SpectreRF does make some assumptions, but they tend to be similar to those made by SPICE: that the accuracy settings are tight enough so that the an accurate result is computed and that the noise is small enough so that it by itself does not generate a nonlinear response. There are two more assumptions that it makes by virtue of it performing a noise analysis of a periodic system: that the circuit must have a periodic solution and that enough of the mixing terms are accounted for to provide accurate prediction of noise folding. This section addresses this last assumption.

As described above, aliasing of the noise, or noise folding, plays an important role in this circuit as it does in all switched-capacitor filters. The *maxsideband* parameter is used by SpectreRF's PNoise analysis to determine how much of this aliasing to consider. This is an important parameter because it sets an important accuracy versus speed trade-off. Providing too large a value will result in simulations that run unnecessarily long, whereas too small a value cause the noise to be under-estimated. This section presents guidelines that help you choose a good value for *maxsideband*.

Conceptually, the *maxsideband* parameter tells the PNoise analysis how many noise folds to account for when computing the noise at the output of the circuit. More precisely, if *maxsideband* = N_{msb}, then the noise aliasing up or down as a result of it mixing with the up to the first N_{msb} harmonics of the clock should be included in the output noise. In general, the most common strategy for determining the best value to use for *maxsideband* involves starting with a small value and repeatedly increasing it and performing a noise analysis. The lowest value that gives a consistent value for the noise is the one used. This often results in a relatively conservative choice for *maxsideband*, and as a result, relatively slow simulations. Once this value is known, consider determining how much accuracy is needed, and then backing off to a value for *maxsideband* that just achieves this level of accuracy. This almost always reduces the amount of output noise reported by PNoise. Given the PNoise results with this smaller value for *maxsideband*, it is in general fairly easy to estimate with reasonable confidence what the actual noise will be. Simply changing the accuracy requirement from 0.5% to 5% can result in a 10 reduction in simulation time.

For this particular circuit, it is possible to explicitly calculate the error that results from using a finite *maxsideband*. To do so, remember that the switch is essentially sampling the band-limited white noise process of (2), and that the PNoise analysis will include noise folding terms up a frequency of $N_{msb}f_c$ where N_{msb} is the value given for *maxsideband*†. Determining the error in the total noise is then simply a matter of evaluating

†. Actually, $f_{max} = N_{msb}f_c + f$, but f is usually less than f_c and N_{msb} is usually large, so f is ignored.

the integral in (3) up to this frequency rather than to infinity. The computed noise power is then

$$\overline{\tilde{v}_{\mathrm{RC}}^2} = 4kTR_{\mathrm{on}}\int_0^{f_{\max}} \frac{1}{1+(2\pi R_{\mathrm{on}}Cf)^2}df = \frac{kT}{C}\left(\frac{2}{\pi}\tan^{-1}(2\pi R_{\mathrm{on}}Cf_{\max})\right), \tag{18}$$

where $\overline{\tilde{v}_{\mathrm{RC}}^2}$ is the approximation to the total noise $\overline{v_{\mathrm{RC}}^2}$ that results by ignoring noise at frequencies greater than $f_{\max}$. Define F_{norm} to be the highest frequency considered in the noise folding calculations ($f_{\max} = N_{\mathrm{msb}}f_{\mathrm{c}}$) normalized to the bandwidth of the RC network ($f_{\mathrm{BW}} = 1/(2\pi R_{\mathrm{on}}\mathrm{C})$),

$$F_{\mathrm{norm}} = \frac{f_{\max}}{f_{\mathrm{BW}}} = \frac{f_{\max}}{\left(\frac{1}{2\pi R_{\mathrm{on}}C}\right)} = 2\pi R_{\mathrm{on}}Cf_{\max} = 2\pi R_{\mathrm{on}}CN_{\mathrm{msb}}f_{\mathrm{c}}. \tag{19}$$

where $f_{\max} = N_{\mathrm{msb}}f_{\mathrm{c}}$. From (20) and (19), the error in terms of total noise voltage becomes

$$\varepsilon = \sqrt{\overline{\tilde{v}_{\mathrm{RC}}^2}} - \sqrt{\overline{v_{\mathrm{RC}}^2}} = \sqrt{\frac{kT}{C}\frac{2}{\pi}\tan^{-1}(F_{\mathrm{norm}})} - \sqrt{\frac{kT}{C}} = \sqrt{\frac{kT}{C}}\left(\sqrt{\frac{2}{\pi}\tan^{-1}(F_{\mathrm{norm}})} - 1\right), \tag{20}$$

or in terms of percent,

$$E = \left(\frac{\sqrt{\overline{\tilde{v}_{\mathrm{RC}}^2}} - \sqrt{\overline{v_{\mathrm{RC}}^2}}}{\sqrt{\overline{v_{\mathrm{RC}}^2}}}\right) \times 100\% = \left(\sqrt{\frac{2}{\pi}\tan^{-1}(F_{\mathrm{norm}})} - 1\right) \times 100\%. \tag{21}$$

Switched capacitor circuits are usually designed so that the RC time constant is 7-10 times smaller than the duration of the clock phase for which the switch is closed. From this, the number of sidebands needed to achieve a particular error can be estimated for a typical switched-capacitor circuit. Rearranging (21) to find F_{norm} as a function of E,

$$F_{\mathrm{norm}} = \tan\left(\frac{\pi}{2}\left(\frac{E}{100}+1\right)^2\right). \tag{22}$$

And from (19),

$$F_{\mathrm{norm}} = 2\pi R_{\mathrm{on}}CN_{\mathrm{msb}}f_{\mathrm{c}} = 2\pi\tau N_{\mathrm{msb}}f_{\mathrm{c}}, \tag{23}$$

where $\tau = R_{\mathrm{on}}C$. Define α such that $T_{\mathrm{p}} = \alpha\tau$, where T_{p} is the duration of the on phase of the switch and α is the number of time constants that pass during this phase. Typically, $7 < \alpha < 10$ in order to allow the capacitors to fully charge on each phase of the clock. Assume that the clock cycle is divided into N_{p} equal intervals and that $T_{\mathrm{c}} = N_{\mathrm{p}}T_{\mathrm{p}}$. Then

$$\tau = \frac{T_{\mathrm{c}}}{\alpha N_{\mathrm{p}}}. \tag{24}$$

Combining (22), (23), and (24) gives

$$2\pi\frac{T_{\mathrm{c}}}{\alpha N_{\mathrm{p}}}N_{\mathrm{msb}}f_{\mathrm{c}} = \tan\left(\frac{\pi}{2}\left(\frac{E}{100}+1\right)^2\right), \tag{25}$$

$$N_{msb} = \frac{\alpha N_p}{2\pi}\tan\left(\frac{\pi}{2}\left(\frac{\varepsilon}{100}+1\right)^2\right). \quad (26)$$

This function is tabulated in Table 1 for typical values of ε, α, and N_p. This table shows that cases while trying to achieve high accuracy is very expensive, when the error requirements are modest (≥ 5%) the simulations can be quite efficient for typical switched-capacitor filters.

TABLE 1 *The value of maxsideband (N_{msb}) needed to predict the total noise to within a given error E for typical values of α and N_p.*

E	N_{msb}	N_{msb} with $\alpha = 7$, $N_p = 2$	N_{msb} with $\alpha = 10$, $N_p = 2$
–10%	$\alpha N_p/2$	7	10
–5%	αN_p	14	20
–2%	$2.5\alpha N_p$	35	50
–1%	$5\alpha N_p$	70	100
–0.5%	$10\alpha N_p$	140	200
–0.2%	$25\alpha N_p$	350	500
–0.1%	$50\alpha N_p$	700	1000

In the case where an error of –5% is acceptable and a two phase clock is used, then for most circuits *maxsideband* should be set to $14 \le N_{msb} \le 20$. The circuit of Listing 1 is atypical in that it has the unusually large value of $\alpha = 43.5$, and so $N_{msb} \cong 87$. This large value of α is used as a way of making it easier to distinguish between v_t and v_h in v_c. With a large value of α, v_t is essentially constant over the band of interest, so any variations in that band result from v_h.

To illustrate the effect that the value of the *maxsideband* parameter has on the results produced by SpectreRF, the netlist in Listing 1 was run with a range of values for *maxsideband* (and corresponding values for *maxacfreq*). The results are shown in Table 2 as well as in Figures 4 and 5. Remember that because α is so large the value of *maxsideband* needed to achieve accurate results in this example are considerably higher than what is needed for a more typical switched-capacitor circuit.

2.2.2 Results

The circuit of Figure 1 was simulated with SpectreRF, with C = 10 pF, f_c = 400 kHz, and $m = 0.4$. This section compares the results generated by SpectreRF against the predictions made in the beginning of Section 2.2.

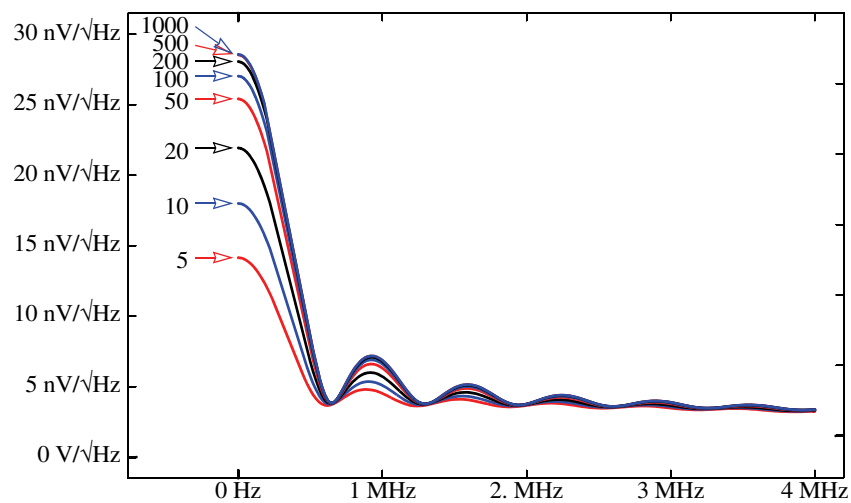
SpectreRF used the analysis *unsmpledNoise* to compute $S_c(f)$. At DC $S_c(0)$ can be computed from (4), (15), and (16)

$$S_t(0) = 4mkTR_{on} = 4\left(\frac{2}{5}\right)(13.81\times10^{-24})(300)(2.3\ \text{k}\Omega) = 15\ \text{aV}^2/\text{Hz}. \quad (27)$$

TABLE 2 *Effect of maxsideband on the number of time steps, the analysis time, and the total noise $\overline{v_s^2}$. The value of maxsideband needed to achieve accurate results in this example are considerably higher than what would be needed for a more typical switched-capacitor circuit.*

maxsideband	F_{norm}	***maxacfreq***	**Time Steps**	**Analysis Time**	**Total Noise**	**E from (21)**
5	0.29	6 MHz	228	0.8 s	9.28 µV	–58%
10	0.58	8 MHz	228	1.8 s	12.44 µV	–42%
20	1.2	12 MHz	228	2.7 s	15.49 µV	–25%
50	2.9	24 MHz	325	5.6 s	18.14 µV	–11%
100	5.8	44 MHz	572	19 s	19.36 µV	–5.6%
200	12	84 MHz	1070	67 s	20.13 µV	–2.7%
500	29	204 MHz	2565	404 s	20.51 µV	–1.1%
1000	59	404 MHz	5061	1620 s	20.48 µV	–0.54%

FIGURE 4 *$S_c(f)$, as computed with analysis unsmpledNoise in Listing 1 with various values of maxsideband.*



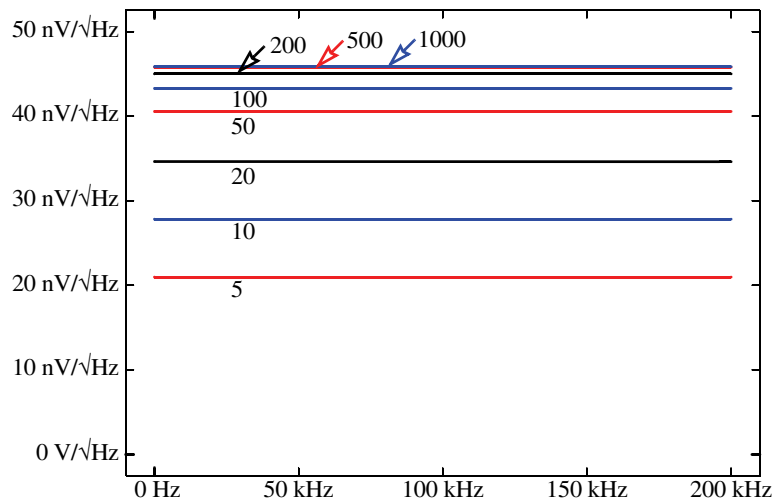
$$S_h(0) = 2(1-m)^2\frac{kT}{f_c C} = 2\left(1-\frac{2}{5}\right)^2\frac{(13.81\times10^{-24})300}{(400\text{ kHz})(10\text{ pF})} = 750\text{ aV}^2/\text{Hz}\,. \tag{28}$$

$$S_c(0) = S_t(0) + S_h(0) = 765\text{ aV}^2/\text{Hz}\,, \tag{29}$$

and so at DC the single-sided noise density is expected to be $27.7\text{ nV}/\sqrt{\text{Hz}}$, which is consistent with the result of $27\text{ nV}/\sqrt{\text{Hz}}$ computed by SpectreRF, as shown in Figure 4.

The sinc function in (16) has nulls where its argument is a multiple of 1 (is a non-zero integer), or where $f = n/((1-m)T_c)$, $n = 1, 2, \ldots$. At these frequencies f_n, the contribution from $S_h(f)$ goes to zero, and so $S_c(f_n) = S_t(f_n)$. At the first of these, $f_1 = 666.7$ kHz, the noise is computed from (4) to be $3.9\text{ nV}/\sqrt{\text{Hz}}$, which is close to the value of $3.85\text{ nV}/\sqrt{\text{Hz}}$ computed by SpectreRF.

FIGURE 5 *$S_s(f)$, as computed with the analysis smpledNoise in Listing 1 with various values of maxsideband.*



$S_s(f)$ was computed using the analysis *smpledNoise*. This analysis uses the time-domain noise feature of SpectreRF (as indicated by the *noisetype* = *timedomain* as a parameter to the analysis). With this feature the user can specify a series of time points at which a sampled-data noise analysis will be performed. In this way, the user can see how the noise statistics evolves with time. However, in this case the sampled noise at only a single point ($t = 0$) is needed (as indicated by the *noisetimepoints* = [0] and *numberofpoints* = 1 parameters). Conceptually you can imagine SpectreRF sampling the output waveform every T_c seconds starting at $t = 0$ and then reporting the power spectral density of this sequence, S_s, which is shown in Figure 5.

Using (10) gives

$$S_s(f) = \frac{2kT}{f_c C} = \frac{2(13.81\times10^{-24})300}{(400\text{ kHz})(10\text{ pF})} = 2\text{ fV}^2/\text{Hz}, \tag{30}$$

and so the noise density is expected to be $45\text{ nV}/\sqrt{\text{Hz}}$, which is consistent with that computed by SpectreRF, as shown in Figure 5. From (11)

$$\overline{v_s^2} = \frac{kT}{C} = \frac{(13.81\times10^{-24})300}{10\text{ pF}} = 415\text{ pV}^2, \tag{31}$$

and so the total noise is expected to be 20 µV, which is consistent with that computed by SpectreRF ($(45\text{ nV}/\sqrt{\text{Hz}})\sqrt{200\text{ kHz}} = 20\ \mu\text{V}$).

2.3 Gain

The gain of a track and hold is measured by applying a small sinusoid at the input and measuring the amplitude of the signal at the output. This is somewhat complicated by two factors. First, the signal at the output may only approximate a sinusoid. Remember that it is being sampled at the clock rate. This means that care must be taken when defining what is meant by the output amplitude. With SpectreRF, the output amplitude is

defined by first performing a Fourier analysis on the output signal and taking the amplitude to be the amplitude of the first harmonic. Second, the input and output sinusoids may be at different frequencies. If the input frequency is small relative to the clock frequency, then generally the input and output are at the same frequency. But it is also common for the input frequency to near or above the clock frequency, in which aliasing acts to down convert the input signal to baseband. In this case, designers are purposely exploiting aliasing to down-convert a signal. SpectreRF handles both cases.

With SpectreRF there are two analyses commonly used to measure small-signal gain. They are referred to as the PAC and PXF analyses. The PAC (periodic AC) analysis is similar in concept to SPICE's AC analysis, with the difference being that SpectreRF's PAC analysis performs a small-signal analysis about a periodically-varying operating point whereas SPICE's AC analysis assumes a time-invariant operating point. Thus, with SPICE the clock is effectively disabled before running an AC analysis, whereas SpectreRF's PAC analysis effectively performs the AC analysis with the clock present. This is a critical difference as the circuit does not operate as a track and hold if the clock is disabled.

With the clock applied, the circuit naturally translates signals from one frequency to another. A signal applied at frequency f_{in} can be translated to any frequency

$$f_{out} = kf_{clk} + f_{in} \tag{32}$$

where k is any integer. Similarly, a signal observed at a frequency f_{out} could have been injected into the circuit at any frequency

$$f_{in} = kf_{clk} + f_{out}. \tag{33}$$

As such, the PAC and PXF analyses compute the transfer functions. In particular, they compute the transfer functions associated with a range of values of k as defined by the *maxsideband* parameter, where PAC computes the transfer functions described by (32) and PXF computes the ones described by (33).

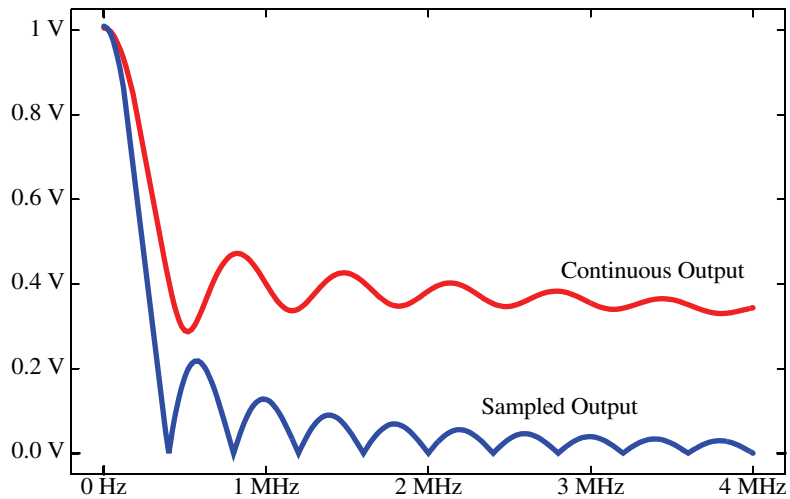
The netlist of Listing 1 uses the PAC analysis named *directGain* to compute the direct gain ($k = 0$) for the circuit. The PAC analysis applies a small sinusoid to the circuit whose frequency sweeps from *start* and *stop*. The sinusoid is applied at every source where *pacmag* is specified as being nonzero, in this case, on the input source *Vin*. The amplitude of the sinusoid is given by the *pacmag* parameter, however the circuit is linearized before performing the PAC analysis, so the output amplitude will scale linearly with the input amplitude, so generally one simply uses the most convenient value, in this case 1 since it allows us to directly compute the gain at the output. The last parameter specified on the PAC analysis is *maxsideband*. This parameter specifies how many output sidebands should be computed. Remember that the output frequency need not be the same as the input frequency. The output frequencies are given by

$$f_{out}(k) = kf_{clock} + f_{in}, \tag{34}$$

where k is the sideband index. In this case where we are computing the direct gain, the output frequency is the same as the input frequency, so $k = 0$. The PAC analysis will compute all of the sidebands for $k \leq maxsideband$; since we are only interested in the case where $k = 0$ the *maxsideband* parameter is set to 0. The results are shown in Figure 6. This result consists of two components, a direct feed through piece due to the signal in track mode, and a $\sin(x)/x$ piece due to the signal in hold mode. Often times one is only interested in the hold mode component. SpectreRF naturally considers the

whole signal†, to get it to ignore the track mode component one has to create a new signal that does not contain the track mode component. To do so, an idealized sample-and-hold is constructed with Verilog-A‡. It is used to sample the output of the track-and-hold during the hold phase, as shown in Figure 7. The sample-and-hold is specially designed to work with SpectreRF [3]. The transfer function from the input source to the output of the idealized sample-and-hold is also shown in Figure 6. Be aware that the idealized sample and hold effectively adds a half clock cycle of delay to the transfer function that should be manually removed.

FIGURE 6 *Direct transfer function from the input to the output of the track-and-hold (continuous output) and to the output of the idealized sample-and-hold that follows the track-and-hold (sampled output).*

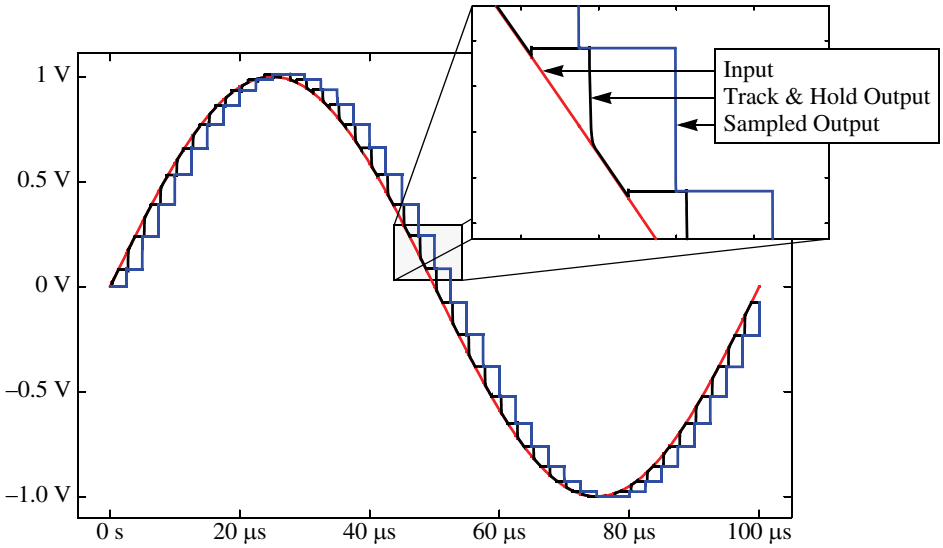


A important question for designers that are planning to use sampling nature of the track-and-hold to down-convert a signal to baseband is "what is the effective bandwidth of the track-and-hold when used in this mode". This question cannot be answered by looking at the direct gain as plotted in Figure 6 because this gain is colored by the sin(x)/x filtering that results from the zero-order hold nature of the track and hold. In fact, one cannot determine the overall effective bandwidth of the track-and-hold by looking at any one sideband because each is colored by sin(x)/x filtering. Instead, one must look at all of the sidebands together. It is when looking at the sum of all the sidebands that one can see the overall bandwidth of the track-and-hold. This is illustrated in Figure 8, where the gain of all sidebands up to ±25 is shown individually. Here it can easily be seen that the overall gain of the track-and-hold is dropping with frequency and that it reaches its 3 dB bandwidth at just over 6 MHz. This particular result would be difficult to measure using a PAC analysis. The PAC analysis allows one to specify a particular signal at a particular frequency at the input and then to determine the response at the output at any sideband. In Figure 8 the opposite is occurring, the transfer functions from the various sidebands at the input to a single output sideband are shown. This would require running many PAC analyses, each with the input set to run over a different range of frequencies and the

†. The sampling feature used earlier for the noise analysis is not available with the PAC or PXF analyses.

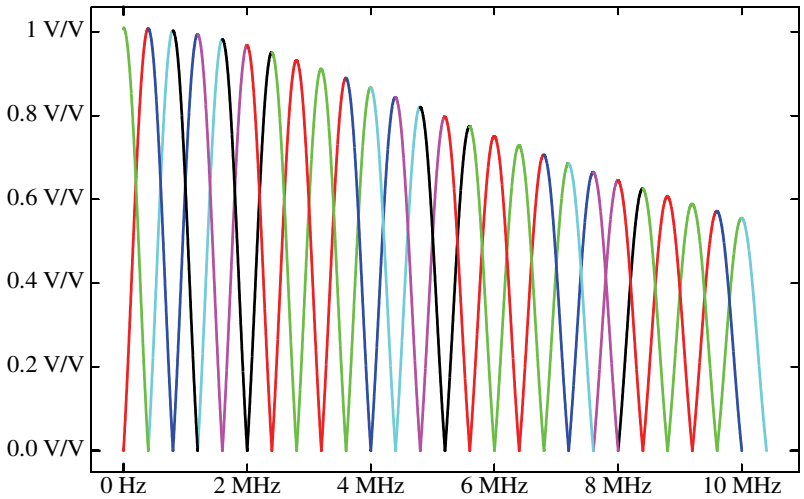
‡. Verilog is a trademark of Cadence Design Systems licensed to Accellera.

FIGURE 7 *The input and output waveforms for the track-and-hold. Also shown is the signal produced by the idealized sample-and-hold that follows the track-and-hold.*



output produced for a different sideband. Instead, one can use a PXF analysis. Whereas the PAC analysis computes the output signal at every node and every sideband given a single input, the PXF analysis computes the transfer function from every input source at every sideband to a single output. Thus, all of the sidebands displayed in Figure 8 were computed in a single PXF analysis. With the same analysis one could also determine the transfer function from each source. For example, the power supply rejection is available for free with a PXF analysis because it computes the transfer function from the supply source (at all sidebands) to the output.

FIGURE 8 *The transfer functions from the input at various sidebands (up to ±25) to the output at baseband.*



2.4 Distortion

Distortion can be expensive to calculate in switched-capacitor filters because narrow filter bandwidths combined with high clock rates require that many cycles be simulated when using a transient analysis. For lowpass filters, narrow bandwidths require that the input signal frequency be small so that both the test signal and its harmonics fit comfortable within the bandwidth of the filter. With bandbass filters, intermodulation distortion is the concern rather than harmonic distortion. In this case, one applies two input tones, both well within the filter bandwidth so that the intermodulation distortion tones also fall in-band. SpectreRF provides the Quasi-Periodic Steady-State analysis, or QPSS, for these situations. Quasiperiodic steady-state is a generalization of periodic steady-state. With periodic steady-state there is only one fundamental frequency (all frequencies present in the circuit are integer multiples of the fundamental frequency); while in quasiperiodic steady state there may be several fundamental frequencies. In the case of a lowpass filter there are two fundamental frequencies, that of the input signal and the clock. For bandpass filters there would be three; the two input signals and the clock. The benefit of QPSS analysis is that the time it requires is only dependent on the size of the circuit, and the number of frequency components that must be computed, and is not dependent on the actual frequencies chosen for the fundamentals.[†] Thus, which lowpass filters, the input signal frequency may be arbitrarily small without increasing the cost of the simulation. Similarly, with bandpass filters, the two input signals may be arbitrarily close.

The particular example being considered here, a simple track-and-hold, is not a narrowband and so it may be possible to carefully choose the input signal frequency to make either transient or PSS analysis reasonably efficient. However, for purposes of illustration, QPSS analysis will be used to compute both the harmonic and intermodulation distortion.

QPSS analysis differs somewhat from PSS analysis in that it expects you to identify the various fundamental frequencies by name. Notice that *Vphi* in Listing 1 has a parameter *fundname="clock"*. In this netlist, this parameter is only needed to support the QPSS analyses. It specifies that during the QPSS analysis *Vphi* will be producing one of the fundamental frequencies, and that the name of the fundamental frequency shall be *clock*. In the case where there are several sources that are used to generate a multiphase clock, all of the sources would share the same fundamental frequency and so would share the same *fundname*. Even in cases where the sources might be producing signals at different frequencies, they would still share the same *fundname* if one were a small integer multiple of another. This might be the case in switched-capacitor filters where early stages might have a higher clock frequency by a multiple of 2 or 4 than later stages.

The input source produces the second fundamental frequency. When computing intermodulation distortion, it also produces the third fundamental frequency. The parameters *fundname="input"* and *fundname2="input2"* are specified on the input source *Vin*. In the example given in Listing 1 these tones are initially disabled by setting *type = dc* on

†. Many engineers are often surprised and somewhat skeptical that the time required for the analysis could be independent of the ratio between the clock frequency and beat frequency. The QPSS analysis accomplishes this by formulating the circuit equations partially in the frequency domain. The way it does so is beyond the scope of this paper, but you can learn more about the way it works by reading about the Mixed Frequency/Time or MFT method [4].

Vin. During the course of running the given analyses, the two input tones are enabled one at a time, first by setting *type* = *sine*, and then by setting *ampl2* = 1.

The QPSS analysis takes a list of *fundnames* to determine the number of fundamentals, it then internally queries the sources to determine the fundamental frequencies. The first *fundname* should be associated with the clock, and the remaining should be associated with sinusoidal sources. QPSS also takes the list of maximum harmonics (*maxharms*) considered for each fundamental. It will compute the response of the circuit at all of the specified harmonics for each fundamental, plus at each of the associated intermodulation frequencies. Thus, it will compute the response at the output at each frequency f where $f = k_0 f_{clk} + k_1 f_1 + k_2 f_2 + \ldots$ where $|k_i| \leq maxharm[i]$ and $f \geq 0$.

The netlist in Listing 1 includes two QPSS analyses. The first, *harmDisto*, computes the quasiperiodic steady-state solution with 2 fundamentals, the clock and one input tone. This is useful for computing the harmonic distortion of lowpass filters. The second, *intermodDisto*, computes the quasiperiodic stead-state solution with 3 fundamentals, the clock and two input tones. The results at the sampled output for *harmDisto* are shown in Table 3. The distortion in the results is caused by the nonlinear charge injection from the CMOS switch.

TABLE 3 *Distortion of the simple track-and-hold.*

harmonic	sout
0	5 mV@ DC
1	1 V @ 10 KHz
2	–69 dB @ 20 KHz
3	–76 dB @ 30 KHz

The *intermodDisto* analysis represents the traditional approach to measuring the 3rd order intermodulation of a narrowband filter. Though not described here, a much more efficient approach that combines a two-fundamental QPSS analysis (the clock and one large input tone) and a quasiperiodic small-signal analysis to predict the intermodulation distortion can be used. [2]

3 Lowpass Filter

The switched-capacitor filter being simulated is a 5-pole 2.2 kHz low-pass elliptic filter. The filter was characterized using the sequence of analyses shown in the netlist given in Listing 3. Each analysis, along with its results, is described in the following sections.

3.1 PSS Analysis *clockAlone*

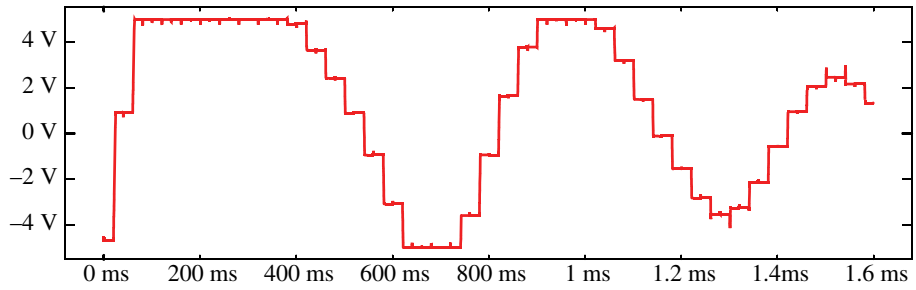
This analysis computes the steady-state response of the circuit with only the clock applied. The results of this analysis are used to measure the offset voltage at the output of the filter that results from the offset voltage of the op-amps and charge injection from the switches into the integrators. This analysis is also a prerequisite to the periodic small-signal analyses that follow because it sets the periodic operating point.

LISTING 3 *The analyses run on the switched-capacitor filter. The full set of files associated with this example can be found at www.designers-guide.org/Analysis.*

```
// PSS analysis (sets periodic operating point)
clockAlone                pss      fund=Fclk saveinit=yes maxacfreq=6MHz \
                                   writefinal="%C:r.ic" tstab=1.5ms swapfile="swap"

// Measure transfer functions (use if only interested in gain)
TFin                      pac      stop=10kHz lin=200

// Measure transfer functions (use to get all transfer functions including PSR)
unsmpldTFall    (out 0)   pxf      stop=10kHz lin=200 maxsideband=5
smpldTFall      (sout 0)  pxf      stop=10kHz lin=200 maxsideband=5

// Measure noise
unsmpldNoise    (out 0)   pnoise   start=100 stop=25kHz maxsideband=200
smpldNoise      (out 0)   pnoise   start=0_Hz stop=0.5*Fclk maxsideband=200 \
                                   noisetype=timedomain noisetimepoints=[0] \
                                   numberofpoints=1

// Measure harmonic distortion
enableVin                 alter    dev=Vin param=type value=sine
harmDisto                 qpss     funds=["clock" "input"] maxharms=[0 5] tstab=10/Fin
```

The PSS analysis, like a conventional transient analysis, starts off from an initial condition. If you do not specify an initial condition, then Spectre uses a DC analysis to determine the initial condition. During a DC analysis, the clocks are not operating, so the integrators have no feedback, causing their outputs to get stuck at the rails. The initial condition computed during the DC analysis causes the filter to react wildly in transient analysis as shown in Figure 9, bouncing off the rails several times before it settles. This can cause convergence difficulties during PSS analysis that could be resolved several ways. One approach that works in this case is to simply specify zero initial conditions for all capacitors in the circuit. This avoids a bad starting point. Another approach that is widely applicable when confronting convergence problems in PSS analysis is to use the *tstab* parameter, which delays the start of the PSS shooting loop. In this example, *tstab* = 1.5 ms, which causes the transient analysis to be performed to at least t = 1.5 ms before the PSS analysis attempts to compute the steady-state solution. Waiting until t = 1.5 ms allows the simulator to get beyond the difficult time when the filter is bouncing off the rails, which results in PSS analysis converging easily.

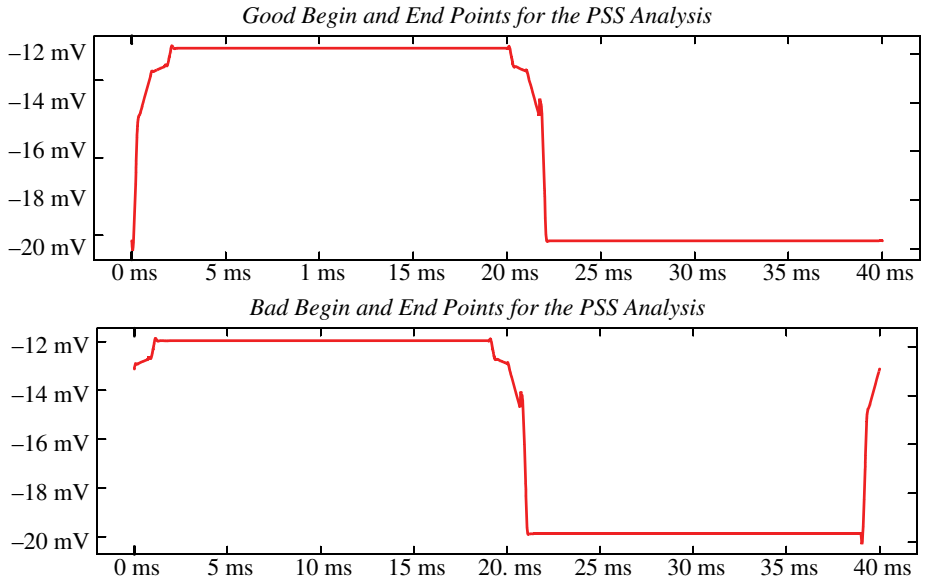
FIGURE 9 *The transient behavior of an internal node in the filter shows clipping that results from using the DC operation point as the initial condition. The clipping causes convergence problems for PSS analysis. Convergence is easily achieved either by setting tstab = 1.5 ms so that the PSS analysis gets beyond the clipping, or specifying initial conditions on the capacitors to avoid the clipping.*



Once the steady-state response has been computed, recomputing could be accelerated by saving the final point computed by the PSS analysis and using it as an initial condition for subsequent PSS analyses. To do so, replace *tstab = 1.5ms* with *readic = "%C:r.ic"*. Doing so first avoids simulating through a long *tstab* interval, and second causes it to read the initial condition from a file. The name of the file is determined by interpreting the string "%C:r.ic". Spectre replaces %C:r with the root of the name of the input circuit file. Thus, if the input netlist is contained in *filter.scs*, then Spectre will look for the initial conditions in *filter.ic* (the %C is replaced with *filter.scs*, the :r causes the *.scs* extension to be removed, and the *.ic* is added to the end to complete the name). The initial condition file is updated after every simulation as a result of the *writefinal = "%C:r.ic"*. It causes the final condition at the end of the PSS analysis to be written to the same file. Remember that because the steady-state response is periodic, the final point is the same as the initial point.

In order to further improve the efficiency of the PSS analysis, the phasing of the clock signals relative to the simulation interval is chosen carefully. It is best to have the simulation interval begin and end at points where the signals are not changing abruptly. For example, the phasing shown in the top of Figure 10 results in convergence in fewer iterations and less time than the phasing shown in the bottom.

FIGURE 10 *An example of good begin and end points for PSS analysis is shown in the top figure. The signal is settled and unchanging immediately prior to the end of the analysis. Contrast this with the situation shown in the lower figure, in which the waveform is changing abruptly at the end point.*



The PSS analysis in Listing 3 specifies the swapfile parameter. This is useful when simulations become very large. The swapfile is a temporary file that SpectreRF uses to hold data used in intermediate calculations. In doing so, SpectreRF trades a small amount of performance for a considerably smaller in-memory footprint, which allows you to simulate very large circuits without running out of physical memory, which would cause the simulations to run very slowly. The swapfile can become very large, so it is important to put it on a filesystem that has plenty of free space, at least several gigabytes. This circuit

is probably not large enough to justify using a swapfile; it was added simply as an example.

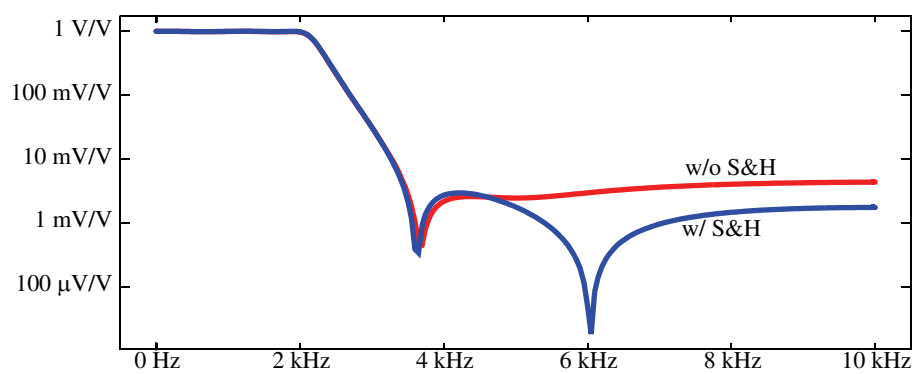
3.2 PAC Analysis *TFin*

The PAC analysis applies a “small” signal at the input and computes the response at two outputs. The first output is the normal output of the filter. The signal at this output is continuous in time and includes various imperfections such as glitches and regions of slew-rate limiting and settling. It also contains output from both phases of filter. This output is interesting if the filter is followed with a continuous-time filter. The second output is the first output after being passed through a sample-and-hold. This models the situation where the filter would be followed by a sampled-data circuit such as an analog-to-digital converter (ADC) for further processing. In this case, most of the imperfections on the normal output are ignored by the sampling nature of the ADC. Taking into account the sampling nature of the ADC is important when trying to measure the transfer function, the noise, or the distortion of any clocked analog circuit such as a switched-capacitor filter.

As before, an idealized sample-and-hold is constructed with Verilog-A and added to the circuit in order to produce the sampled-data output. This sample-and-hold is implemented in such a way that there is no hidden state that would prevent the RF analyses from being used. [3].

With this circuit, only the magnitude and phase of the response at the fundamental frequency is interesting. The small input signal is considered to have unit magnitude and so the transfer functions are computed directly. These transfer functions are shown in Figure 11. Notice that the second null is missing from the transfer function of the normal (continuous-time) output. This is a consequence of blending the response from both phases of the filter.

FIGURE 11 *The transfer function from the input voltage to the outputs. There are two possible outputs, each of which are used in different applications. The first is the normal output of the filter and the second is the normal output after being passed through a sample-and-hold.*



3.3 PXF Analyses *unsmpldTFall* and *smpldTFall*

These analyses directly compute the transfer function from all sources at all sidebands to either the unsampled or sampled output. They differ from the PAC analysis *TFin* in

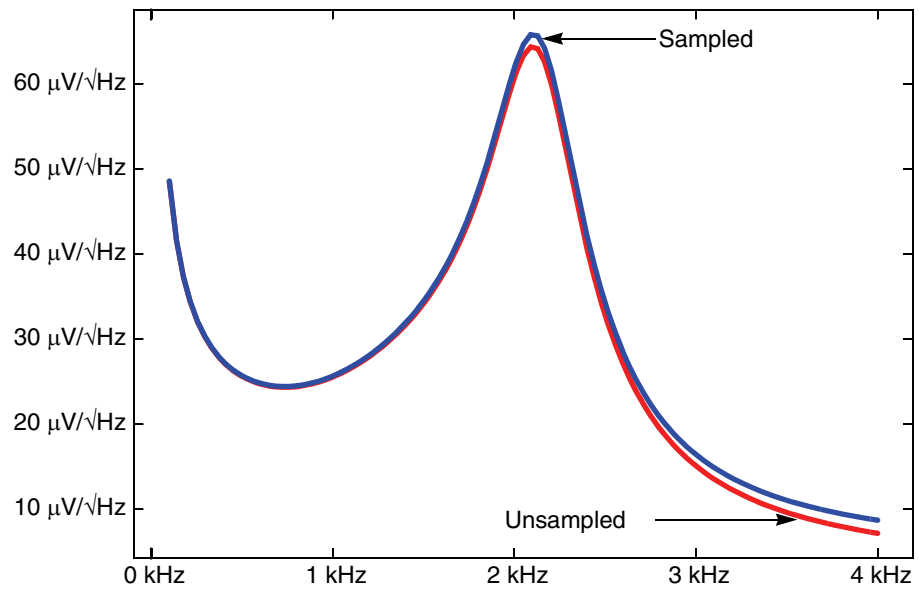
that they compute transfer functions from all sources to a single output rather than the transfer functions to all outputs from a single input. They are also somewhat slower than the PAC analysis. They are useful when measuring the ability of the filter to reject unwanted signals on the input near multiples of the clock frequency, as well as signals on the clock, on bias lines, and on the supplies.

The input-to-output gain is produced by both *TFin* and the two PXF analyses. Since the PXF analyses would be slower, they would not be used if only the gain were desired. Conversely, if the two PXF analyses are needed, then generally *TFin* can be skipped.

3.4 PNoise Analyses *unsmpldNoise* and *smpldNoise*

These analyses compute the total spot noise as a function of frequency at the sampled and unsampled outputs. The results are shown Figure 12, They include noise folding (noise being converted down from sidebands of the clock by the sampling nature of the switched-capacitor filter).

FIGURE 12 *Noise response of switched-capacitor filter including effects of noise folding.*



3.5 QPSS Analysis *harmDisto*

This analysis is used to compute the harmonic distortion produced by the filter.

4 Conclusion

This paper described how SpectreRF can be applied to some rather conventional switch-capacitor filters. However, it is important to realize that SpectreRF has the ability to similarly analyze a broad range of circuits that have similar characteristics. For example, circuits that attempt to suppress offsets and low frequency noise by using either chopper stabilization of correlated double-sampling can be directly simulated with

SpectreRF. CCD image sensors operate in a related way and given the right models can be analyzed with SpectreRF. Noise is a critical concern in image sensors, and SpectreRF's ability to accurately compute the noise should be tremendously useful. SpectreRF can also be used to predict the noise performance of pipelined ADC. In this case, one would generally use a constant-valued input signal and make the period for the PSS analysis equal to the cycle time for a complete conversion.

There are also switched-capacitor circuits for which SpectreRF cannot, or should not, be directly applied. SpectreRF needs to compute a periodic or quasiperiodic solution in order to operate, and circuits such as ΔΣ converters generally operate in chaotic steady-state and so cannot be directly simulated with SpectreRF. However, generally in these cases it is possible to break the system into smaller pieces that can be characterized by operating them such that they have a periodic solution, and SpectreRF can be used for this characterization. For example, a ΔΣ converter consists of conventional switched-capacitor filter inside a feedback loop with a nonlinear quantizer. The filter alone can be characterized with SpectreRF, and then hand calculations can be used to estimate the performance of the complete converter. [6]

4.1 Things to Remember

The following items are some of the key points presented in this paper. They are worth remembering when trying to simulate switched-capacitor filters.

1. SpectreRF has a unique ability to simulate switched-capacitor filters because it performs small-signal analyses about a periodically time-varying operating point. This allows for the effect of the clock to be considered during AC and noise analyses. This is critical, as switched-capacitor filters do not operate without their clock. (§1)
2. It is necessary to run a PSS analysis before running a small signal AC or noise analysis. This computes the periodic operating point. Be sure only the clock signal is applied (the input signal is turned off). (§2.1)
3. Use the *maxacfreq* parameter to communicate the highest small-signal analysis frequency to the PSS analysis so that it can choose the time-step accordingly. It is generally not necessary to do this proactively. Instead, you can wait for warnings from the small-signal analyses that indicate you should set *maxacfreq*. Be sure to look for them. (§2.1)
4. Convergence difficulties with the PSS analysis can usually be resolved by adjusting the *tstab* parameter. This parameter sets the length of the transient analysis that precedes the steady-state analysis. In general, you should set *tstab* so that the steady-state analysis begins after most of the initial transients have dissipated. You can use *saveinit* to specify that the PSS analysis saves the results from its initial transient analysis, which allows you to view the transient waveforms. It is also best to set *tstab* so that the boundaries of the steady-state analysis fall on the flat parts of the solution. In other words, the times where the steady-state interval begins and ends should be points where the signals are not changing, or changing only slowly. (§3.1)
5. You must understand up front the difference between the continuous-time and discrete time behavior of your circuit, and you must decide which behavior you are interested in. Results can be quite different between the two, as illustrated in Figures 4 and 5, Figure 6, and Figure 11. Usually, with switched-capacitor circuits, designers are interested in the discrete-time behavior as switched-capacitor filters are embedded in sampled-data systems. SpectreRF is set up to report on the continu-

ous-time behavior. As such, you must take steps to get the discrete-time behavior if that is what you need. The PNoise analysis does provide a built-in feature that can be invoked in order to determine the noise of a filter acting as a discrete-time system. Otherwise you will need to add a idealized sample-and-hold to your circuit. (§2.2, §2.3, and §3.2)

6. You can use Verilog-A to describe an idealized sample-and-hold that is used to isolate the discrete-time behavior of your circuit, but you must take care to avoid hidden state, which makes the model unusable with PSS or QPSS [3]. Also, a sample-and-hold adds a sin*x*/*x* coloring to the results of which you must be aware. You can reduce this coloring by shrinking the output pulse width, but do not increase pulse height to compensate unless you use *relref* = *alllocal*. (§2.3)
7. In noise analyses it is important to set the *maxsideband* parameter to accurately account for noise folding. Setting it too low results in a systematic underestimation of the noise whereas setting it to high results in excessively long simulations. The best approach to setting the *maxsideband* parameter is to increase it until the noise results stabilize, then back off a bit to achieve an acceptable level of accuracy. Once *maxsideband* is set, go back to the PSS analysis and make sure *maxacfreq* is set accordingly. (§2.2)
8. Transfer functions can be measured either with a PAC or a PXF analysis. PAC measures transfer functions from a single input (or a single composite input, such as a pair of inputs configured as a differential input) to any output (any node voltage or terminal current at any sideband), whereas PXF measures the transfer function from any input (any source at any sideband) to a single output (node voltage (floating or not) or terminal current). If you have a single input and output, either analysis works equally well. PAC is preferred if there are more outputs than inputs and PXF is preferred if there are more inputs than outputs. When performing a PAC, you must set *pacmag* in order to identify the driven input. The output scales linearly with *pacmag*, so generally it is most common to simply set it to 1 to cause PAC to directly compute the transfer functions. (§2.3)
9. The amplitude of the signals computed by PAC and PXF is the magnitude of the Fourier coefficient of the corresponding signal component. Some signals produced by switched capacitor circuits, particularly those involve in correlated double sampling, do not have a 100% duty cycle. In these cases the amplitude produced by SpectreRF will be scaled relative to the amplitude of the envelope by the duty cycle. (§2.3)
10. QPSS analysis efficiently computes the harmonic distortion or intermodulation distortion of narrowband switched-capacitor filters because unlike with either transient or PSS analyses, the time required by QPSS analysis is independent of the spacing of the input frequencies. In addition, in most cases the rapid approach to computing intermodulation distortion described in [2] can be used to accelerate the simulation still further. (§2.4)

4.2 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 Comunity* website. Do so by going to *www.designers-guide.org/Forum*.

Acknowledgements

Thanks to those individuals who have reported errors in this manuscript. Alberto Espinosa reported errors in Figure 3 and (16), and suggested the addition of (17). Saeed Chehrazi reported a factor of 2 error in (9). Peter Kurahashi reported an extra factor of $1/2\pi$ in (3). And Paul Geraedts reported errors in the lead up to (18). He also reported that the extra factor of $1/2\pi$ that Peter found was also in (18).

If you find any errors or have suggestions concerning this paper, contact the author by emailing to *ken@designers-guide.com*. Technical questions about the this document's contents should be posted on the forum as described in Section 4.2.

References

[1] R. Gregorian and G. C. Temes. Analog MOS Integrated Circuits for Signal Processing. Wiley-Interscience, 1986.

[2] Ken Kundert. Accurate and Rapid Measurement of IP_2 and IP_3. In *www.designers-guide.org/Analysis*.

[3] Ken Kundert. Hidden state in SpectreRF. In *www.designers-guide.org/Analysis*.

[4] Ken Kundert. Introduction to RF simulation and its application. In *www.designers-guide.org/Analysis*.

[5] Ken Kundert. Noise in mixers, oscillators, samplers, and logic: an introduction to cyclostationary noise. In *www.designers-guide.org/Theory.*

[6] Manolis Terrovitis and Ken Kundert, Device noise simulation of $\Delta\Sigma$ modulators. In *www.designers-guide.org/Analysis*.

[7] SwitCap, *www.cisl.columbia.edu/projects/switcap*.