

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/295256513>

CMOS Mixed-Signal Circuit Design, First Edition

Book · August 2002

CITATIONS

4

READS

238

1 author:



R. Jacob Baker

University of Nevada, Las Vegas

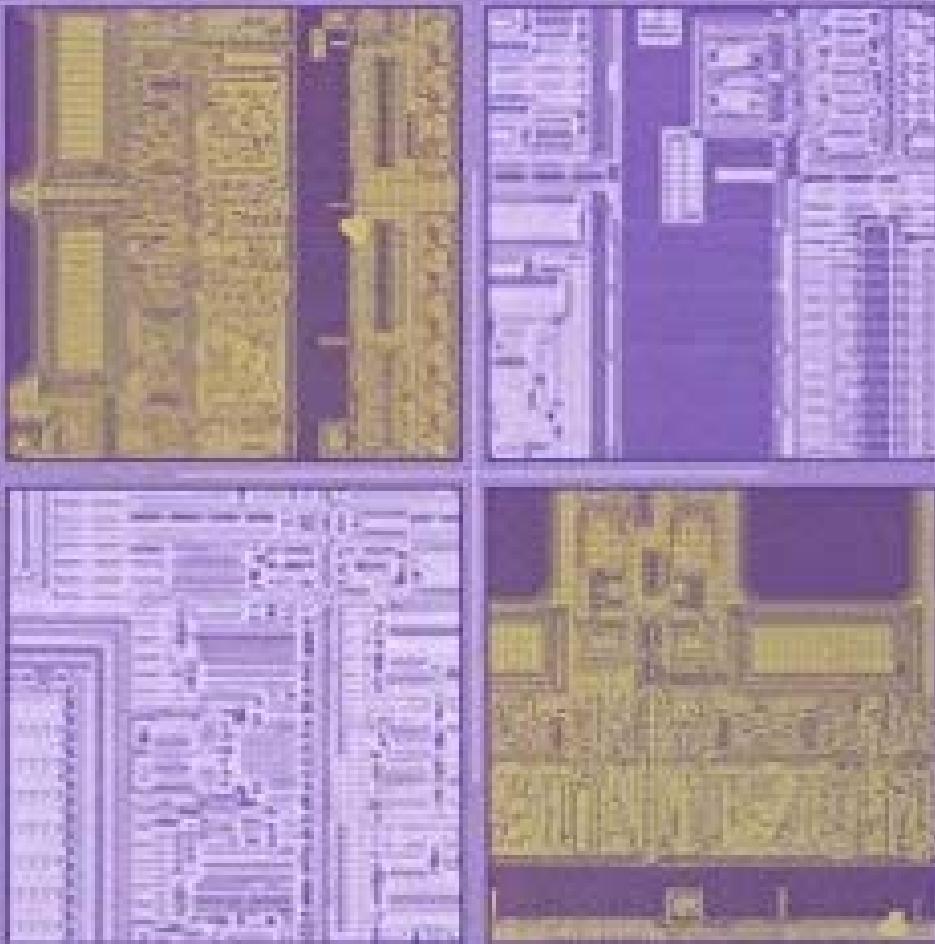
307 PUBLICATIONS 2,736 CITATIONS

[SEE PROFILE](#)

*The essential companion to the bestselling
CMOS: Circuit Design, Layout, and Simulation*

CMOS

MIXED-SIGNAL CIRCUIT DESIGN



R. Jacob Baker

IEEE Press Series on Microelectronic Systems
SOLID STATE TECHNOLOGY AND HIGH-DENSITY CIRCUITS

Copyright © 2000

Chapter

30

Data Converter Modeling

In this chapter we continue our discussion of data converters by discussing methods to model ideal data converters and their components using SPICE. The main goal of this chapter is to provide tools for evaluating mixed-signal designs with large complexity, which can be used later in the book. In particular, we will generate SPICE models, using behavioral elements, for ideal analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) blocks. This allows us to analyze the performance of a mixed-signal circuit block in a SPICE simulation within a reasonable amount of time. For example, if we have designed a DAC at the transistor level and want to use SPICE to simulate its operation, under various temperatures and matching conditions, we may apply a digital input code generated from our ideal ADC with a sinewave input as seen in Fig. 30.1. Similarly, given a digital signal processing (DSP) system, we can drop our ideal DAC into the simulation at any point where there is a digital word and get an analog waveform output.

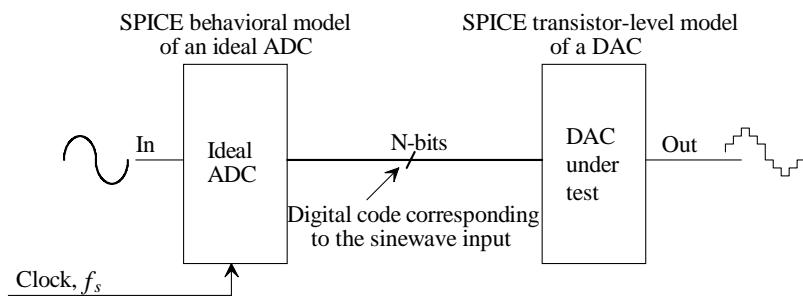


Figure 30.1 Generating the sinewave digital code for DAC simulation with an ideal ADC.

Also, in this chapter we look at how the analog-to-digital and digital-to-analog conversion process affects the signals in the system. Figure 30.2 shows the basic conversion process. We will make extensive use of the spectral analysis capability (discrete Fourier transform or DFT) available in SPICE to look at the digital data (and analog signals) in the frequency domain.

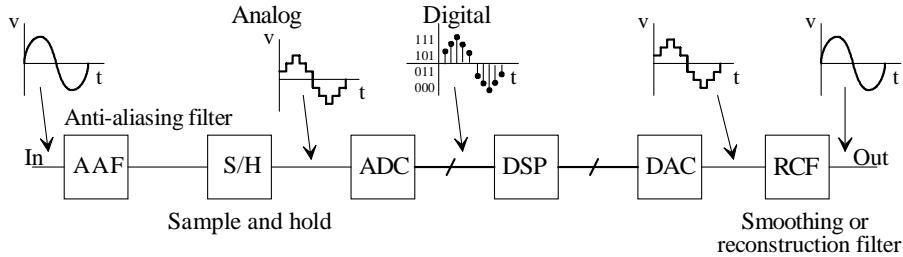


Figure 30.2 Signals resulting from A/D and D/A conversion in a mixed-signal system.

30.1 Sampling and Aliasing: A Modeling Approach

In this section we discuss how sampling a signal changes the signal's spectrum. We also discuss how to model the sampling process in SPICE.

30.1.1 Impulse Sampling

Consider the simple sampling gate shown in Fig. 30.3a. Let's assume we apply a sinewave input, $x(t)$, to this sampling gate of the form, $V_p \sin(2\pi f_{in} \cdot t)$ (for the moment, a single frequency input). The output of the sampling gate (a.k.a. sampler), $y(t)$, is the product of the input and a sampling unit impulse signal or

$$y(t) = \sum_{n=-\infty}^{\infty} V_p \sin(2\pi f_{in} \cdot nT_s) \cdot \delta_u(t - nT_s) \quad (30.1)$$

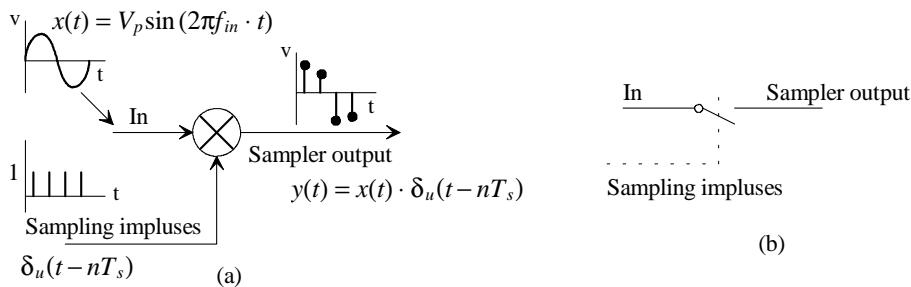


Figure 30.3 (a) Simple sampling gate and (b) SPICE implementation of a sampling gate.

Noting that the frequency of the input is f_{in} while the sampling frequency is $f_s (= 1/T_s)$, the spectrum of the input signal is seen in Fig. 30.4a. If we take the Fourier transform of the input signal after sampling, that is, we look at the spectrum on the output of the sampler, we get

$$Y(f) = \frac{V_p}{T_s} \cdot \sum_{k=-\infty}^{\infty} [\delta(f - f_{in} + kf_s) + \delta(f + f_{in} + kf_s)] \quad (30.2)$$

This is the familiar result that a sampled spectrum is repeated, at intervals of f_s , as seen in Fig. 30.4b (shown is the one-sided spectrum, which is what we will use throughout the book). Note that if an ideal lowpass filter (LPF) is applied to the output spectrum of the sampler (the output of the sampler is connected to an LPF) with a bandwidth greater than f_{in} (and lower than f_n [the Nyquist frequency]), then the higher order frequency components can be removed so that only f_{in} remains (this is our smoothing or reconstruction filter shown in Fig. 30.2).

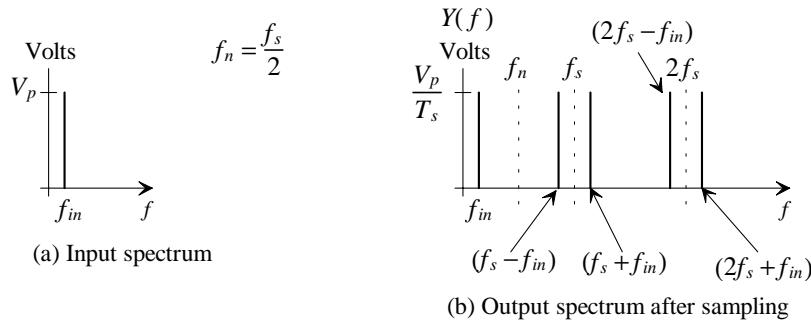


Figure 30.4 One-sided spectrum of a sinewave (a) before and (b) after sampling.

Example 30.1

A sampling gate is strobed with an impulse train running at a frequency of 100 MHz ($f_s = 100$ MHz and the time between the impulses, T_s , is 10 ns). Sketch the resulting output frequency spectrum if a 60 MHz sinewave is applied to the sampler. Also, sketch the time domain input and output of the sampler.

The resulting frequency spectrum is shown in Fig. 30.5. Notice how connecting the output of the sampler through an LPF, with an ideal abrupt cutoff frequency of f_n , results in an output sinewave with a frequency of 40 MHz. In order to avoid this situation, that is, to avoid ending up with the wrong, or alias, signal after sampling and reconstructing, we need to ensure that the signal frequencies applied to the sampler are less than $f_s/2$ (the Nyquist frequency, again, f_n). Reviewing Fig. 30.2, we see that this is the purpose of the antialiasing filter (AAF). Notice how, ideally, both the AAF and RCF (reconstruction filter) in Fig. 30.2 are both ideal LPFs with a cutoff frequency equal to half the sample frequency (the Nyquist frequency). Figure 30.6 shows the time domain sketch of the sampler's output. ■

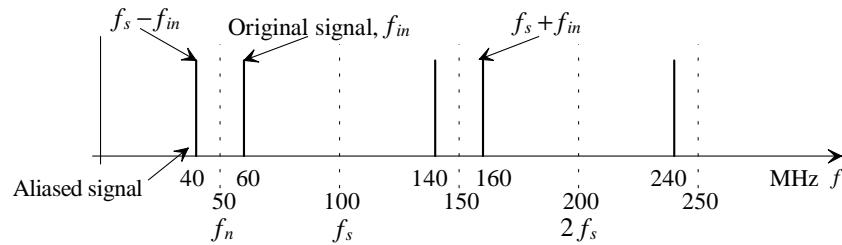


Figure 30.5 Spectrum of a 60 MHz sinewave sampled at 100 MHz.

It should be clear from the preceding discussion that (1) sampling a signal results in a reproduction of the sampled signal's spectrum at DC, f_s , $2f_s$, $3f_s$, etc., (2) the input signal's spectrum should have no significant spectral content above f_n in order to avoid aliasing, (3) to avoid aliasing both filtering the input signal using an AAF and increasing the sampling frequency should be used, and (4) to reproduce the sampled signal from the output of the sampler (which is nonzero only during the sampling impulse times) a lowpass RCF should be used.

Note that our discussion illustrates the operation of a sampling gate driven with impulse signals. As shown in Fig. 30.2, a practical system would have other building blocks. We would rarely, if ever, sample a signal and then reconstruct it without processing it first.

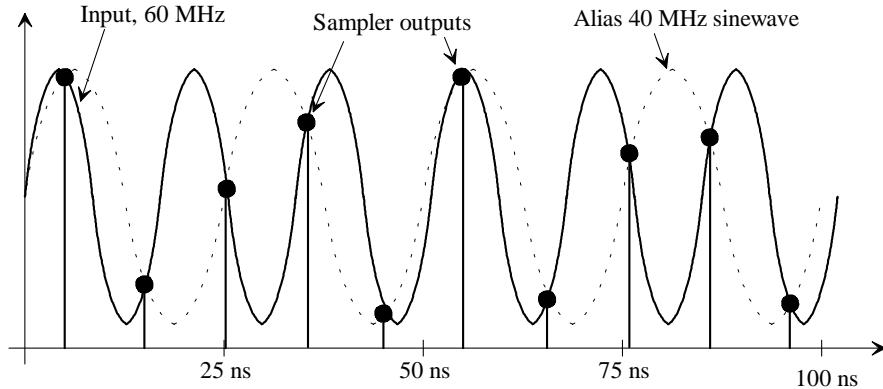


Figure 30.6 Time domain input and output for Ex. 30.1.

A Note Concerning the AAF and the RCF

Before going any further, we should discuss the ideal characteristics of the AAF and the RCF. The ideal characteristics of these filters are shown in Fig. 30.7. Note that both of

these filters must be analog by design. The ideal cutoff frequency for the filters is f_n (assuming the sampling rate on the input of the system is the same as the sampling rate on the system's output) and the filters should ideally have linear phase. Let's discuss these two ideal characteristics.

The ideal magnitude response, shown in Fig. 30.7a, passes all spectral content below the Nyquist frequency while removing all signals above this frequency. The ideal phase response, shown in Fig. 30.7b, provides a constant delay, t_o , to all signals below f_n . In other words, the filters remove all unwanted signals while not distorting the wanted signals.

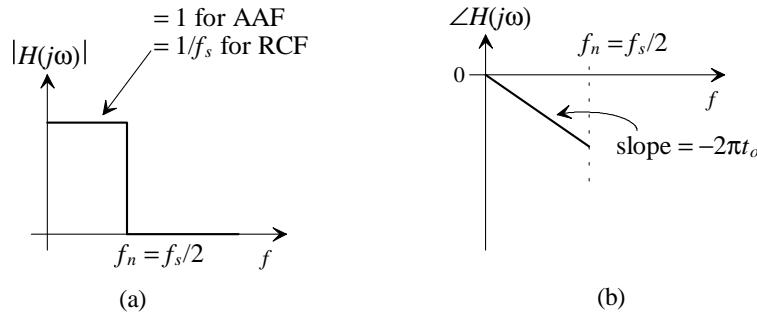


Figure 30.7 (a) Ideal magnitude and (b) phase responses for the AAF and RCF.

Example 30.2

Discuss why the ideal AAF filter will not introduce distortion into the desired portion of an input signal.

If our input signal is called $v_{in}(t)$ and the desired spectral content of this signal after filtering is called $v'_{in}(t)$ (that is, $v'_{in}(t)$ contains nonzero spectral content only at frequencies below f_n), then the output of the AAF, $v_{out}(t)$, will be a time-shifted (with a constant delay of t_o) and filtered version of the input, as seen in Fig. 30.8. Note that *linear phase* is equivalent to saying "constant delay." If our input signal is already bandlimited to f_n , then the output of the AAF is simply a time-shifted version of the input. ■

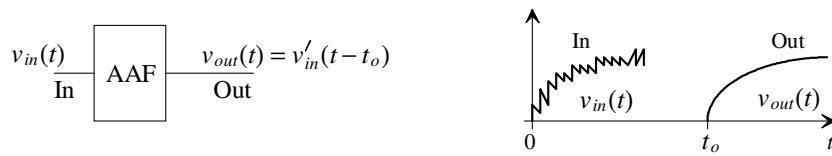


Figure 30.8 Results for Ex. 30.2.

Example 30.3

Suppose that the circuit, shown in Fig. 30.9, is used as an AAF filter in a data conversion system. If the inputs to the system are two sinewaves with frequencies of 4 MHz and 40 MHz determine whether the waveforms coming out of the AAF will be distorted. Using SPICE show the input and output signals of the AAF.

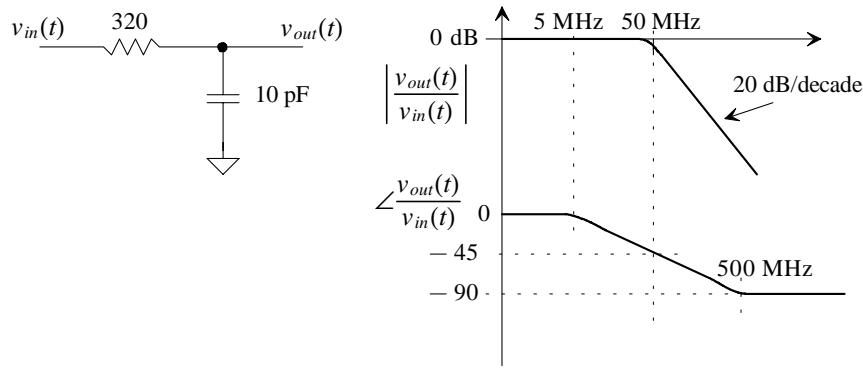


Figure 30.9 AAF filter for Ex. 30.3.

The amplitude response of the simple RC filter is given by

$$\left| \frac{v_{out}}{v_{in}} \right| = \frac{1}{\sqrt{1 + (2\pi \cdot RC \cdot f)^2}}$$

The 4 MHz input doesn't see any attenuation. The gain, or amplitude response, of the filter at 4 MHz is unity (0 dB). The filter attenuates the 40 MHz input by 0.779 (-2.17 dB).

The phase response of the simple RC filter is given, in degrees by

$$\angle v_{out}/v_{in} = \theta(f) = -\tan^{-1}(2\pi \cdot RC \cdot f)$$

The phase shift through the filter at 4 MHz is approximately zero (the 4 MHz input doesn't see any delay while passing through the filter). This is the ideal phase response of this filter, i.e., \$t_o = 0\$. Looking at Fig. 30.9 we can conclude that only at frequencies below approximately 5 MHz will the filter not exhibit phase distortion. The phase shift through the filter at 40 MHz is \$-39^\circ\$ (the negative sign indicates that the output is lagging the input or, in other words, occurs later in time than the corresponding point on the input). Since phase is related to delay by

$$\theta(f) = \underbrace{\frac{t_o}{T}}_{\% \text{ of period, } T} \cdot 360 = t_o \cdot f \cdot 360$$

the delay the 40 MHz sinewave sees passing through the filter is 2.7 ns. The SPICE simulation results are shown in Fig. 30.10 assuming each sinewave input is centered at ground and has an amplitude of 1V.

Also note that this filter does a *poor* job attenuating frequencies above 50 MHz. For example, the attenuation at 500 MHz (one decade above 50 MHz) is only -20 dB (0.1). It can be concluded that unless $f_s/2$ (the Nyquist frequency) is much larger than the cutoff frequency of the simple RC LPF aliasing will (possibly) still occur in significant amounts. In fact, we could argue that because of the inherent noise present in any electronic circuit, aliasing will always occur when sampling a signal (the wideband noise gets aliased down into the base spectrum [the spectrum below the Nyquist frequency]). The question then becomes, "How much aliasing is OK?" ■

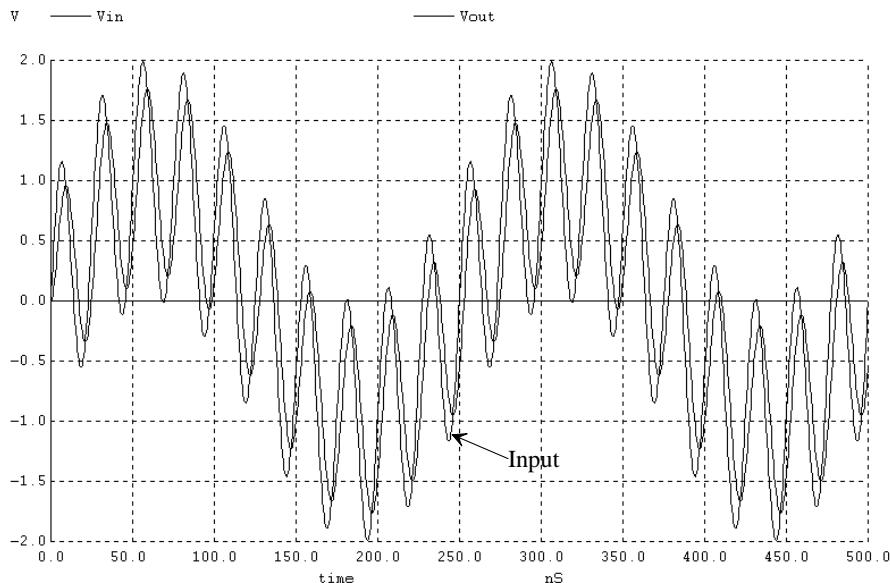


Figure 30.10 SPICE simulation results for Ex. 30.3.

Example 30.4

Determine the transfer function of the filter made with a 5 ns long ($= t_o$) ideal transmission line shown in Fig. 30.11. Simulate the filter's frequency and phase response using SPICE. This filter is called a continuous-time comb filter.

In this analysis, we assume that 500-ohm resistors do not load the input or output of the delay line so that

$$v_{out} = \frac{v_{in} + v_{in} \cdot e^{j2\pi f(-t_o)}}{2}$$

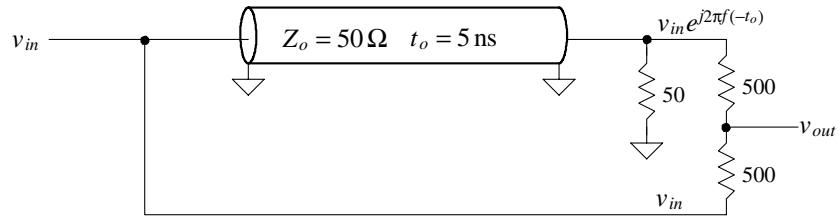


Figure 30.11 A continuous-time comb filter.

The transfer function can be written as

$$\frac{v_{out}}{v_{in}} = \frac{1}{2}(1 + e^{j2\pi f \cdot (-t_o)}) = \frac{1}{2} \left(\overbrace{1 + \cos 2\pi f \cdot (-t_o)}^{\text{Real}} + j \cdot \overbrace{\sin 2\pi f \cdot (-t_o)}^{\text{Imaginary}} \right)$$

The magnitude response of this filter is

$$\left| \frac{v_{out}}{v_{in}} \right| = \frac{1}{2} \sqrt{(1 + \cos 2\pi f \cdot (-t_o))^2 + (\sin 2\pi f \cdot (-t_o))^2} = \frac{1}{2} \sqrt{2(1 + \cos 2\pi f \cdot t_o)}$$

The phase response of this filter is given by

$$\angle \frac{v_{out}}{v_{in}} = \tan^{-1} \left[\frac{\sin 2\pi f \cdot (-t_o)}{1 + \cos 2\pi f \cdot (-t_o)} \right]$$

Notice at $f = 1/(2t_o)$ the phase is $\tan^{-1}(0/0)$, which evaluates to $\pm 90^\circ$. Using

$$\cos^2 x = \frac{1 + \cos 2x}{2} \quad \text{and} \quad \sin 2x = 2 \sin x \cos x$$

the phase response is given by

$$\angle \frac{v_{out}}{v_{in}} = \pi(-t_o) \cdot f \quad \text{for } f < 1/(2t_o)$$

Notice that the phase response of this filter is linear. The SPICE simulation results, plotted on a linear frequency scale, are shown in Fig. 30.12. The reason this filter is called a "comb filter" should be obvious (the response looks like a comb). Notice how the delay line length is related to the points where the magnitude response goes to zero. Also note that this filter could be useful to isolate channels in a communication system and easily implemented on a PC board using a microstrip transmission line. The SPICE netlist that generated this figure is given below.

* Figure 30.12 CMOS: Circuit Design for Mixed-Signal Systems *

```
.AC LIN 1000 1MEG 1000MEG
Vin   Vin   0     DC   0     AC   1
Rtout Vtout  0     50
Rt1    Vtout  Vout  500
Rt2    Vin    Vout  500
```

```
T1      Vin      0      Vtout     0      ZO=50 TD=5n
.end
```

Finally, before leaving this example, consider the dB (magnitude) and phase responses of this filter on a log frequency plot, Fig. 30.13. It would appear that the magnitude of the transfer function at 100 MHz, 300 MHz, 500 MHz, etc., is nonzero. However, as the equation for the magnitude response shows, this isn't the case. At these frequencies $|v_{out}/v_{in}|$ is zero indicating, in the plots shown in Fig. 30.13, that the lower limit is set by step size (number of points per decade) used in AC SPICE simulation. Increasing the number from 1,000, which is what was used to generate Fig. 30.13 to, say, 10,000 will give more accurate results. ■

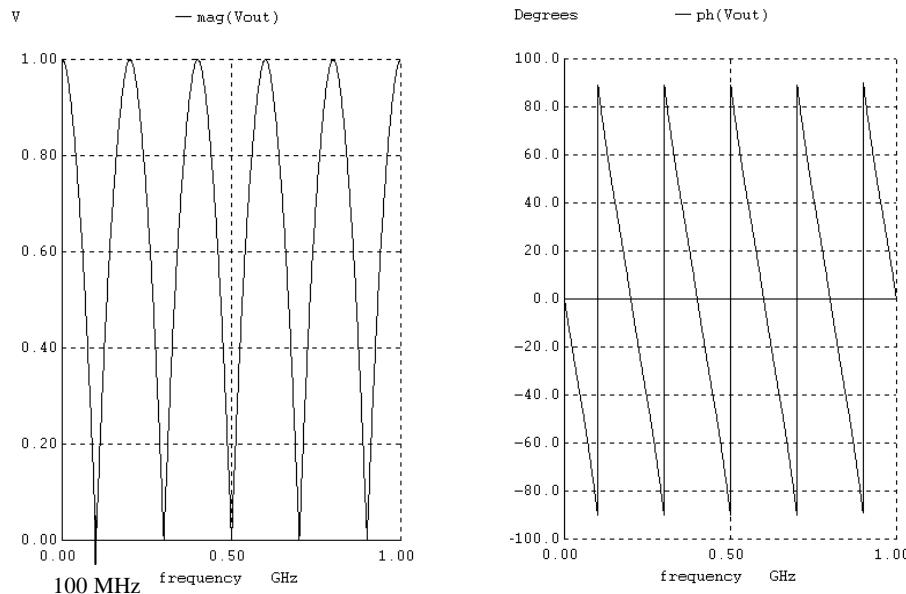


Figure 30.12 SPICE simulation results for the comb filter of Ex. 30.4.

Time Domain Description of Reconstruction

In this section we show why the filter shown in Fig. 30.7, an ideal brick wall lowpass filter with linear phase response, is the ideal RCF on the output of our impulse sampler. Shown in Fig. 30.14 is a 20 MHz sinewave sampled at 100 MHz. Suppose we want to reconstruct the original input 20 MHz sinewave from the sampler output (the weighted impulse functions). After reconstruction the output of the RCF should be a single-tone, 20 MHz sinewave (it should be an exact replica of the sampler input). To determine what happens when the output of our sampler is applied to the ideal RCF, we need to determine the time domain response of the filter when the filter's input signal is the Dirac delta function, $\delta(t)$ ($\delta(0) = \infty$, else $\delta(t) = 0$).

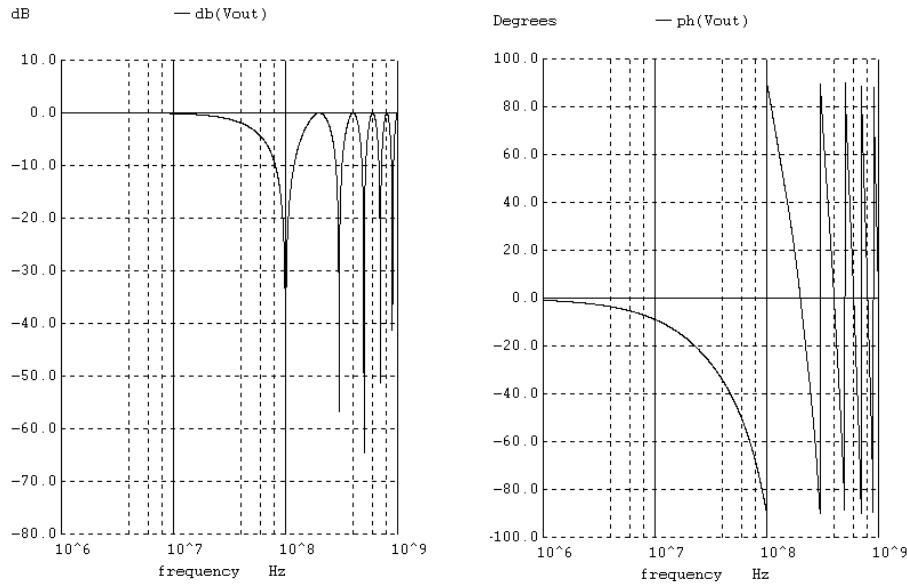


Figure 30.13 SPICE simulation results, in dB, for the comb filter of Ex. 30.4.

We know the transfer function of a system is the Fourier transform of the system's time domain impulse response (what we are trying to find here). In other words, to determine the transfer function of the system, we apply the unit impulse to the input of the system (a very large amplitude, very short time duration pulse, see Fig. 30.15). We then look at the system's output in the time domain followed by taking the Fourier transform of this output to get the system's transfer function. Therefore (in the reverse order), to determine the time domain response of the ideal RCF, given the transfer function, we take

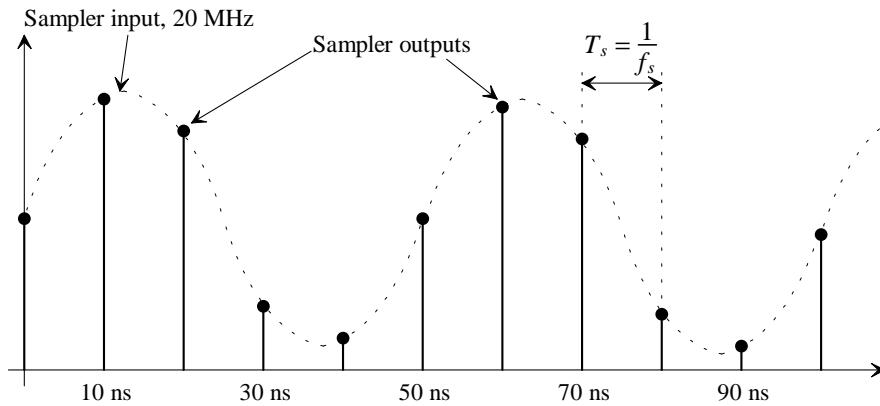
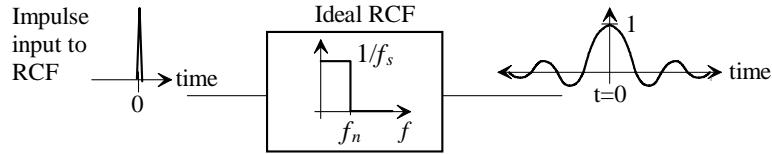


Figure 30.14 Impulse sampling, at 100 MHz, a sinewave at 20 MHz.

**Figure 30.15** Time domain impulse response of the ideal RCF.

the inverse Fourier transform of the transfer function. The ideal RCF's transfer function (Fig. 30.7) can be defined by

$$|H(f)| = 1/f_s \text{ for } |f| < f_n \text{ else } |H(f)| = 0 \quad (30.3)$$

The time domain response is then given, remembering $2f_n = f_s$, by

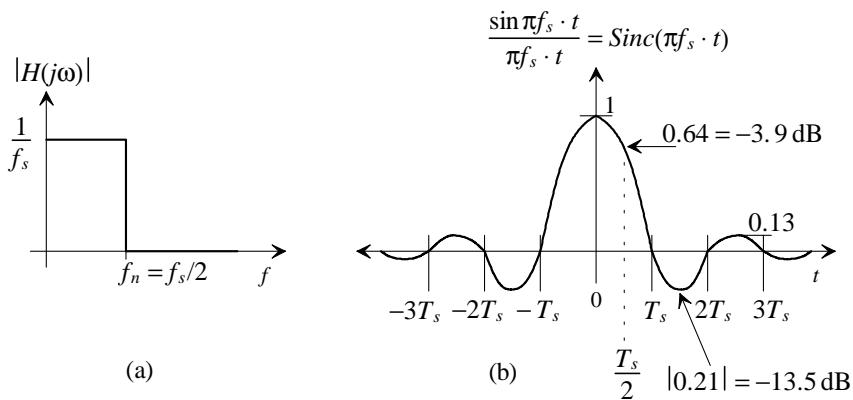
$$h(t) = \int_{-f_n}^{f_n} \frac{1}{f_s} \cdot e^{j2\pi f_n t} \cdot df = \frac{e^{j2\pi f_n t} - e^{-j2\pi f_n t}}{j \cdot 2\pi \cdot f_s \cdot t} = \frac{\sin 2\pi f_n \cdot t}{\pi f_s \cdot t} \quad (30.4a)$$

$$\frac{\sin 2\pi f_n \cdot t}{\pi f_s \cdot t} = \frac{\sin \pi f_s \cdot t}{\pi f_s \cdot t} = Sinc(\pi f_s \cdot t) \quad (30.4b)$$

where

$$\frac{\sin x}{x} \equiv Sinc(x) \quad (30.5)$$

The time-domain response of our ideal RCF is shown in Fig. 30.16. Notice that our impulse is applied to the system's input at $t = 0$ and that the output actually anticipates, or starts, before the application of the input! This indicates that the filter is noncausal and can't be built in a practical analog circuit. Before we discuss the implications of this severe limitation (an ideal reconstruction filter can't actually be built), consider Fig. 30.17. Figure

**Figure 30.16** (a) Ideal RCF frequency response and (b) impulse response (time).

30.17 shows the individual response outputs of an ideal RCF with the impulse train of Fig. 30.14 as the input. The output of the RCF is the weighted sum of the individual responses, of the form $Sinc(x)$, from each of the weighted impulse inputs into the RCF (using superposition). While this figure is "busy," the basic concept of reconstruction should be obvious.

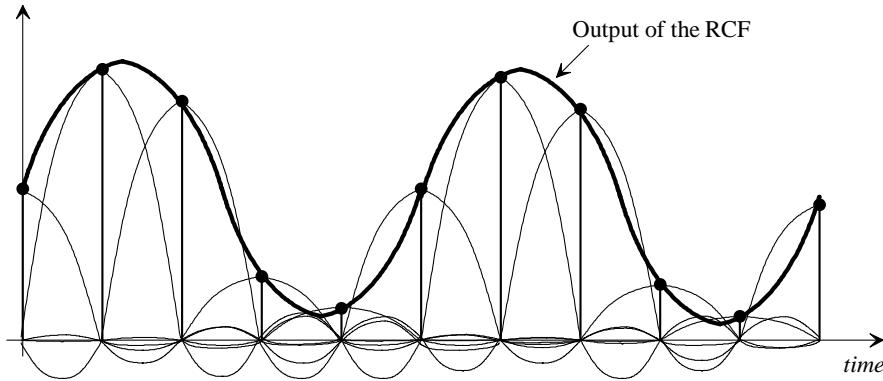


Figure 30.17 Reconstructing the 20 MHz sinewave of Fig. 30.14.

Practically, we can't make an ideal reconstruction filter, which is a requirement for reconstructing a waveform consisting of frequency components between DC (0) and f_n . For example, sampling a 49 MHz sinewave at 100 MHz (essentially two samples per cycle) would require, for reconstruction, a filter with characteristics close to the ideal RCF in order to get a signal resembling the 49 MHz input out of the system. What can we do to ease the requirements on the RCF? Here are two possibilities:

1. Increase the sampling rate, f_s . If we were to sample the 49 MHz sinewave mentioned above at 500 MHz, we would then have roughly ten samples per cycle. The RCF used on the output of the system can then have a slower roll-off rate. At the extreme end, taking $f_s \rightarrow \infty$, eliminates the need for any RCF.
2. After sampling has taken place and using a DSP, add additional points in between the sampling times. This effectively increases the sampling rate coming out of the system. This increase in the effective sampling frequency is known as *interpolation* because the values of the additional points are determined by interpolating between the existing data points. The increase in the effective output sampling rate eases the requirements placed on the RCF.

SPICE Modeling the Impulse Sampler

The SPICE model of the impulse sampling gate was shown in Fig. 30.3. A voltage-controlled switch was used to connect the input signal to the sampler's output for very brief periods of time. In order to make the sampler more ideal, that is, with infinite input

resistance, zero output resistance, etc., the SPICE model shown in Fig. 30.18 will be used. Voltage-controlled voltage sources, with the E prefix, are used to model the ideal operational amplifiers. The switch is modeled with a voltage-controlled switch, an S device. The model is used in the following netlist:

```
* Figure 30.19 CMOS: Mixed-Signal Circuit Design *
.tran .1n 500n 0 .1n UIC
Vin Vin 0 DC 0 Sin 0.75 0.75 5MEG
Vclock Clock 0 DC 0 Pulse 0 1.5 0 0 0 100p 10n
Vtrip Vtrip 0 DC .75
Ebufin Vinb 0 Vin Vinb 100MEG
S1 Vinb Vins CLOCK VTRIP switmod
Rout Vins 0 10k
Ebufout Vout 0 Vins Vout 100MEG
.model switmod SW
.end
```

In this netlist, a 1.5 V, peak-to-peak, 5 MHz sinewave centered at 0.75 V is sampled at 100 MHz. The impulses are generated using the pulse statement with zeroes for both rise and fall times. In the simulation, the actual rise and fall times will be limited by the transient simulation step time (which is set at a maximum of 100 ps using the .tran statement in the netlist above). The impulses have logic amplitude levels of 1.5 V. The switch's trip point is set at 0.75 V so that when the impulse goes above 0.75 V, the switch is closed. Because the impulse is at 1.5 V for only 100 ps, and the input is slow in relation to this time, the netlist approximates an ideal impulse sampling circuit. Running the netlist above results in Fig. 30.19.

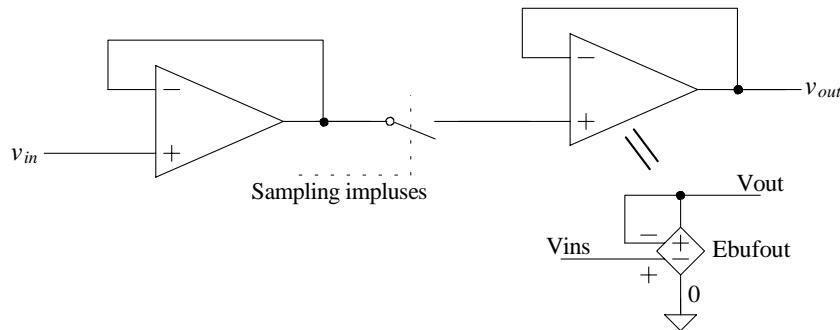


Figure 30.18 SPICE model of the ideal impulse sampler.

Using SPICE for Spectral Analysis (Looking at the Spectrum of a Signal)

Figure 30.19 shows the sampled output of our impulse sampler, in the time domain. It is very useful, as we've already seen, to be able to look at these data in the frequency domain. SPICE (actually Nutmeg) has the feature that it can take the discrete Fourier transform (DFT) of a time domain signal. Performing a DFT consists of (1) windowing the

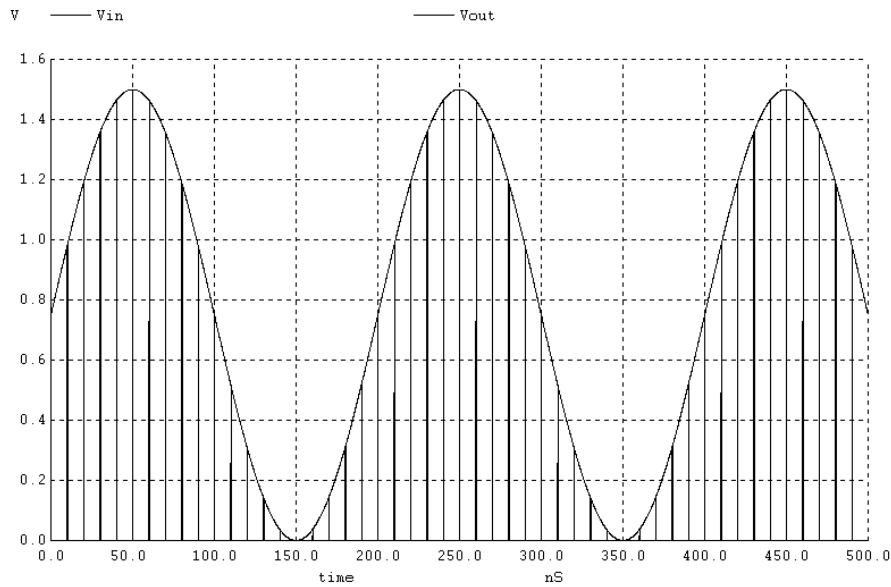


Figure 30.19 Impulse sampling a 5 MHz sinewave at 100 MHz.

time domain signal (we will use the Hanning window [a.k.a. von Hann window] unless otherwise indicated), (2) sampling the signal, and (3) taking the Fourier transform of the signal. Windowing ensures that abrupt transitions do not occur at the beginning and end of the signal to be transformed. It's important to realize that taking the DFT of a signal that has already been sampled results in a spectrum with amplitude errors (more on this in a moment.) To perform a DFT in SPICE we first ensure that the signal to be transformed has a linear time step. To do this we use

```
linearize Vout Vin
```

where Vout and Vin are the signals we are interested in transforming. The *linearize* command with no arguments will linearize all of the variables available in the simulation. Next, we use the *spec* command (spectral analysis command) in SPICE

```
spec 0 200MEG 2MEG Vout Vin
```

This command takes the DFT of Vout and Vin over the range of DC to 200 MHz with a resolution of 2 MHz. The minimum resolution allowed when using the *spec* command (DFT) is set by the transient simulation time, or

$$\text{DFT resolution} \geq \frac{1}{\text{simulation time}} \quad (30.6)$$

If we simulate a circuit for 500 ns, then our minimum resolution is 2 MHz. We can look at the spectrum of a signal using the following list of commands:

```

linearize Vout Vin
spec 0 200MEG 2MEG Vout Vin
* Set noise floor at 120dB by adding 1uV
let voutdb=db(Vout+1e-6)
let vindb=db(Vin+1e-6)
plot voutdb
plot vindb

```

The spectrums, as an example, on the input and output of an impulse sampler sampling a 10 MHz sinewave at 100 MHz are shown in Fig. 30.20. Note that 1 μ V was added to both signals so as to set the noise floor in the display to -120 dB. The DC portion of the input signal is 0.75 V while the peak voltage of the 10 MHz sinewave is also 0.75 V (0.75 V = -2.5 dB). Note that in Fig. 30.20b, because of the double sampling mentioned above, the amplitude of the signals in the output is different than that predicted (see Fig. 30.4). We can *estimate* the baseline reduction, resulting from taking the DFT of an impulse sampled signal (the signal has nonzero values only during the sampling times) using

$$\text{Baseline reduction (or duty cycle)} \approx \frac{2 \cdot (\text{step size})}{T_s} \quad (30.7)$$

For Fig. 30.20b the maximum stepsize specified in the transient simulation was 100 ps (for each cycle of T_s one point is defined as having a total deviation, after being linearized, of approximately 200 ps including rise and fall times, hence the factor of 2 in Eq. (30.7). The baseline, using Eq. (30.7), is 200 ps/10 ns or 0.02 (-34 dB). The 10 MHz signal in Fig. 30.20b is -2.5 dB below the -34 dB baseline. Note how the DC signal is aliased up to the sampling frequency (100 MHz). At DC, with reference to the baseline, the signal amplitude is also -2.5 dB (-36.5 dB). However, when it is aliased up to the 100 MHz (the sampling frequency) it is doubled ($+6$ dB). The doubling comes from adding the images $f_s + 0$ and $f_s - 0$ and results in an amplitude of -36.5 dB + 6 dB or -30.5 dB.

The step size used in a transient simulation, as we saw above, is an important parameter that needs specification when performing a spectral analysis using SPICE. In the netlist that generates Figs. 30.19 and 30.20, we used 100 ps, but made no comment on why this value was selected. Poor selection of the step sizes can give erroneous results if the values are too large or cause the simulation to last a long time if the values are too small. The transient simulation characteristics in a SPICE netlist are specified using

```
.tran print-step stop-time delay-time maximum-stepsizes <UIC>
```

The step size, for a general simulation with nonideal components, can be set using

$$\text{step size} = 1\% \cdot T_s \text{ or with ideal components } 10\% \cdot T_s \quad (30.8)$$

If our sample frequency, f_s , is 100 MHz then we would set our step size to 100 ps using

```
.tran 100p 2000u 0 100p UIC
```

The term UIC forces the simulation to start with initial conditions (use initial conditions), such as an initial voltage across a capacitor. The simulation always starts at zero time. However, specifying a delay time in the simulation will make SPICE start *saving* data at

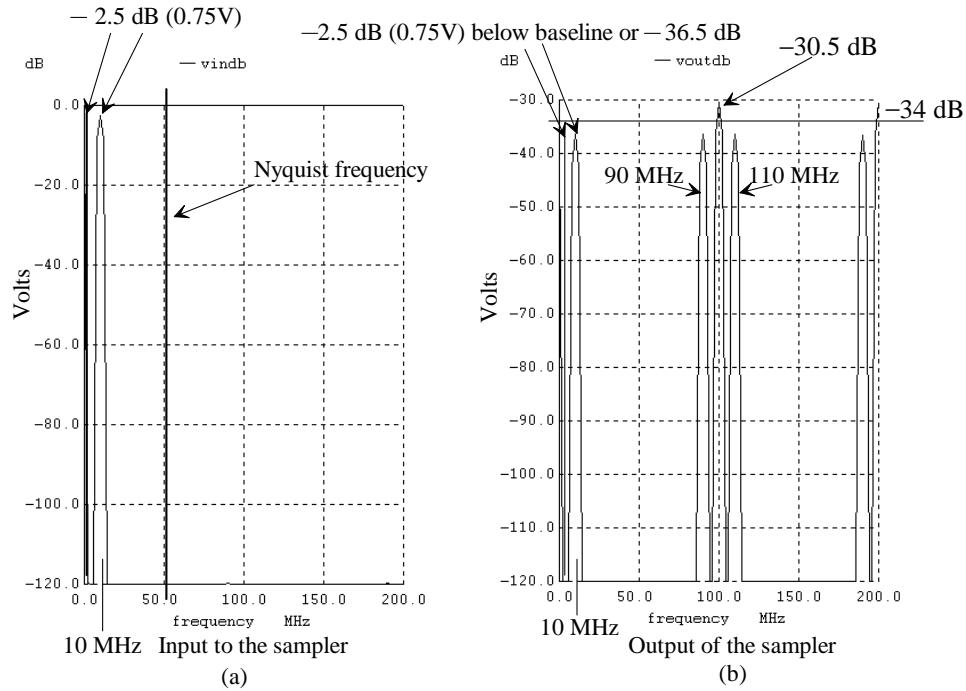


Figure 30.20 Spectrums of the signals shown in Fig. 30.19.

the time specified by the delay-time parameter. This parameter is useful in removing, from a spectral response, for example, the start-up transients in a simulation or keeping the size of a raw output file from getting too large.

Representing the Impulse Sampler's Output in the z-Domain

Consider the output of an impulse sampler, $y(t)$, with an input of $x(t)$ shown in Fig. 30.21. The sampler output can be written as

$$y(t) = x(t) \cdot \sum_{k=-\infty}^{\infty} \delta_u(t - kT_s) \quad (30.9)$$

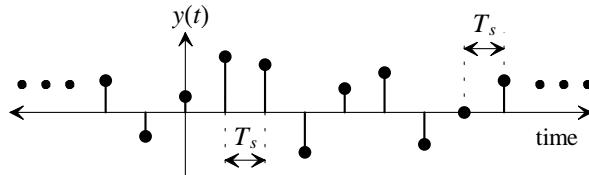


Figure 30.21 Output of an impulse sampler.

We can rewrite Eq. (30.9) as

$$y(t) = x(t)[\dots + \delta_u(t+T_s) + \delta_u(0) + \delta_u(t-T_s) + \delta_u(t-2T_s) + \delta_u(t-3T_s) + \dots] \quad (30.10)$$

Taking the Fourier transform of this equation results in

$$Y(f) = \dots + x(-1)e^{(1)j2\pi f T_s} + x(0)e^{(0)j2\pi f T_s} + x(1)e^{(-1)j2\pi f T_s} + x(2)e^{(-2)j2\pi f T_s} + \dots \quad (30.11)$$

where the term $e^{j2\pi f T_s}$ corresponds to a phase shift of $2\pi \cdot f \cdot T_s$ (radians) when the output of the sampler, $Y(f)$, is evaluated at the frequency f . In other words, each consecutive sample coming out of the impulse sampler is shifted in the time domain by T_s (which is simply saying, in words, what Fig. 30.21 shows). If we define

$$z \equiv e^{j2\pi f T_s} = e^{j2\pi \frac{f}{f_s}} \quad (30.12)$$

then the output of our sampler can be written as

$$Y(z) = \dots + x(-1)z^1 + x(0)z^0 + x(1)z^{-1} + x(2)z^{-2} + \dots \quad (30.13)$$

or

$$Y(z) = \sum_{k=-\infty}^{\infty} x(k) \cdot z^{-k} \quad (30.14)$$

Example 30.5

Determine the output of an impulse sampler in the z-domain if its input, $x(t)$, is a unit step. What is the sampler's impulse response $H(z)$ [$= Y(z)/X(z)$]?

The unit step, $u(t)$, is defined by

$$\begin{aligned} u(t) &= 1 \text{ for } t \geq 0 \\ u(t) &= 0 \text{ for } t < 0 \end{aligned}$$

The time domain output of the sampler, with $u(t)$ as an input, is given by

$$y(t) = \sum_{k=0}^{\infty} u(t-kT_s)$$

or looking at the output in the z-domain

$$Y(z) = \sum_{k=0}^{\infty} z^{-k}$$

The time domain signals used in this example are shown in Fig. 30.22. The sampler's impulse response is simply unity since the z-transform of the input (assuming the input was passed through an ideal impulse sampler so that we can take the z-transform of the signal) and output of the sampler are identical, that is, $H(z) = 1$. ■

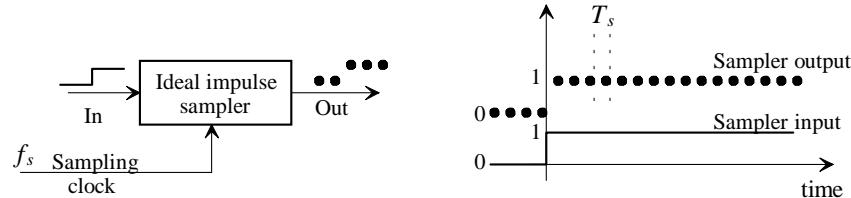


Figure 30.22 Sampling the unit step with an impulse sampler.

Example 30.6

What is the effect of multiplying $H(z)$, in Ex. 30.5 (or any z-domain transfer function), by z^{-1} ?

Multiplying any z-domain transfer function by z^{-1} is equivalent to shifting the system's output later in time by T_s . The result of changing the ideal sampler's transfer function from unity to z^{-1} is shown in Fig. 30.23. Multiplying by z^{-L} shifts the output of the system later in time by $L \cdot T_s$ ■

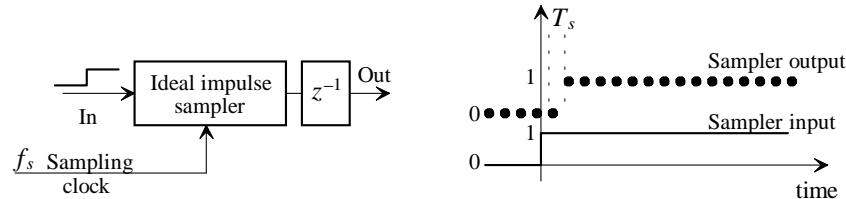


Figure 30.23 Multiplying the output of the ideal sampler by z^{-1} .

An Important Note

It's important to note that our impulse sampler quantizes¹ the input signal in *time* but not amplitude (unlike an analog-to-digital converter which quantizes the input in both time and amplitude). The amplitude out of the ideal impulse sampler is exactly the same as the amplitude input to sampler at the sampling impulse time. We'll find that the z-transform can be used to describe systems using both quantization in time as well as in amplitude. In other words, whether we are discussing digital words, in a binary format or sampled-analog waveforms with amplitudes of volts, amps, or coulombs, we can use the z-transform to represent the discrete-time systems that process the signals.

¹ Quantize: to limit the possible values of a quantity to a discrete set of values. Quantizing in time, for example, means that the output amplitude is only defined at certain discrete times (such as the sampling impulse times for the ideal impulse sampler) or that the amplitude is unchanging during certain discrete time intervals (such as seen in the output of the ideal sample-and-hold discussed in the next section).

30.1.2 The Sample and Hold

Understanding the operation of the impulse sampler of Sec. 30.1.1 is important in understanding the concepts of aliasing and reconstruction. However, as seen in Fig. 30.2, most mixed-signal systems employ a sample and hold (S/H) rather than an impulse sampler so that the sampled waveform is available at times other than the sampling impulse times. Having the samples "held" in between the sampling impulse times is important for proper ADC operation. The disadvantage of using the S/H, as we shall shortly see, is that it will introduce distortion into our signal.

SPICE Modeling the Sample and Hold

The block diagram of the SPICE model, for the ideal S/H, is shown in Fig. 30.24. The inputs are *Clock* and V_{in} , while the output is labeled V_{out} . A SPICE netlist, using the ideal S/H, is shown below in which an 8 MHz sinewave is sampled at 100 MHz. The simulation results, using this netlist, are shown in Fig. 30.25.

```
* Figure 30.25 CMOS: Mixed-Signal Circuit Design *
.tran .1n 500n .1n UIC
Vin Vin 0 DC 0 Sin 0.75 0.75 8MEG
Vclock Clock 0 DC 0 Pulse 0 1.5 0 0 0 4.9n 10n
Vtrip Vtrip 0 DC .75
VDD VDD 0 DC 1.5
Ein Vinbuf 0 Vin Vinbuf 100MEG

S1 Vinbuf VinS VTRIP CLOCK switmod
Cs1 VinS 0 1e-10
S2 VinS Vout1 CLOCK VTRIP switmod
Cout1 Vout1 0 1e-16

Eout Vout 0 Vout1 Vout 100MEG
.model switmod SW
.end
```

The switches S1 and S2 in the netlist above sample the input using the input clock. Note that both switches can be closed, momentarily in Fig. 30.24, at the same time. The time

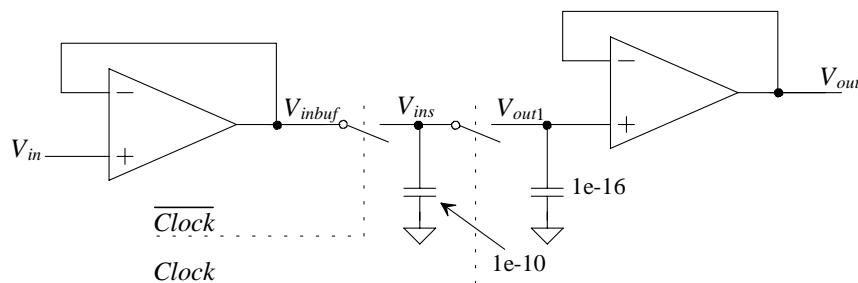


Figure 30.24 SPICE model of the ideal sample and hold (S/H).

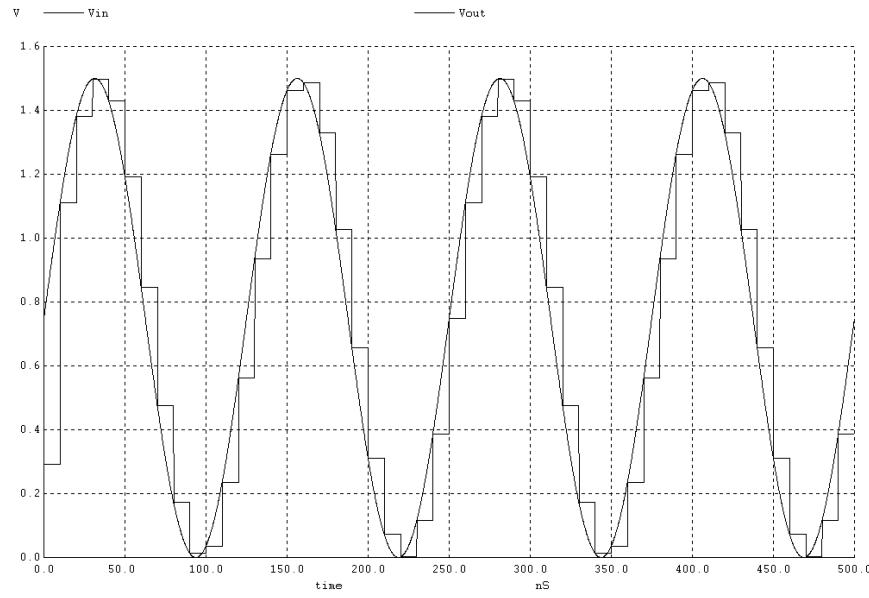


Figure 30.25 Ideal S/H with an 8 MHz input.

that the switches are closed is approximately equal to the transient step time. The charge sharing between the capacitors is affected by having both switches closed at the same time. Values given in this figure were selected so that a million-to-one ratio existed between the two capacitors (120 dB range.) Because both switches are closed at the same time, the difference between the two capacitors can be made smaller without affecting the circuit's operation. Also note that over a time set by GMIN (remember a resistor with a value of $1/GMIN$ is placed across every pn-junction in a SPICE simulation and GMIN's default value is $1e-12$ or $1\text{G}\Omega$) and the capacitor values the charge on the capacitors will leak off causing droop. For the 0.1f capacitor the associated RC time because of GMIN is $100\ \mu\text{s}$ (increasing this capacitor to 10f won't affect the sampling operation and pushes the RC time up to 10 ms).

The accuracy of the S/H is ultimately limited by the tolerances, that is, RELTOL, ABSTOL, and VNTOL of the simulation. For an accurate simulation we may add

```
.options RELTOL=1u VNTOL=1u ABSTOL=1p
```

to the netlist. The accuracy of a simulation will be discussed in greater detail later.

S/H Spectral Response

Consider the application of a sinewave, at a frequency f_{in} , to the ideal S/H shown in Fig. 30.26. To make the discussion as general as possible assume that the output of the S/H can return-to-zero (RZ) as shown in Fig. 30.27 (which shows coarse time quantization for

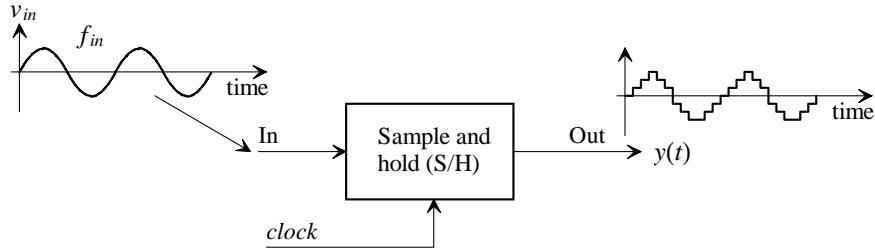


Figure 30.26 Sampling and holding an input sinewave.

a simpler figure and illustration of the concept of RZ). Note that as T approaches T_s we get the operation of the S/H in Fig. 30.26. The output of the ideal S/H is given by

$$y(t) = \sum_{n=-\infty}^{\infty} \left(\underbrace{V_p \sin(2\pi f_{in} \cdot nT_s)}_{v_{in}(t)} \cdot \underbrace{h(t)}_{[u(t-nT) - u(t-(n+1)T)]} \right) \quad (30.15)$$

Note that the *sine* term is only defined at discrete sampling instances so that its spectrum is given by Eq. (30.2). The spectrum of the sampling pulse, $|H(f)|$, because of the duality of the Fourier transform, is given by reviewing Fig. 30.16 or calculated using

$$\text{Fourier}[u(t-nT) - u(t-(n+1)T)] = \int_0^{T_s} [u(t-nT) - u(t-(n+1)T)] e^{-j2\pi f t} \cdot dt \quad (30.16)$$

which is evaluated as

$$H(f) = \frac{e^{-j2\pi f T} - 1}{-j \cdot 2\pi \cdot f} = e^{-j\pi f T} \cdot \frac{e^{j\pi f T} - e^{-j\pi f T}}{j \cdot 2\pi \cdot f} = \underbrace{e^{-j\pi f T}}_{\text{phase}} \cdot \underbrace{\frac{1}{T} \cdot \text{Sinc}(\pi \cdot f \cdot T)}_{\text{magnitude}} \quad (30.17)$$

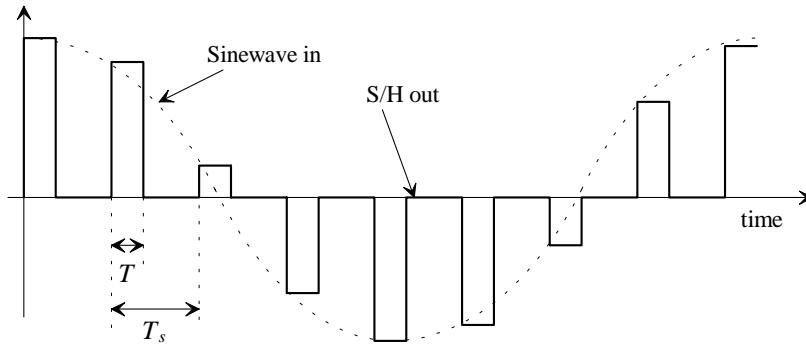


Figure 30.27 Sample-and-hold output with return to zero format.

The magnitude of Eq. (30.17), $|H(f)|$, is plotted in Fig. 30.28. The phase response corresponds to a shift in time of $T/2$ so, to simplify the math below, we will only concern ourselves with the magnitude response of $H(f)$.

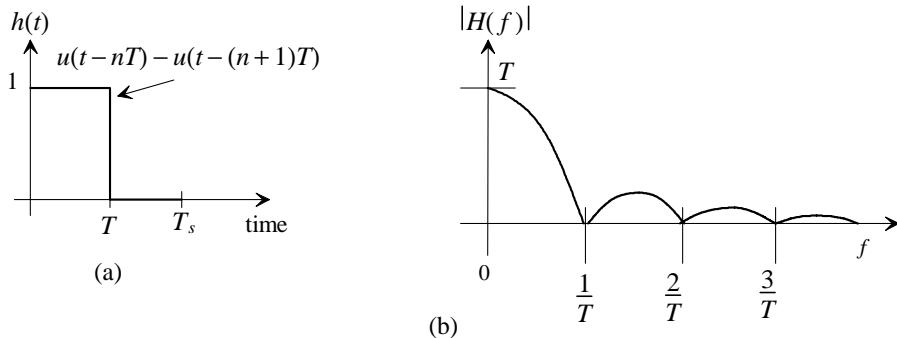


Figure 30.28 (a) Sampling pulse and (b) its spectrum.

Multiplication in the time domain can be evaluated using convolution in the frequency domain. The frequency spectrum of a sinewave, sampled with an ideal S/H, is given by

$$Y(f) = H(f) * V_{in}(f) = \int_{-\infty}^{\infty} H(L) \cdot V_{in}(f-L) dL \quad (30.18)$$

or

$$|Y(f)| = \underbrace{T \cdot |Sinc(\pi \cdot T \cdot f)|}_{\text{Weighting from S/H, } |H(f)|} \cdot \underbrace{\left[\frac{V_p}{T_s} \cdot \sum_{k=-\infty}^{\infty} [\delta(f - f_{in} + kf_s) + \delta(f + f_{in} + kf_s)] \right]}_{\text{Ideal impulse sampler response}} \quad (30.19)$$

As $T \rightarrow 0$ ($h(t) \rightarrow \delta_u(t)$), the frequency response of the sample-and-hold approaches the ideal impulse sampler of Sec. 30.1.1. Also, note that using an RZ format (making $T < T_s$) can reduce the amount of attenuation introduced by the S/H ($|H(f)|$ doesn't roll off as fast.)

For most circuit designs, $T = T_s$ so that, as Eq. (30.19) shows, the sample-and-hold operation weights the amplitude of the ideal impulse sampler's frequency response by $Sinc\left(\frac{\pi f}{f_s}\right)$ or $Sinc\left(\frac{\pi f}{2f_n}\right)$. Note that at the sampling frequency ($f_s = 1/T_s$) the output of the ideal S/H goes to zero. Let's illustrate the frequency response of an ideal S/H using an example.

Example 30.7

Using the ideal S/H SPICE model show and discuss the spectrum resulting from sampling a 3 MHz sinewave at 100 Msamples/s.

The results of passing a 0.75 V (peak) sinewave centered at 0.75 V (-2.5 dB) through the ideal S/H are shown in Fig. 30.29. We have also plotted the response of the S/H, $|H(f)|$ in this figure. The attenuation the 97 MHz image sees is

$$\text{Attenuation} = \text{Sinc}\left(\frac{\pi \cdot 97}{100}\right) = 0.031 = -30.2 \text{ dB}$$

The amplitude of the 97 MHz image is -2.5 dB below the attenuation resulting from using a S/H or -32.7 dB. Note how at the Nyquist frequency of 50 MHz, the signal is attenuated by -3.9 dB. Also note how the DC image at f_s is attenuated by the S/H instead of being doubled as in the impulse sampler (Fig. 30.20).

Two additional notes: First, the S/H cannot be used as an AAF since any aliasing that occurred using the impulse sampler still occurs using the S/H. For example, sampling a 60 MHz sinewave at 100 MHz still results in a 40 MHz alias signal in the base spectrum (the spectrum from DC to f_n), as shown in Fig. 30.5. Now, however, the signal is attenuated by the S/H (the attenuation is -2.4 dB at 40 MHz when sampling at 100 Msamples/s.) In other words, the S/H can be thought of as an ideal impulse sampler followed by a *Sinc* response filter. Second,

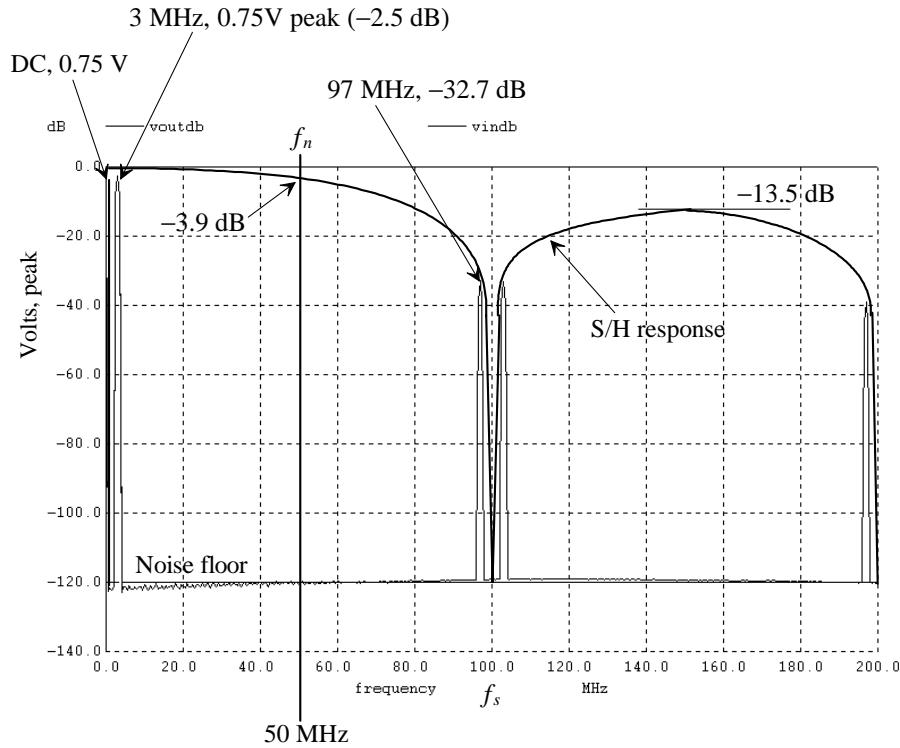


Figure 30.29 Output of a S/H after sampling a 3 MHz sinewave at 100 MHz.

repetitively sampling and holding a signal results in only one S/H attenuation hit (assuming the timing is such that a sampling operation is not occurring when the previous S/H stage's output is changing). This means that topologies that use several S/H operations on an input signal, such as a pipeline ADC, only attenuate the signal by $Sinc(\pi f/f_s)$ once. ■

The output of the S/H (assuming $T = T_s$) should be passed through a two-stage reconstruction filter, to recover the input signal. One of the stages will have the frequency response of the ideal RCF of Fig. 30.16. The other stage will have a frequency response given by

$$H_{RCFSH}(f) = \frac{1}{Sinc\left(\frac{\pi f}{f_s}\right)} = \frac{\pi f}{2f_n \sin\left(\frac{\pi f}{2f_n}\right)} \quad (30.20)$$

to compensate for the attenuation of the S/H Sinc response. The shape of the ideal reconstruction filter is shown in Fig. 30.30. Again, increasing the sampling frequency relative to the input frequency will ease the requirements placed on the reconstruction filter. Note how using the RZ format modifies the requirements placed on the reconstruction filter to the point, when using impulse sampling, of having the brick wall ideal RCF of Fig. 30.7.

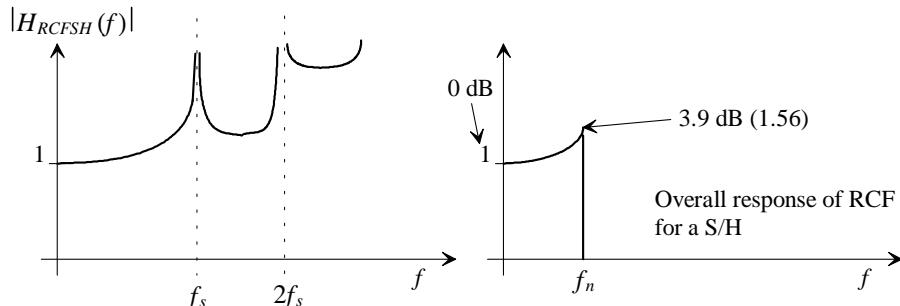


Figure 30.30 Ideal reconstruction filter response for a S/H.

Before we leave this section, let's answer the question: "What sets the value of the noise floor in SPICE (Fig. 30.29)?". We can *limit* the noise floor, in Fig. 30.29 for example, by adding 1 μ V to voltages in the circuit. However, the SPICE-simulated spectrum's noise floor, which is set by simulation variations, is limited by the RELTOL parameter. Also, the length of the simulation can be important. ABSTOL, which defaults to 1 pA, and VNTOL, which defaults to 1 μ V, signify when a current or voltage has converged in a SPICE simulation. If the step change in the simulation, for all currents and voltages at a given time, is within ABSTOL (for currents) or VNTOL (for voltages), then SPICE moves on to the next step in time (for a transient simulation). The parameter RELTOL was added to SPICE so that simulations involving large currents and voltages

were not forced to use ABSTOL and VNTOL to signify convergence. In other words, if a current is approximately 10 A, we won't force the SPICE number for the current to be 10.000000000001. Instead we use 10.01 (the product of RELTOL [assuming = 0.001] and 10 A) to signify convergence. To signify that a current has converged, we use the larger of

$$\text{ABSTOL or RELTOL} \cdot I_{\text{simulated}} \quad (30.21)$$

while for a voltage we use the larger of

$$\text{VNTOL or RELTOL} \cdot V_{\text{simulated}} \quad (30.22)$$

For the simulation shown in Fig. 30.29, we set RELTOL to 10^{-6} so that our 1 V level signals simulate to within 1 μ V of their "actual" values. This keeps the simulation noise from setting our noise floor. The practical problem of reducing RELTOL is convergence when nonideal components (e.g. MOSFETs) are added to the simulation. Trade-offs must be made between simulation noise and convergence when using both ideal and nonideal components in a simulation.

Circuit Concerns for Implementing the S/H

Figure 30.31 shows a single-ended input and output S/H implementation using either an op-amp or an OTA (operational transconductance amplifier). At the time t_0 , the ϕ_1 and ϕ_2 switches are closed while the ϕ_3 switches are open. During the time between t_1 and t_2 the input charges the hold capacitor C_H . The input is connected to the left side, or bottom plate (the plate closest to the substrate), of C_H , while because of the op-amp, the right side (top plate) is connected to ground (or a common mode voltage, V_{CM}). At t_1 the ϕ_1 switch opens and for a very short time (set by $t_3 - t_1$) the op-amp operates open loop (no feedback). As the top plate is always at ground (or V_{CM}) at t_1 , the charge injection and capacitive feedthrough resulting from the ϕ_3 switches turning off are independent of the input signal. When the ϕ_2 switch turns off, the charge injection will, ideally, flow into the low-impedance input, v_{in} , since the impedance looking into the right of the ϕ_2 switch is large. This, again ideally, leaves the voltage across the hold capacitor unaffected by the charge injection resulting from turning off the switches. This sequence of turning off the

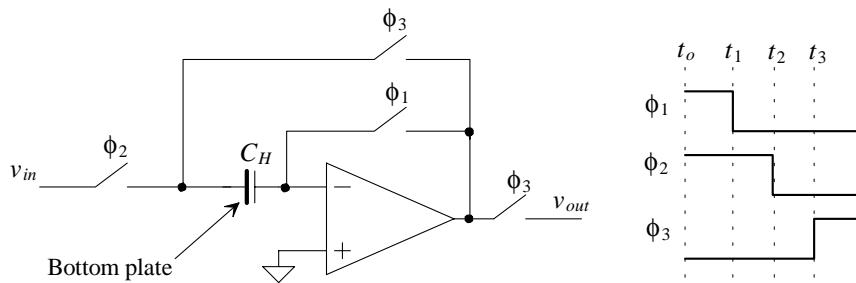


Figure 30.31 Single-ended S/H operation.

switch to the right of C_H followed by turning off the switch connecting v_{in} to C_H is often, confusingly, called *bottom plate sampling*. Bottom plate sampling is illustrated in its simplest form in Fig. 30.32. In this figure the switch connected to the bottom plate of the capacitor is turned off first. When the ϕ_2 switch turns off, the charge can be injected into the low-impedance node, the input v_{in} , or into the series combination of C_H and the off switch. The charge takes the lowest impedance path to ground and thus most of the charge injection resulting from the ϕ_2 switch turning off flows through v_{in} , leaving the voltage across the hold capacitor unaffected. We should see why the name "bottom plate sampling" is confusing. Reviewing Fig. 30.31, we see that the top plate of the hold capacitor is connected to the ϕ_1 switch while, in Fig. 30.32, the bottom plate of the hold capacitor is connected to the ϕ_1 switch.

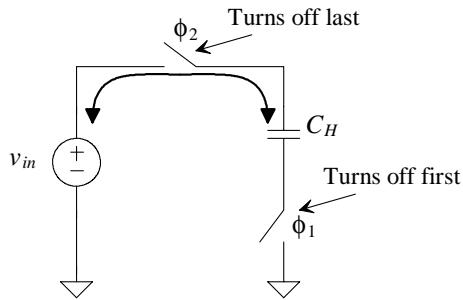


Figure 30.32 Bottom plate sampling.

Returning to the discussion of the operation of the S/H of Fig. 30.31 we see that at t_3 the ϕ_3 switches turn on and the op-amp behaves as a voltage follower holding the sampled input voltage. The sampling instant occurs between t_1 and t_3 (which should be short to keep the op-amp output from drifting toward VDD or ground.)

30.2 SPICE Models for DACs and ADCs

In this section we develop SPICE models for ideal digital-to-analog converters (DACs) and analog-to-digital converters (ADCs). Our goal is to have SPICE code, or subcircuits, that we can place in a mixed-signal simulation to either (1) generate a digital word based on an analog input (using the ideal ADC) or (2) look at the spectrum of a digital signal (using the ideal DAC and the spectral analysis capability in SPICE [using the discrete Fourier transform].)

30.2.1 The Ideal DAC

While there are an infinite number of ways to implement an ideal DAC in SPICE, we use a method that results in a computationally efficient model for a DAC. Before we discuss the implementation, let's review some fundamental characteristics of a DAC.

Consider the ideal transfer characteristics of a 3-bit DAC shown in Fig. 30.33. (For a detailed review of general DAC characteristics, see Ch. 28.) Notice in this figure that we have drawn two reference voltages, V_{REF+} and V_{REF-} , and are assuming that $V_{REF+} > V_{REF-}$. When a digital input of 000 is applied to the DAC, the output voltage becomes V_{REF-} . When the input code is increased to 001, the output of the DAC (an analog voltage defined at discrete amplitude levels) increases by one least significant bit (LSB). If the DAC has an input code with a number of bits, N , then we can define an LSB as

$$1 \text{ LSB} = \frac{V_{REF+} - V_{REF-}}{2^N} = V_{LSB} \quad (30.23)$$

If, for example, $V_{REF+} = 1.25$ V and $V_{REF-} = 0.25$ V and $N = 3$, then our LSB, the vertical distance between adjacent points in Fig. 30.33, is 0.125 V. Note that in our discussion of

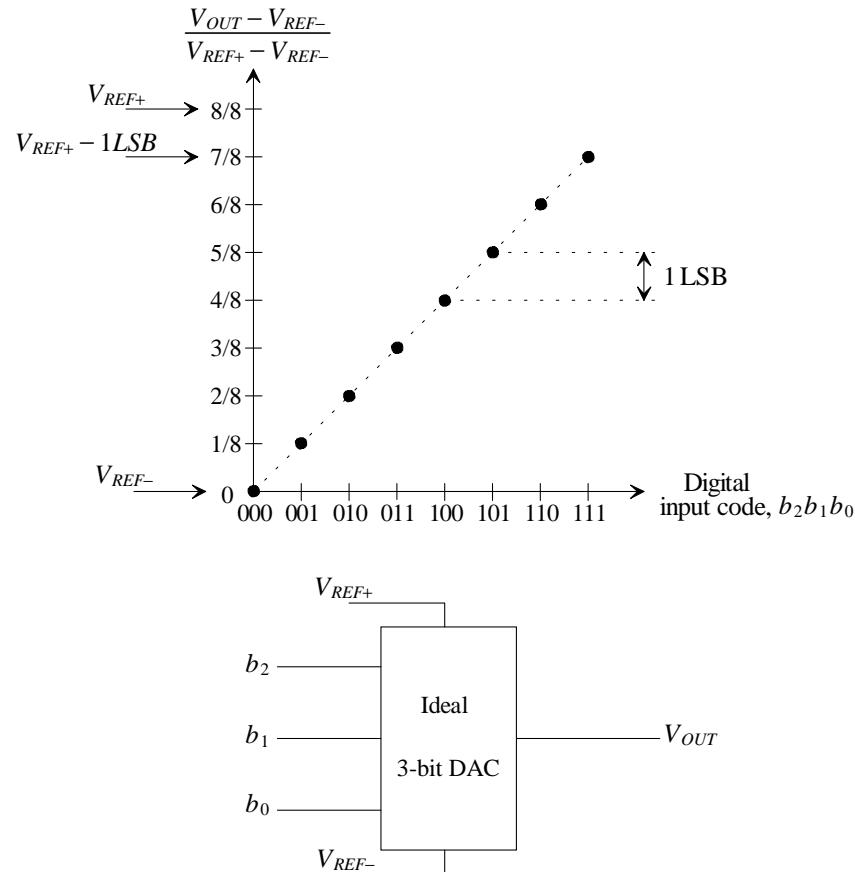


Figure 30.33 An ideal 3-bit DAC.

an ideal DAC we are assuming that the output of the DAC ranges from V_{REF-} up to $V_{REF+} - 1$ LSB. We could just as easily have assumed that the output ranged from $V_{REF-} + 1$ LSB up to V_{REF+} . The important thing to notice is that the DAC output range is 1 LSB smaller than the difference between the positive and negative reference voltages. For the DAC developed in this chapter, we will assume $V_{REF+} = VDD = 1.5$ V and $V_{REF-} = VSS = 0$ V. In Ch. 33 we discuss a submicron CMOS process using these power supply voltages, 1.5 V and 0 V. Selection of the power supply rails, which are noise free in a SPICE simulation, allow the maximum output range for the DAC (assuming the reference voltages are indeed the maximum and minimum voltages in the system, i.e., no charge pumps or external, larger, power supply voltages). If we need more resolution when using our ideal DAC, we will simply increase the number of bits, N , used and hence decrease the value of the DAC's LSB.

SPICE Modeling Approach

We can write the output of the ideal DAC in terms of the reference voltages and digital input codes b_N (which are logic "0" or "1"), and assuming that an input code of all zeroes results in an output voltage of V_{REF-} , as

$$V_{OUT} = (V_{REF+} - V_{REF-}) \cdot \left(\frac{b_{N-1}}{2^1} + \frac{b_{N-2}}{2^2} + \dots + \frac{b_1}{2^{N-1}} + \frac{b_0}{2^N} \right) + V_{REF-} \quad (30.24)$$

or

$$V_{OUT} = (V_{REF+} - V_{REF-}) \cdot \frac{1}{2^N} \cdot (b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \dots + b_1 \cdot 2^1 + b_0) + V_{REF-} \quad (30.25)$$

We can implement this equation, in SPICE, using a nonlinear dependent source (a B source). For a 3-bit, ideal DAC, the statement that implements this equation may look like

*Nonlinear dependent source, B, for generating the DAC output
 Bout Vout 0 V=((v(vrefp)-v(vrefm))/8)*(v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)

The terms BXL correspond to logic signals that have a value of 1 V or 0 V.

Example 30.8

Write the nonlinear dependent source statement for an ideal 12-bit DAC.

The statement follows:

```
Bout Vout 0 V=((v(vrefp)-v(vrefm))/4096)*  

+(v(B11L)*2048)+v(B10L)*1024+v(B9L)*512+v(B8L)*256  

+v(B7L)*128+v(B6L)*64+v(B5L)*32+v(B4L)*16+v(B3L)*8+  

+v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)
```

remembering that a "+" in the first column of a line indicates that the text on the remainder of the line behaves as if it were typed at the end of the previous line. It doesn't indicate addition. ■

The next thing we need to concern ourselves with is the digital logic levels. We want to use our ideal DAC with nonideal (real) circuits where the logic voltage levels may not be well defined. We need to determine and use a switching-point voltage based on the

power-supply voltage VDD . We will assume the input logic code is a valid logic "1" if its amplitude is greater than $VDD/2$ and a logic "0" if its amplitude is less than $VDD/2$. We can implement the $VDD/2$ switching point, or trip voltage, using the following SPICE lines

```
*Generate Logic switching point, or trip, voltage
R1 VDD trip 100MEG
R2 trip 0 100MEG
```

The solid logic levels can be generated using the following subcircuit SPICE code. The switch implementation is shown in Fig. 30.34.

```
.subckt Bitlogic trip BX BXL
Vone one 0 DC 1
SH one BXL BX trip Switmod
SL 0 BXL trip BX Switmod
.model switmod SW
.ends
```

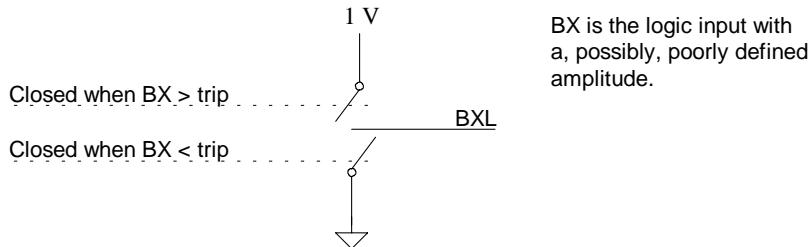


Figure 30.34 Generating logic levels using voltage-controlled switches.

Using the above code, the subcircuit definition for an ideal 8-bit DAC can be written, as shown in Fig. 30.35. Using this subcircuit in the following netlist, we can show the operation of an ideal 8-bit DAC:

```
VDD VDD 0 DC 1.5
VREFP VREFP 0 DC 1.5
VREFM VREFM 0 DC 0.0

VB7 B7 0 DC 0 pulse 1.5 0 0 200p 200p 1279.8n 2560n
VB6 B6 0 DC 0 pulse 1.5 0 0 200p 200p 639.8n 1280n
VB5 B5 0 DC 0 pulse 1.5 0 0 200p 200p 319.8n 640n
VB4 B4 0 DC 0 pulse 1.5 0 0 200p 200p 159.8n 320n
VB3 B3 0 DC 0 pulse 1.5 0 0 200p 200p 79.8n 160n
VB2 B2 0 DC 0 pulse 1.5 0 0 200p 200p 39.8n 80n
VB1 B1 0 DC 0 pulse 1.5 0 0 200p 200p 19.8n 40n
VBO B0 0 DC 0 pulse 1.5 0 0 200p 200p 9.8n 20n
```

```
X1 VDD VREFP VREFM Vout B7 B6 B5 B4 B3 B2 B1 B0 DAC8bit
```

```

*** Start Ideal 8-bit DAC Subcircuit ****
*.subckt DAC8bit VDD VREFP VREFM Vout B7 B6 B5 B4 B3 B2 B1 B0
*Generate Logic switching point, or trip, voltage
R1 VDD trip 100MEG
R2 trip 0 100MEG

*Change input logic signals into logic 0s or 1s
X7 trip B7 B7L Bitlogic
X6 trip B6 B6L Bitlogic
X5 trip B5 B5L Bitlogic
X4 trip B4 B4L Bitlogic
X3 trip B3 B3L Bitlogic
X2 trip B2 B2L Bitlogic
X1 trip B1 B1L Bitlogic
X0 trip B0 B0L Bitlogic

*Nonlinear dependent source, B, for generating the DAC output
Bout Vout 0 V=((v(vrefp)-v(vrefm))/256)*(v(B7L)*128+v(B6L)*64+
+v(B5L)*32+v(B4L)*16+v(B3L)*8+v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)

.ends

.subckt Bitlogic trip BX BXL
Vone one 0 DC 1
SH one BXL BX trip Switmod
SL 0 BXL trip BX Switmod
.model switmod SW
.ends

*** END DAC Subcircuit ****

```

Figure 30.35 SPICE subcircuit netlist for an ideal 8-bit DAC.

In this netlist we are assuming $V_{REF+} = 1.5$ V and $V_{REF-} = 0$. The pulse sources step the DAC through all possible codes, i.e. from 00000000 (= 0 V) all the way up to 11111111 (= 1.5 V – 1 LSB) in increments of 1.5/256 or 5.859 mV (= 1 LSB.) The simulation results are shown in Fig. 30.36. It should be very easy to see how to implement any resolution of ideal DAC at this point using SPICE.

Before leaving the ideal DAC let's discuss how to shift the ideal output characteristics up by 1 LSB. The DAC in Fig. 30.35 has an output range of 0 V (V_{REF-}) to $VDD - 1$ LSB ($V_{REF+} - 1$ LSB). We can rewrite Eq. (30.25) as

$$V_{OUT} = \underbrace{(V_{REF+} - V_{REF-})}_{1 \text{ LSB}} \cdot \frac{1}{2^N} \cdot (b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \dots + b_1 \cdot 2^1 + b_0) + V_{REF-} \quad (30.26)$$

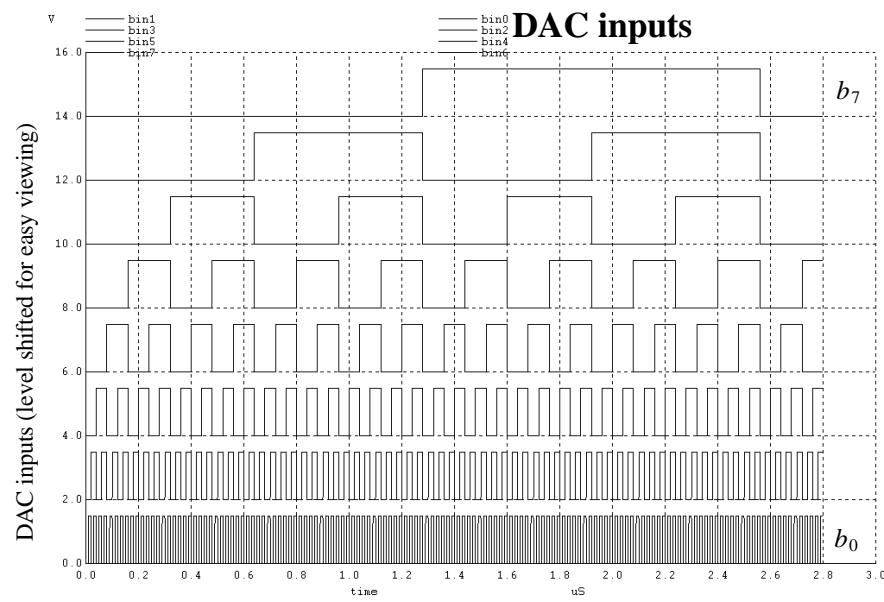
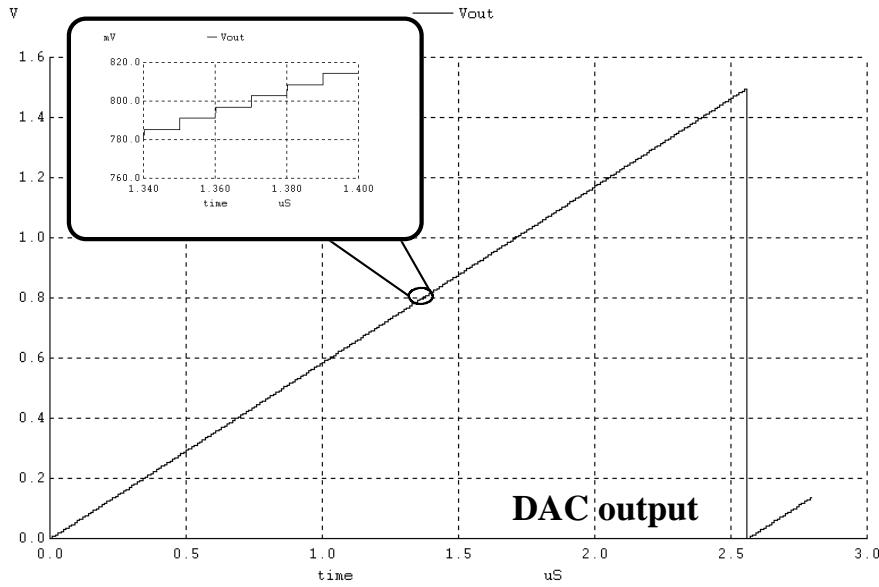


Figure 30.36 Simulating an ideal 8-bit DAC.

To shift the output up by 1 LSB (so the output of the ideal DAC ranges from 1 LSB above V_{REF-} to V_{REF+}) we simply add one to the binary-weighted term in the parentheses

$$V_{OUT} = \overbrace{(V_{REF+} - V_{REF-}) \cdot \frac{1}{2^N} \cdot (b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \dots + b_1 \cdot 2^1 + b_0 + 1)}^{1LSB} + V_{REF-} \quad (30.27)$$

This equation is trivial to implement in our ideal DAC by adding two characters to our nonlinear dependent source, i.e., "+1."

30.2.2 The Ideal ADC

The characteristics of our ADC are shown in Fig. 30.37. (Again, a complete discussion of ADC characteristics was given in Ch. 28.) Notice how in this figure the transfer curve is shifted to the left. If we were to flip the curve on its side and mark, with black dots, the intersection of the analog input voltage with the ADC transfer curve, we would have the DAC transfer curve of Fig. 30.33. Again 1 LSB is given by Eq. (30.23). Notice how converting a (normalized) input voltage of 0.1 V will result in an output code of 000 which is the same output code resulting from converting 0 V. Unlike the ideal DAC, the ideal ADC quantizes its input with the practical result of adding noise to the input signal. This noise is often called *quantization noise*.

The implementation of the ideal ADC consists of an ideal S/H followed by passing the output of the S/H (the held signal) through an algorithm to generate the output bits. The algorithm we use is based on a pipeline ADC and follows:

1. The input signal is sampled and held.
2. This held signal is input to a comparator that compares the input value to a reference voltage.
3. If the input signal is greater than the reference voltage, the output bit is set to a high, and the reference signal is subtracted from the input. The difference is multiplied by two and passed to the output of stage.
4. If the input signal is less than the reference voltage, the output bit is set low. The input signal is multiplied by two and passed to the output of the stage.
5. This output is used as the input to the next stage and steps 2, 3, and 4 above are repeated. This continues for N stages (where N is the number of bits in the ADC).

The reference voltage, or common mode voltage V_{CM} , can be determined by calculating the midpoint between V_{REF+} and V_{REF-} followed by subtracting V_{REF-} so that the V_{CM} is referenced to 0 V. This can be written as

$$V_{CM} = \frac{V_{REF+} + V_{REF-}}{2} \rightarrow V_{CM0} = \frac{V_{REF+} + V_{REF-}}{2} - V_{REF-} = \frac{V_{REF+} - V_{REF-}}{2} \quad (30.28)$$

We also want to level shift the input signal so that it is referenced to 0 V. In addition, we want to shift the transfer curves to the left by 1/2 LSB as seen in Fig. 30.37. To do this we

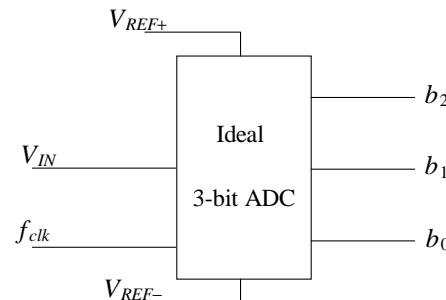
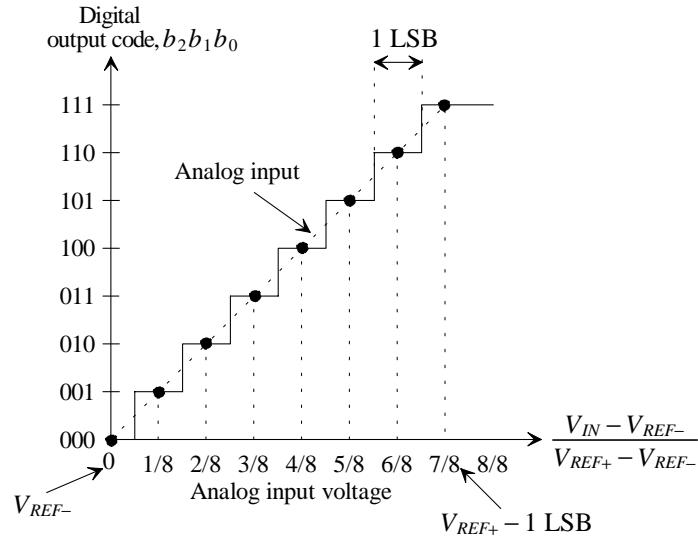


Figure 30.37 An ideal 3-bit ADC.

use the following SPICE statement (for an 8-bit ADC where V(OUTSH) is the output voltage of the ideal S/H [the input to the pipeline algorithm above])

* Level shift by VREFM and 1/2LSB
 BPIP PIPIN 0 V=V(OUTSH)-V(VREFM)+((V(VREFP)-V(VREFM))/2^9)

The last term in this statement is 1/2 LSB, which is given by

$$\frac{1}{2} \text{ LSB} = \frac{V_{REF+} - V_{REF-}}{2^{N+1}} \text{ assuming } V_{REF+} > V_{REF-} \geq 0 \quad (30.29)$$

We are level-shifting the input and common-mode voltage because we want to make the model as flexible as possible. For example, we want the ADC model to function if \$V_{REF+} = 0.5\$ V and \$V_{REF-} = 0.25\$ V. Note that if \$V_{REF-} = 0\$ and \$V_{REF+} = VDD\$, the model can be simplified.

```

*** START IDEAL 8-BIT ADC Subcircuit ****
.subckt ADC8bit VDD VREFP VREFM Vin B7 B6 B5 B4 B3 B2 B1 B0 CLOCK

    * Set up common mode voltage
    BCM VCM 0 V=(V(VREFP)-V(VREFM))/2

    * Set up logic switching point
    R3 VDD VTRIP 100MEG
    R4 VTRIP 0 100MEG

    * Ideal input sample and hold
    XSH VDD VTRIP VIN OUTSH CLOCK SAMPHOLD

    * Level shift by VREFM and 1/2LSB
    BPIP PIPIN 0 V=V(OUTSH)-V(VREFM)+((V(VREFP)-V(VREFM))/2^9)

    * 8-bit pipeline ADC
    X7 VDD VTRIP VCM PIPIN B7 VOUT7 ADCBIT
    X6 VDD VTRIP VCM VOUT7 B6 VOUT6 ADCBIT
    X5 VDD VTRIP VCM VOUT6 B5 VOUT5 ADCBIT
    X4 VDD VTRIP VCM VOUT5 B4 VOUT4 ADCBIT
    X3 VDD VTRIP VCM VOUT4 B3 VOUT3 ADCBIT
    X2 VDD VTRIP VCM VOUT3 B2 VOUT2 ADCBIT
    X1 VDD VTRIP VCM VOUT2 B1 VOUT1 ADCBIT
    X0 VDD VTRIP VCM VOUT1 B0 VOUT0 ADCBIT
.ends

    * Ideal Sample and Hold subcircuit
.SUBCKT SAMPHOLD VDD VTRIP Vin Vout CLOCK
Ein Vinbuf 0 Vin Vinbuf 100MEG
S1 Vinbuf VinS VTRIP CLOCK switmod
Cs1 VinS 0 1e-10
S2 VinS Vout1 CLOCK VTRIP switmod
Cout1 Vout1 0 1e-16
Eout Vout 0 Vout1 0 1
.model switmod SW
.ends

    * Pipeline stage
.SUBCKT ADCBIT VDD VTRIP VCM VIN BITOUT VOUT
S1 VDD BITOUT VIN VCM switmod
S2 0 BITOUT VCM VIN switmod
Eouth Vinh 0 VIN VCM 2
Eoutl Vinl 0 VIN 0 2
S3 Vinh VOUT BITOUT VTRIP switmod
S4 Vinl VOUT VTRIP BITOUT switmod
.model switmod SW
.ends
*** END ADC Subcircuit ****

```

Figure 30.38 SPICE subcircuit netlist for an ideal 8-bit ADC.

Example 30.9

Modify the SPICE code of Fig. 30.38 so that the subcircuit simulates an ideal 12-bit ADC.

We can change the level-shift statement (change 9 to 13) to

```
* Level shift by VREFM and 1/2LSB
BPIP PIPIN 0 V=V(OUTSH)-V(VREFM)+((V(VREFP)-V(VREFM))/2^13)
```

and add to the pipeline algorithm

```
* 12-bit pipeline ADC
X11 VDD VTRIP VCM PIPIN B11 VOUT11 ADCBIT
X10 VDD VTRIP VCM VOUT11 B10 VOUT10 ADCBIT
X9 VDD VTRIP VCM VOUT10 B9 VOUT9 ADCBIT
X8 VDD VTRIP VCM VOUT9 B8 VOUT8 ADCBIT
X7 VDD VTRIP VCM VOUT8 B7 VOUT7 ADCBIT
```

where the last statement is a modification, in the 8-bit ideal ADC, of the existing statement for X7. ■

We can simulate the operation of our ideal 8-bit ADC in several ways. Let's begin by simply applying a ramp from V_{REF-} to V_{REF+} (0 to 1.5 V) to the ADC while clocking the ADC at 100 MHz. The results are shown in Fig. 30.39. Additional simulations using the ideal ADC will be left as an exercise for the reader. We are now in a position to put our ideal ADC and DAC together so that we can look at the spectral response and limitations resulting from quantization noise.

Summary

It's important to realize the usefulness of the simulation models we have just developed. In any mixed-signal simulation using SPICE we can use our ideal ADC to generate a digital signal, most often a sinewave, as an input source. We can use the DAC to convert a digital word into an analog waveform. We can then take the discrete Fourier transform of the resulting analog waveform, using the SPICE "spec" command, and view the digital data's spectrum.

Note that in this chapter we are only discussing the use of the *offset binary format* (see Ch. 29) for our digital words (0000... corresponds to V_{REF-} and 1111... corresponds to $V_{REF+} - 1$ LSB). It should be clear that we can modify our ideal data converters to work with any data format. We could also add digital logic to our converter subcircuit for the format conversion and continue to use the ideal ADC/DAC developed in this chapter.

30.3 Quantization Noise

At this point we should understand the sampling process and understand the operation of the ideal ADC and DAC. What we want to do in this section is understand quantization noise (the effective noise added to a signal after passing through an ADC) and how it affects the spectrum of a signal.

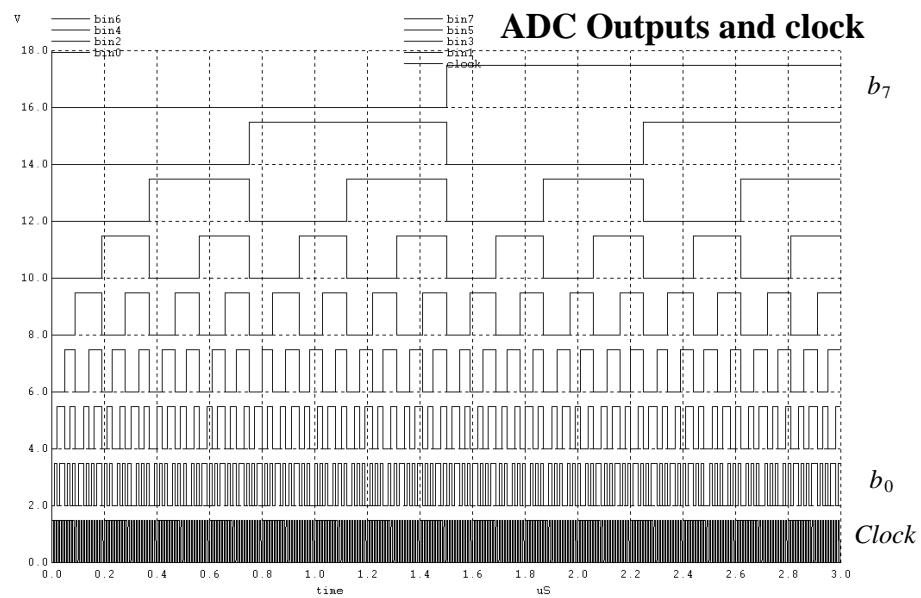
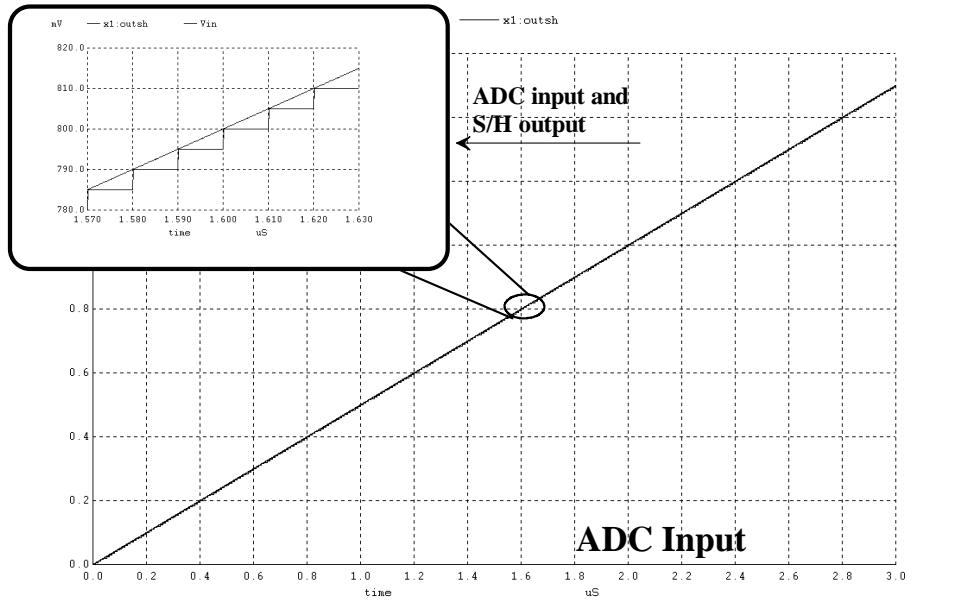


Figure 30.39 Simulating an ideal 8-bit ADC.

30.3.1 Viewing the Quantization Noise Spectrum Using Simulations

Consider the simple connection of an ideal 8-bit ADC to an ideal 8-bit DAC as shown in Fig. 30.40. If we put a 7 MHz sinewave into the ADC with an amplitude of 0.75 V and an offset of 0.75 V (so the sinewave swings from V_{REF-} [= 0 V here] to V_{REF+} [= 1.5 V]) and clock the ADC at 100 MHz the waveforms of Fig. 30.41 result. Note how the output of the DAC looks very similar to the output of an ideal S/H (see Fig. 30.25). Now, however, the amplitude of the DAC output signal is quantized, that is, within 1 LSB (= 1.5/256 or 5.859 mV for the present simulation) of the ADC input. This quantization is not obvious after looking at Fig. 30.41 (the time domain response). However, looking at the spectrums of the ADC input and the DAC output reveals the difference in the noise floor between the two (Fig. 30.42.) The inherent noise floor in the simulation that is associated with the input signal is approximately -140 dB ($0.1 \mu\text{V}$) The noise floor associated with the DAC's output (the signal + quantization noise) is approximately -60 dB (1 mV). It is desirable to determine what sets this value and its spectral content. Again note that the ADC quantizes the signal, which results in the quantization noise.

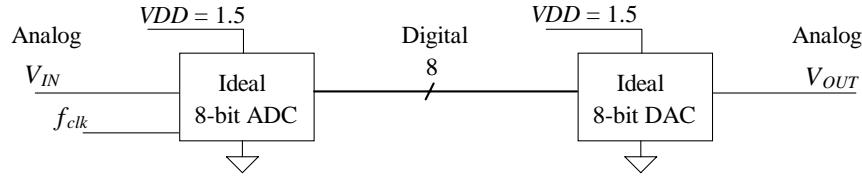


Figure 30.40 Passing a signal through an ADC and then through a DAC.

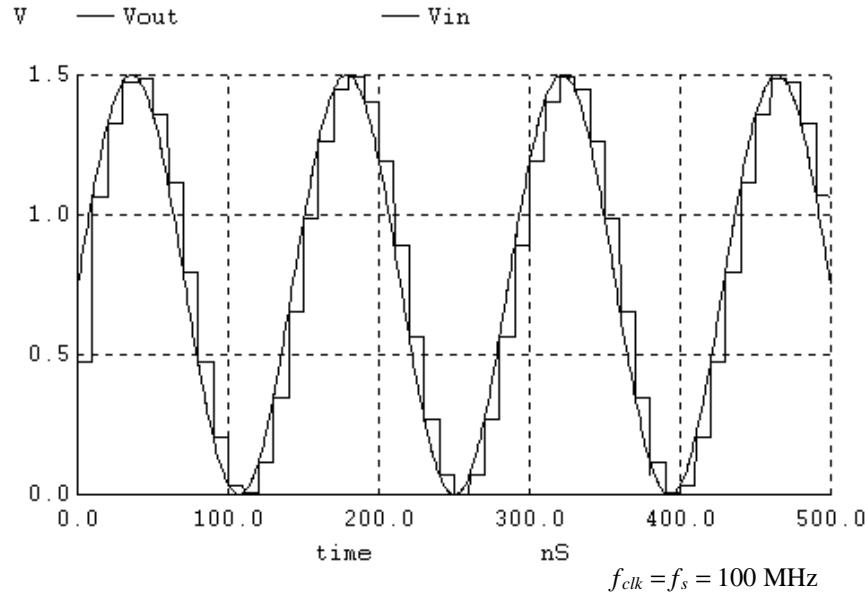


Figure 30.41 Seven MHz ADC input and the corresponding DAC output.

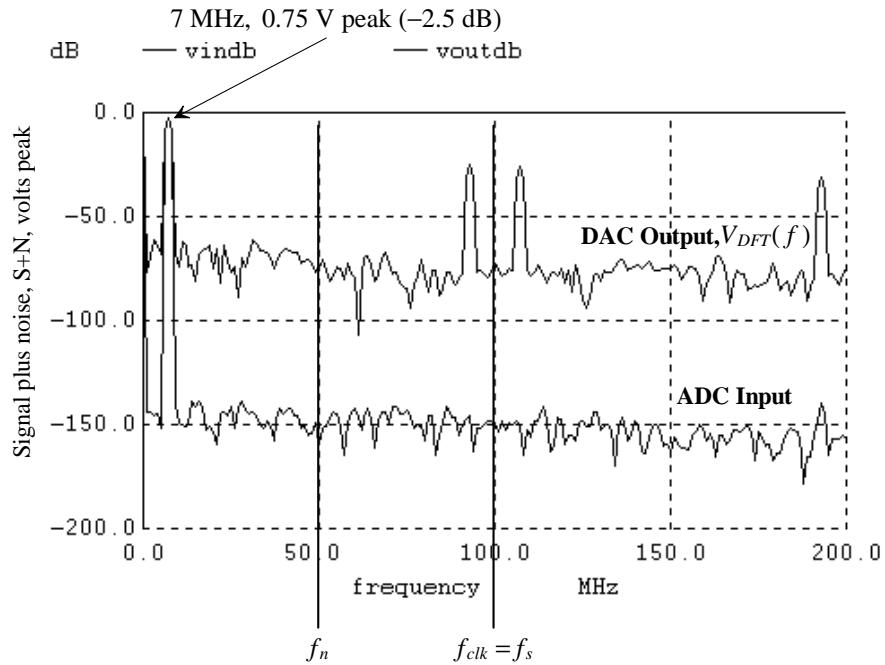


Figure 30.42 Spectrums of the signals shown in Fig. 30.41.

To characterize the spectral characteristics of the quantization noise let's make the following assumptions (Bennett's criteria) concerning the signal we are converting:

1. The input (to the ADC) signal's amplitude variation falls between V_{REF+} and V_{REF-} so that no saturation of the digital output code occurs. Exceeding the normal operating range of the ADC affects the quantization noise spectrum by adding spurs or spikes to the output spectrum.
2. The ADCs LSB is much smaller than the input signal amplitude. When this isn't the case, the output of the ADC can appear squarewave like (when converted back into an analog waveform) and result in a spectrum, once again, that contains spikes or spurs. We'll see later in the book that adding or subtracting a feed-back signal (from the output based on the expected or past quantization noise) to the input modifies this requirement.
3. The input signal is busy (not DC or a low frequency input). We define busy, for the moment, as meaning that no two consecutive outputs of the ADC have the same digital code. For the ideal ADC of Fig. 30.41 1 LSB = 5.86 mV and $T_s = 10$ ns so that the input must change at least 5.86 mV every 10 ns. We'll see that adding a *high-frequency* dither or pseudorandom noise signal to the input, which can be filtered out later (either using a digital filter or when we pass the output through the reconstruction filter), can make the requirement on the input of being busy practical in an actual circuit.

We use these assumptions (Bennett's criteria) in the following discussion unless otherwise indicated.

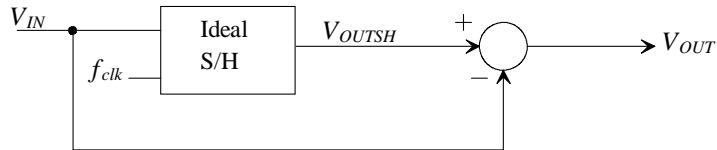


Figure 30.43 Taking the difference between the S/H input and output.

An Important Note

It's important to note that simply sampling an input waveform, using a S/H, does not result in quantization noise, as seen in Fig. 30.29. The amplitude into the ideal S/H, at the sampling instant, is exactly the same as the amplitude out of the ideal S/H. To understand why this is important, consider the test setup shown in Fig. 30.43. If we input the 3 MHz sinewave of Ex. 30.7 into this circuit, we get the outputs shown in Fig. 30.44. Clearly there *is* a difference between the S/H's input and its output. However, this difference has nothing to do with noise, an unwanted signal, since passing the output of the S/H, V_{OUTSH} ,

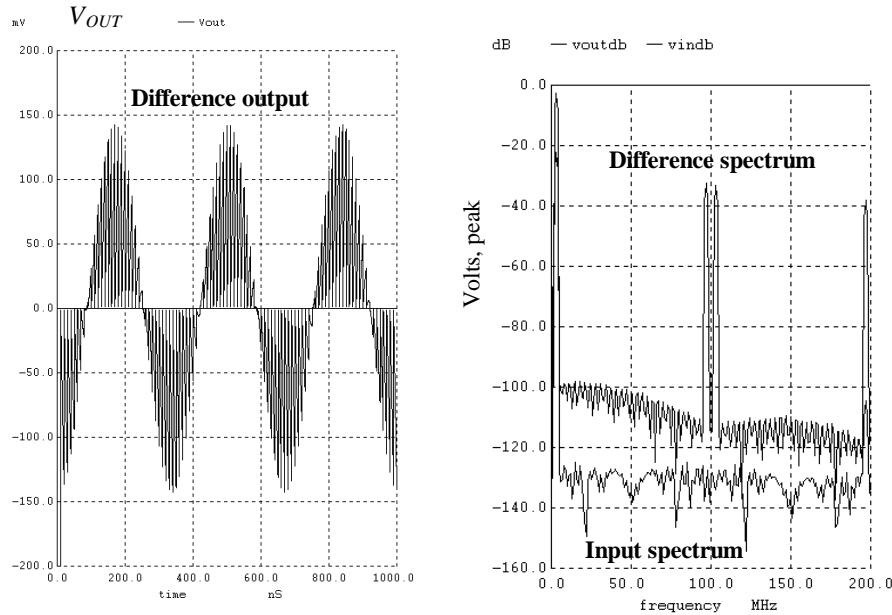


Figure 30.44 (a) Time domain difference between S/H input and output and (b) spectrum.

through the ideal reconstruction filter of Fig. 30.30 results in an exact replica of the S/H input V_{IN} .

RMS Quantization Noise Voltage

If we were to set up a test configuration similar to that shown in Fig. 30.43 (see Fig. 30.45), where the input to the ADC is subtracted from the DAC output, the resulting output waveform would have little to do, in every case, with the quantization noise. This is especially true when the input to the ADC contains a broad frequency spectrum extending from DC to the Nyquist frequency, $f_n = f_s/2$. However, if we simply apply a slow linear ramp to the input of this test setup (to limit the input frequency spectrum), see Fig. 30.45, we can (1) see the resulting quantization noise over a wide frequency spectrum and (2) observe that the transfer curve, in the time domain, is similar to Fig. 30.37. Note that this input violates Bennett's criteria (which, as we'll see, means the noise power spectral density is flat from DC to the Nyquist frequency).

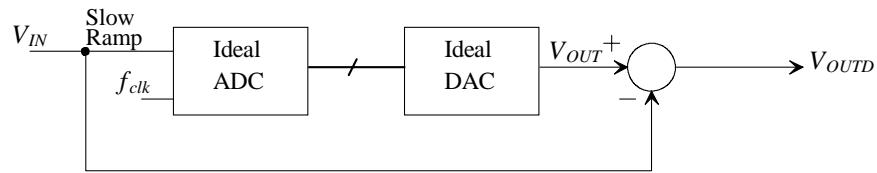


Figure 30.45 Taking the difference an ADC input and the DAC output.

A section of the input and output, using the test setup of Fig. 30.45, is shown in Fig. 30.46a. It's important to understand the input/output relationship between the ideal ADC and DAC shown in this figure. (Note that clocking the ADC too slow or putting in a ramp that rises too quickly will distort this waveform.) As an example, when the ADC input is slightly above 758.79 mV, in this figure, the ADC output code (input to the DAC) changes. The ADC output code can be calculated as $755.9 \text{ mV}/1 \text{ LSB} (1 \text{ LSB} = 1.5/256 = 5.86 \text{ mV}$ for the present simulation) or 129 when the input is slightly below 758.79 mV and 130 when the input is slightly above 758.79 mV. Looking at the transfer curves in this figure it appears as though the output changes when the ADC code is 129.5 or 758.79 mV/1 LSB. This, as seen in Fig. 30.46b and discussed below, results in centering the quantization error around the input (and is the reason we shifted the ADC transfer curves by 1/2 LSB when we developed our ideal ADC model).

The difference output, between the two signals of Fig. 30.46a, is shown in Fig. 30.46b. Some points to note about this sawtooth waveform are that 1) its average value is zero, 2) the waveform contains an abrupt transition (and so we expect a wideband output spectrum similar to that which occurs after sampling a waveform), and 3) its peak-to-peak amplitude is 1 LSB. Like a sinewave, which also has zero average value, we can characterize this quantization error waveform by looking at its root-mean-square (RMS)

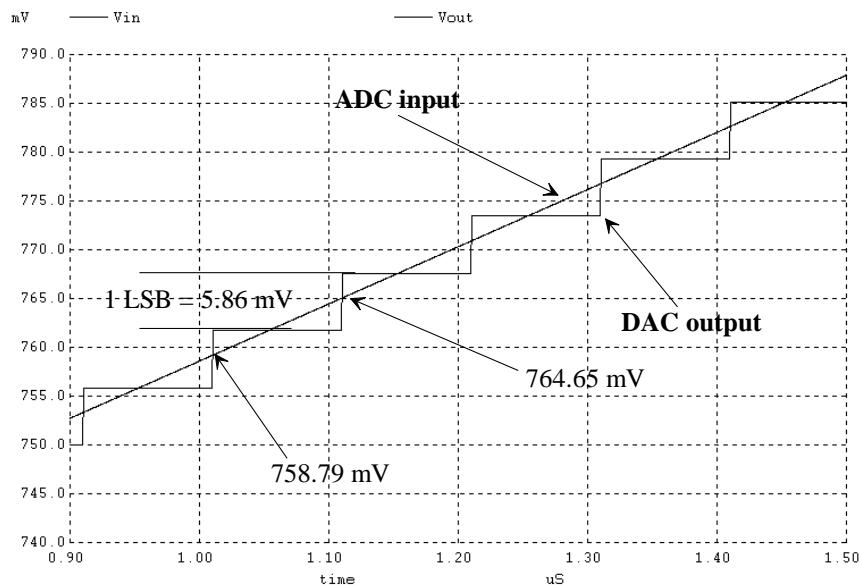


Figure 30.46 (a) ADC input and DAC output.

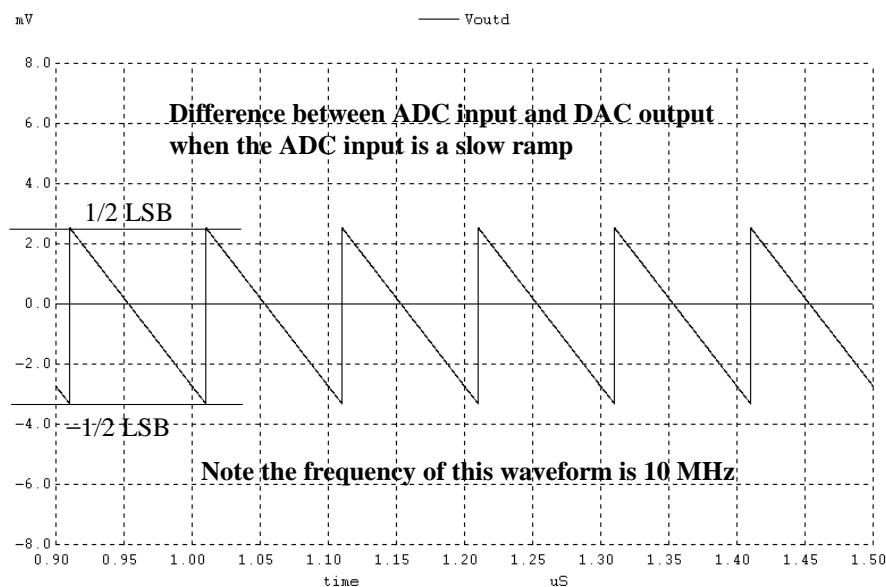


Figure 30.46 (b) Difference between ADC input and DAC output.

value. This value can be calculated using

$$V_{Qe,RMS} = \sqrt{\frac{1}{T} \int_0^T (0.5 \text{ LSB} - \frac{1 \text{ LSB}}{T} \cdot t)^2 dt} = \frac{1 \text{ LSB}}{\sqrt{12}} = \frac{V_{LSB}}{\sqrt{12}} \quad (30.30)$$

This value is the RMS quantization noise voltage for a specific data converter. Note that the value of the period for this sawtooth waveform, T , doesn't appear in the evaluated result of this equation. Also note that the sampling frequency, f_s , isn't present in this equation. For our present discussion where 1 LSB is 5.86 mV, $V_{Qe,RMS} = 1.69 \text{ mV}$ or -55.43 dB .

Treating Quantization Noise as a Random Variable

If Bennett's criteria hold, then the quantization noise voltage can be thought of as a random variable falling in the range of $\pm 0.5 \text{ LSB}$, as seen in Fig. 30.47. The probability that the quantization error is -0.2 LSB is the same as the probability that the error is 0.4 LSB . In other words, there is no reason why the quantization error should have one value more often than another value.

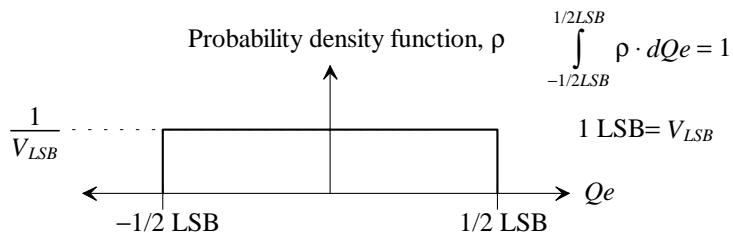


Figure 30.47 Probability density function for the quantization error in an ADC assuming Bennett's criteria hold.

The quantization error noise power is the variance of the probability density function. The RMS quantization error voltage is the square root of the quantization noise power. The variance of the probability density function (the quantization noise power, P_{Qe}) is given, knowing the average of the quantization error, \bar{Qe} , is zero, by

$$P_{Qe} = \int_{-1/2 LSB}^{1/2 LSB} \rho \cdot (Qe)^2 \cdot dQe = \frac{V_{LSB}^2}{12} \quad (30.31)$$

so that, once again, the RMS quantization noise voltage is

$$V_{Qe,RMS} = \frac{V_{LSB}}{\sqrt{12}} \quad (30.32)$$

Again, if our LSB voltage is 5.86 mV, then, once again, $V_{Qe,RMS} = 1.69 \text{ mV}$ (-55.4 dB). If we look at Fig. 30.42, we see that the peak noise voltage, at a given frequency, varies essentially over the entire spectrum (white noise) and has a value ranging from -60 dB down to less than -80 dB . Note that although the entire spectrum contains quantization

noise it is not because of the sampling process used in the ADC (and so quantization noise doesn't experience aliasing). Quantization noise is added to the signal after the sampling process during the analog-to-digital conversion process. To qualitatively understand why the quantization error spectrum is white, in Fig. 30.42, we remember that there are abrupt transitions in the DAC output, and if the quantization error is truly random, the times between the changes have varying periods. We might speculate that by simulating a longer time or using a multiple frequency input so as to "exercise" the ADC, the resulting quantization errors are further randomized and the resulting error spectrum will be flat.

Calculating RMS Quantization Noise Voltage from a Spectrum

The voltage spectrums for the quantization noise and the input signal (Fig. 30.46a and b) are shown in Fig. 30.48. Note that the harmonics of the noise are, as we would expect, spaced by 10 MHz ($= 1/T$). Also note, the sampling frequency doesn't affect the value of the RMS quantization noise voltage. The peak voltage of the fundamental tone in the quantization noise voltage spectrum is approximately -55 dB or -58 dB RMS (peak voltages [magnitudes] are used in the spectrum plots shown in this chapter unless otherwise indicated). To relate the RMS noise voltage calculated above, i.e., -55.4 dB, to the values shown in Fig. 30.48 we would: (1) convert the peak voltages to RMS values by subtracting 3 dB from each value, (2) square each result to get the mean-squared voltage, (3) sum the mean squared values, and (4) take the square root of this sum to get the RMS quantization noise voltage. Looking at the peak values of the first three tones in the

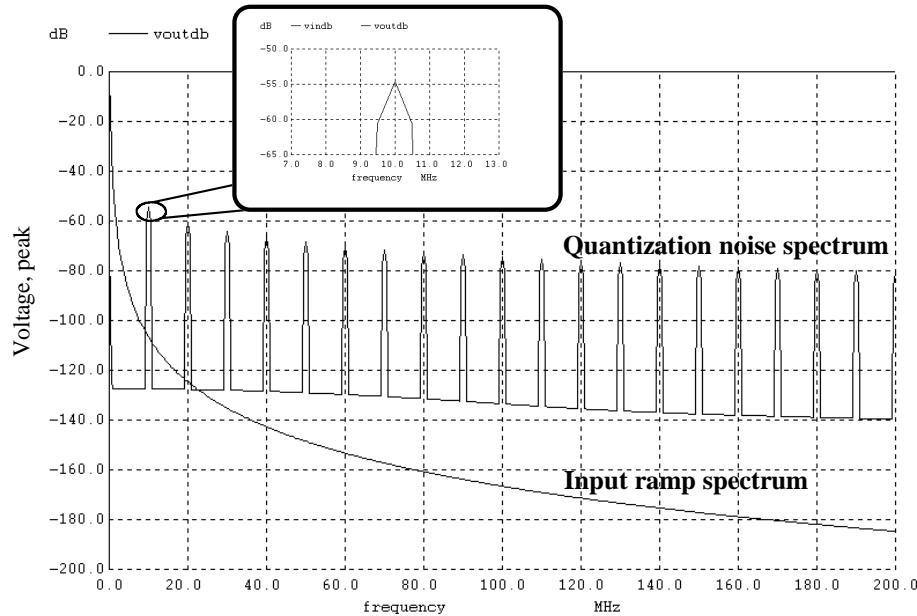


Figure 30.48 Input and quantization noise spectrums for the signals of Fig. 30.46.

quantization noise spectrum, -55 dB, -60 dB, and -65 dB we convert these values to RMS voltages, 1.26 mV, 0.708 mV, and 0.398 mV. The quantization noise, calculated using only the first three tones, is then $\sqrt{(1.26)^2 + (0.708)^2 + (0.398)^2} = 1.5$ mV, RMS or -56.5 dB. Clearly, increasing the number of tones used in this calculation will cause the result to approach Eq. (30.30) (1.69 mV).

To calculate the RMS quantization noise voltage from a DAC output spectrum we sum the mean-squared contribution from each component (after removing the desired tones from the spectrum) and then take the square root of the result (as mentioned above.) See $V_{DFT}(f)$ in Fig. 30.42 as an example. If the resolution of the DFT is f_{RES} , then we can write this as

$$V_{Qe,RMS} = \frac{1}{\sqrt{2}} \cdot \sqrt{\sum_{k=0}^{M-1} V_{DFT}^2(k \cdot f_{RES})} \quad \text{where } M = \#DFTpoints \quad (30.33)$$

The factor of root two comes from changing the peak values in a spectrum to RMS quantities. Note that the term "bin" is often used to describe the fact that the output of the DFT is only valid at discrete frequencies (the bins.) The number of bins is also known as the number of points in an DFT output vector (labeled #DFTpoints or M in Eq. [30.33]). This is seen in Fig. 30.48 (also shown in Fig. 30.48 is a DFT problem known as the "picket fence" effect, which will be discussed in a moment). If the DFT resolution in a simulation is 1 MHz then the DFT output, assuming the starting frequency is DC (0), will have nonzero values at DC, 1 MHz, 2 MHz, and so forth. If the stop frequency is 200 MHz, then the total number of points in a DFT output vector is 201.

Note that if Bennett's criteria hold, Eq. (30.33) will equal $V_{LSB}/\sqrt{12}$. If it doesn't hold, then the $V_{Qe,RMS}$ calculated using Eq. (30.33) will be different from $V_{LSB}/\sqrt{12}$. An input high-frequency sinewave violates Bennett's criteria. For example, consider sampling a 25 MHz sinewave at 100 MHz. If the sample points occur at the zero crossing points on the sinewave and at the peak/valley points, the resulting DAC output will be a rectangular waveform with a well-defined spectrum.

After a simulation the length of the DFT output vectors can be determined using
display

command or for a specific vector, say voutd, we can use

`print length(voutd)`

These commands show, in the WinSPICE command window, the length of the vectors and the type, e.g., complex, real, dB, etc. (for the display command) or the length of a particular vector (for the "print length" command).

If we want to set a component of the DFT to a value, say zero, we may want to use a command sequence like

```
let m=mag(voutd)
let m[7]=0
```

This sequence of commands sets the eighth element of a vector to zero (since we start at element zero). This is often done to remove a tone in an output spectrum resulting from the input signal or some other distortion.

Example 30.10

Using WinSPICE calculate the RMS quantization noise voltage from the spectrum of Fig. 30.48.

We begin by running the simulation that generates this figure (running the netlist file Fig30_48.cir). After the simulation is completed we type, in the WinSPICE command window,

```
display
```

and the following appears:

frequency	: frequency, real, 401 long [default scale]
vin	: voltage, complex, 401 long
vindb	: decibel, real, 401 long
voutd	: voltage, complex, 401 long
voutdb	: decibel, real, 401 long

We see from this that the length of the DFT is 401. Note that we could have used the length command, as we'll do below, to determine the length of the DFT instead of the display command.

To calculate the RMS quantization noise voltage we can use the following:

```
let m=mag(voutd)
let qnoise=0.707*sqrt(mean(m*m)*length(m))
print qnoise
```

which gives a result of 2.08 mV, a value larger than the 1.69 mV calculated for $V_{Qe,RMS}$ earlier. Before we discuss the discrepancy between the two RMS voltages, notice that we took the average (mean) of the mean-squared value of voutd and then multiplied the result by its length to sum the mean-squared voltages as specified by Eq. (30.33). ■

Now we need to discuss the difference between the SPICE simulated and the calculated RMS quantization noise voltages above. While the implementation of a discrete Fourier transform is outside the subject matter of this book, we can comment here on two DFT problems and how to reduce their effects; namely, the *picket-fence effect* and *spectral leakage*.

The picket-fence effect, and the visual reason for its name, is shown in the insert figure in Fig. 30.48. *Coherent sampling* (synchronizing the quantization error, Fig. 30.46, with the sampling clock) was used to magnify the effect. In our discussion above we assumed, for the first tone at 10 MHz, that the contribution to $V_{Qe,RMS}$ was -55 dB. On closer inspection, we see that there are also contributions, -61 dB, to the quantization noise at 9.5 MHz and 10.5 MHz. At these two side frequencies, the amplitude

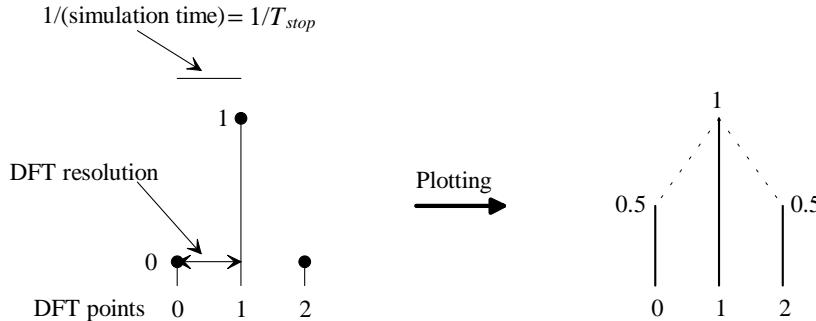


Figure 30.49 Showing the origins of the picket-fence effect.

contribution is one-half of the main contribution (-6 dB below the main contribution). Figure 30.49 shows that if one over the simulation time is equal to the DFT resolution then the boundaries between the adjacent DFT output points, spaced by the reciprocal of the simulation time, are coincident. This results in averaging adjacent contributions when the DFT output is generated. To reduce the effects of this averaging, we can increase the length of the simulation (having the effect of decreasing the window width used with the DFT). We can modify Eq. (30.6) to reduce the picket-fence effects by requiring

$$\text{Simulation time, } T_{\text{stop}} \geq \frac{2}{\text{DFT resolution}} = \frac{2}{f_{\text{res}}} \quad (30.34)$$

Example 30.11

Repeat Example 30.10 if Eq. 30.34 is used to set the DFT resolution and simulation length.

In Example 30.10 the simulation time was 2,000 ns. We could increase the simulation time to 4,000 ns or reduce the DFT resolution from 500 kHz to 1 MHz. In order to keep the simulation time reasonably short (and to avoid spectral leakage discussed next) we will decrease the DFT resolution and leave the simulation time at 2,000 ns. Figure 30.50 shows the resulting output spectrum with the decreased DFT resolution (now 1 MHz). The RMS quantization noise voltage calculated by SPICE, from this spectrum, is 1.71 mV RMS. ■

To understand what is meant by "spectral leakage," consider the sinewave with infinite duration shown in Fig. 30.51a. When a DFT is performed on a time domain waveform, the first step is to "window" the waveform. The simplest window is the rectangular window. In a simulation the duration of the sinewave is finite and set by the simulation time or transient stop time, T_{stop} . We can think of taking the infinite duration sinewave of Fig. 30.51a and multiplying it by the rectangular waveform of Fig. 30.51b to obtain the waveform used in the simulation, Fig. 30.51c. This multiplication means the resulting waveform is the convolution of the original sinewave spectral response (an impulse) and the frequency domain transform of the squarewave (a Sinc waveform) in the

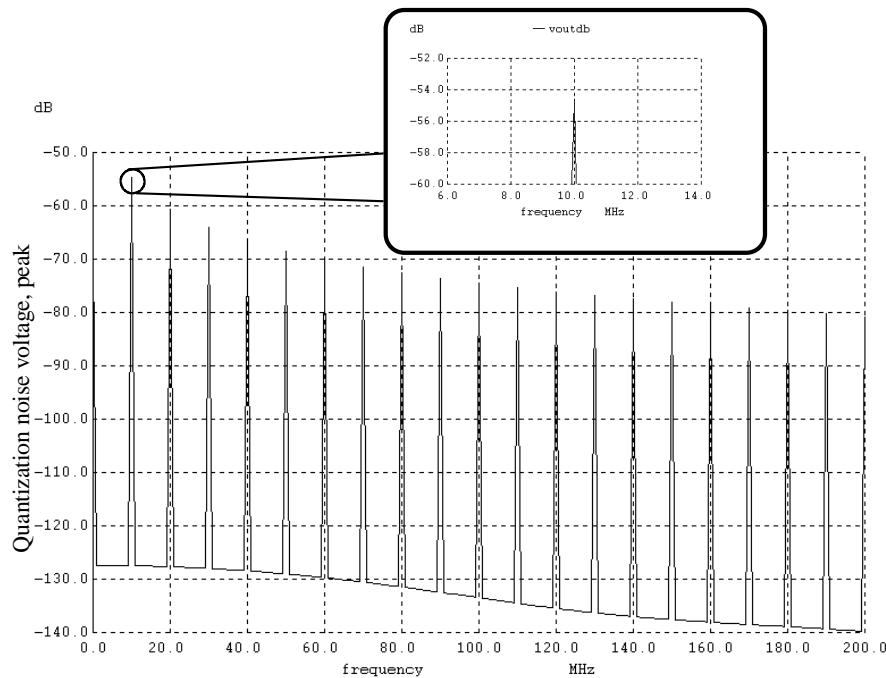


Figure 30.50 Eliminating the picket-fence effect from the simulation in Fig. 30.48.

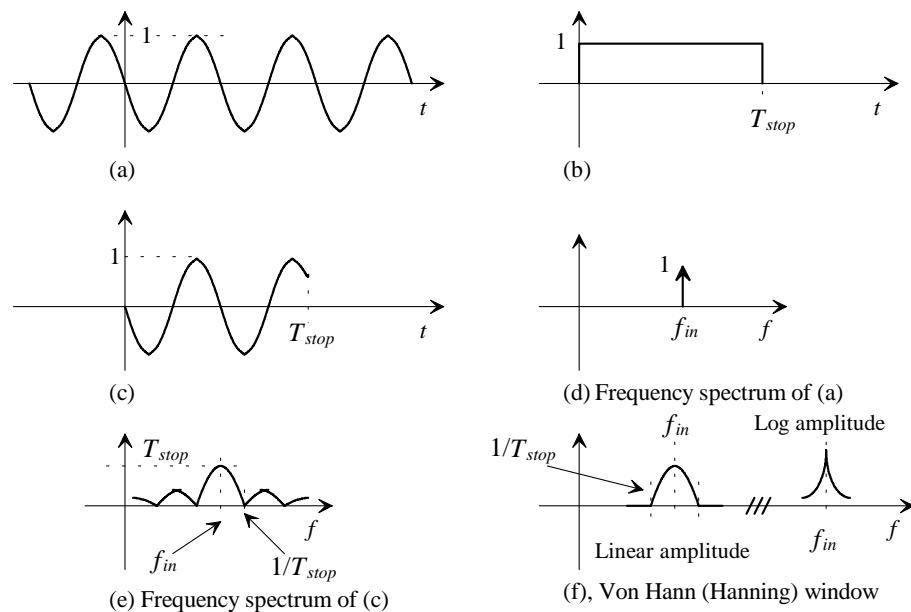


Figure 30.51 Showing how spectral leakage, resulting from a DFT, affects a waveform.

frequency domain. The result is that instead of the sinewave spectral response being an impulse function, as seen in Fig. 30.51d, it is a weighted Sinc waveform, Fig. 30.51e. Note how the DFT spectral response of the sinewave, Fig. 30.51e, is spread out or "leaks" into the frequencies around the actual or continuous time response. The large ratio of the peak value of the Sinc pulse to its first sidelobe is usually undesirable. Rather, to minimize these sidelobes, other windowing functions are used. The one we are using in this chapter is the von Hann (a.k.a. Hanning or Cosine) window shown, without the sidelobes, in Fig. 30.51f. The response is shown on both linear and log amplitude scales and the width of the window is $2/T_{stop}$ at its base ($= 1 \text{ MHz}$ if $T_{stop} = 2,000 \text{ ns}$).

Example 30.12

Using SPICE, show the spectrum of a 1 V (peak) sinewave at 10 MHz over a spectral range of DC to 200 MHz with an DFT resolution of 1 MHz and a simulation time of 2,000 ns (windowed frequency spread of 1 MHz, Fig. 30.51e).

The results are shown in Fig. 30.52. Note how the only point in these figures that has a nonzero value occurs at 10 MHz. The plotting lines are used to connect the five DFT output points shown in each of these figures. ■

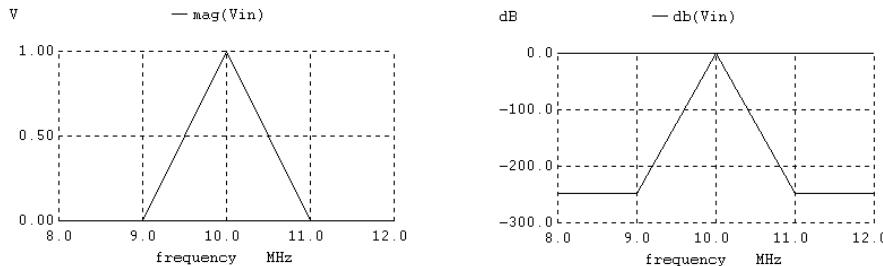


Figure 30.52 Output spectrum of 10 MHz sinewave showing the window's effect.

We were careful, in the previous example, to select the sinewave frequency to coincide exactly with one of the points where the DFT is calculated (10 MHz.) In the previous example the DFT points are calculated at DC, 1 MHz, 2 MHz, ..., 200 MHz. An error in the DFT output response occurs if spectral content doesn't fall on one of these frequencies. Consider the following example.

Example 30.13

Repeat Ex. 30.12 if the sinewave frequency is changed to 10.4 MHz.

The DFT output is shown in Fig. 30.53. Note that although the input frequency is at 10.4 MHz the peak in the DFT response still occurs at 10 MHz (a DFT output point). Also notice how the spectrum of the sinewave is effectively wider than the sinewave of Ex. 30.12. The 10.4 MHz sinewave is within the DFT resolution of both the 10 MHz and 11 MHz points. The result is effective spectral content at these frequencies. Sinewaves that do not fall exactly at the DFT calculation points are *smeared* in the DFT output spectrum. This smearing can spread across the

spectrum and affect spectral content at other frequencies. Consider the following example. ■

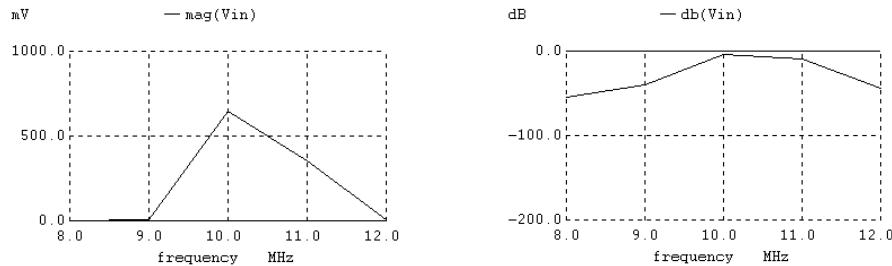


Figure 30.53 Magnitude and spectral response for the 10.4 MHz sinewave of Ex. 30.13.

Example 30.14

Using SPICE, plot the output spectrum resulting from adding the 10 MHz and 10.4 MHz sinewaves from the previous two examples.

The results are shown in Fig. 30.54. Note how, even though the 10 MHz sinewave has an amplitude of 1 V, the resulting output spectrum reports an amplitude of approximately 600 mV at 10 MHz. This is a result of contributions from the 10.4 MHz signal, after windowing, subtracting from the 10 MHz signal calculation point. ■

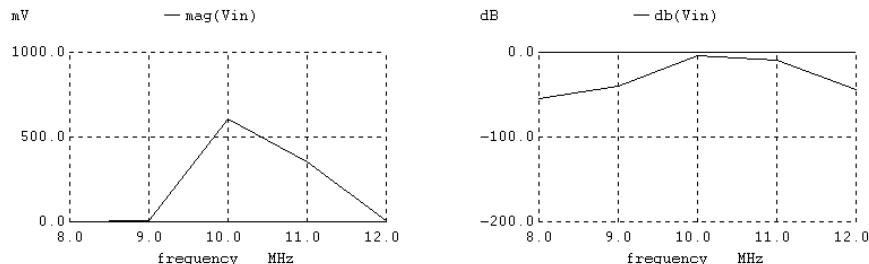


Figure 30.54 Magnitude and spectral response for the sum of the 10 and 10.4 MHz sinewaves.

It's important to understand that by increasing the simulation time, the window length increases, causing the width of the Sinc spectrum, see Fig. 30.51e, to decrease. However, increasing the simulation time without making a corresponding change in the DFT resolution can actually be harmful to the results. For example, if $1/T_{stop}$ is 100 kHz (simulation time of 10,000 ns) and the DFT resolution is 1 MHz then an input sinewave at 10.5 MHz will have no effect on the resulting output spectrum. In general, it's important to make

$$\text{Simulation time, } T_{stop} = \frac{2}{\text{DFT resolution}} = \frac{2}{f_{res}} \quad (30.35)$$

Also note that in a general simulation, which includes MOSFETs, we can set the step size used in a transient simulation with Eq. (30.8). However when using ideal components, as in this chapter, the step size can be increased to speed up the simulation time.

Example 30.15

Determine the RMS quantization noise voltage from the DAC output spectrum shown in Fig. 30.42.

Figure 30.42 was generated with a DFT resolution of 1 MHz and a simulation time of 1,000 ns. We will increase the simulation time to 2,000 ns. The resimulated spectrum of the DAC output noise is shown in Fig. 30.55. Notice in this spectrum that there is a signal at DC and 7 MHz (from the input signal.) Also, the aliased signals are present in the output spectrum at 93 MHz, 107 MHz, and 193 MHz. To calculate the quantization noise we would have to first zero these components out. We can use the following WinSPICE commands to calculate the quantization noise:

```
let m=mag(vout)
let m[0]=0
let m[7]=0
let m[93]=0
let m[107]=0
let m[193]=0
let qnoise=0.707*sqrt(mean(m*m)*length(m))
print qnoise
```

The resulting RMS quantization noise voltage is 1.72 mV. ■

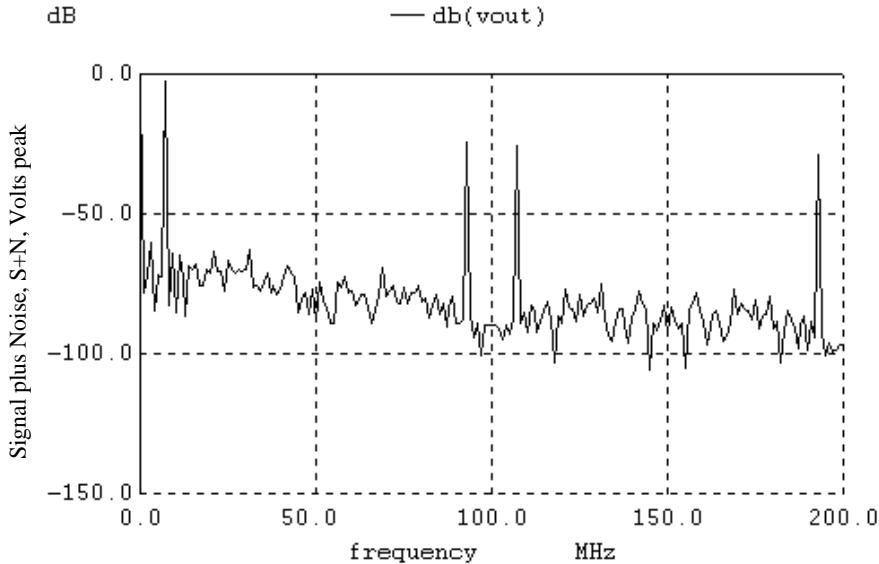


Figure 30.55 Simulating the circuit shown in Fig. 30.40 for 2,000 ns.

Note that the simulations we have shown in this chapter generate spectral responses out to twice the clocking frequency or 200 MHz when using a 100 MHz clock. To reduce simulation time we may limit the spectral response to the Nyquist frequency. Also, an important component of the simulations can be the addition of

```
.options RELTOL=1u
```

to a netlist. Not including this statement or one similar (e.g., $\text{RELTOL} = 10\mu$) in a netlist may limit the simulated noise floor to a significant voltage.

WinSPICE can also be useful if measured data is available. The data from a spectrum analyzer can be written to a text file and loaded into a WinSPICE vector using the *load* command. See the WinSPICE online manual.

The DFT's Relationship to the Continuous Time Fourier Transform

Before we leave this section, let's comment on how the discrete Fourier transform is related to the continuous time Fourier transform. We can write the continuous time Fourier transform of a time-varying function, $v(t)$, using

$$V(f) = \int_{-\infty}^{\infty} v(t) \cdot e^{-j2\pi f t} \cdot dt \quad (30.36)$$

or, if we assume a finite simulation time,

$$V(f) = \int_0^{T_{stop}} v(t) \cdot e^{-j2\pi f t} \cdot dt \quad (30.37)$$

To approximate this continuous time Fourier transform with discrete variables, we will use the following

$$dt = \Delta t \text{ and } t = k \cdot \Delta t \text{ where } k = 0, 1, \dots N \quad (30.38)$$

The variable Δt is the transient step time (the time difference between points in the DFT algorithm) and N is the number of time steps used in the algorithm ($T_{stop} = N \cdot \Delta t$). The frequencies where the DFT is calculated, assuming Eq. (30.35) is valid, are given by

$$f = n \cdot f_{res} = \frac{2n}{T_{stop}} \text{ where } n = 0, 1, \dots M - 1 \quad (30.39)$$

The variable M is the number of points in the DFT output vector (the number of frequencies the DFT is calculated at). Finally, we can relate the continuous time Fourier transform, evaluated at discrete frequencies, to the Discrete Fourier Transform with

$$V(f)|_{f=n:f_{res}} \approx \underbrace{\sum_{k=0}^N v(k \cdot \Delta t) \cdot e^{-j(4\pi/N)nk}}_{\text{discrete Fourier transform, } V_{DFT}(n)} \cdot \Delta t \quad (30.40)$$

(noting that 4π is used in the exponent because our DFT resolution was twice the reciprocal of the simulation time), or

$$V(f)|_{f=n f_{res}} \approx \Delta t \cdot V_{DFT}(n) \quad (30.41)$$

In general, a DFT output is scaled (divided by Δt) so that it approximates the continuous time Fourier Transform when it is plotted. This is usually transparent to the user of the DFT routine.

30.3.2 Quantization Noise Voltage Spectral Density

If the quantization noise voltage spectrum is truly flat (Bennett's criteria hold) we can determine the noise power spectral density of $V_{Qe,RMS}$, $V_{Qe}^2(f)$ with units of V^2/Hz , or the noise voltage spectral density, $V_{Qe}(f)$ with units of $\text{V}/\sqrt{\text{Hz}}$ by solving

$$\frac{V_{LSB}^2}{12} = 2 \int_0^{f_s/2} V_{Qe}^2(f) \cdot df \quad (30.42)$$

where the factor of 2 accounts for the power in the negative frequencies of the spectrum. (See Ch. 7 for a discussion of noise spectral densities.) We are assuming that the noise power is bandlimited to the Nyquist frequency (the output of the DAC is passed through an ideal RCF). Solving Eq. (30.42) yields

$$V_{Qe}(f) = \frac{V_{LSB}}{\sqrt{12f_s}} = \frac{V_{REF+} - V_{REF-}}{2^N \sqrt{12f_s}} \quad (30.43)$$

with units of $\text{V}/\sqrt{\text{Hz}}$. The quantization noise spectral density is inversely proportional to the sampling frequency. Figure 30.56 shows that we can model the ADC as a summing block with $V_{Qe}(f)$ added to the input signal.

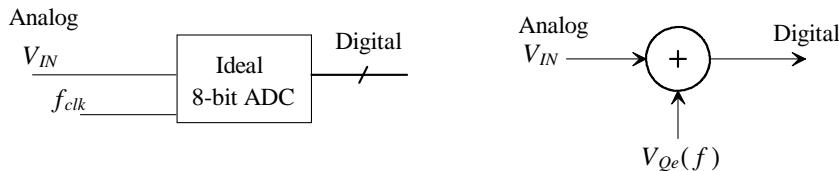


Figure 30.56 Modeling ADC quantization noise.

After looking at Eq. (30.43) we might think that by simply increasing the sampling frequency we can reduce the amount of quantization noise an ADC introduces into an analog input signal. While increasing the sampling frequency spreads the quantization noise spectral density out over a wider range of frequencies (see Fig. 30.57) with a corresponding reduction in amplitude, the sampling frequency doesn't affect the total RMS quantization noise voltage. However, bandlimiting the spectrum using a filter reduces the amount of quantization noise introduced into an input signal. In the simplest case a lowpass filter, which we will think of as an averager, can be used on the digital outputs of

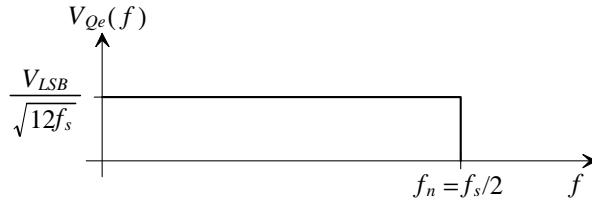


Figure 30.57 Quantization Noise Spectral Density.

the ADC to reduce the amount of quantization noise introduced into the signal. We can write the amount of noise introduced into an input signal over a range of frequencies using

$$V_{Qe,RMS}^2 = 2 \int_{f_L}^{f_H} V_{Qe}^2(f) \cdot df \text{ where } f_L < f_H \leq f_s/2 \quad (30.44)$$

Again the factor of 2 is used to account for the contributions to $V_{Qe,RMS}$ in the negative frequency spectrum (the DFT routine, discussed in the previous section, uses a one-sided spectrum so the factor of 2 is not necessary when making calculations using the SPICE simulation output data). Because the output of the ADC is a digital word, we would require a digital filter to bandlimit the output spectrum of the ADC. We will discuss digital filtering in the next chapter. For now let's show that the sampling frequency indeed doesn't affect the quantization noise, assuming Bennett's criteria are valid, and then let's discuss the concept of averaging to reduce quantization noise.

Example 30.16

Repeat Ex. 30.11 if the sampling frequency is increased from 100 MHz to 200 MHz.

Doubling the sampling frequency has no effect on the output quantization noise. It remains at 1.69 mV RMS. ■

Example 30.17

Repeat Ex. 30.15 if the sampling frequency is increased from 100 MHz to 200 MHz.

Again, the RMS quantization noise voltage remains essentially unchanged, i.e., 1.68 mV RMS. Recall that the circuit shown in Fig. 30.40 is used in this example and Ex. 30.15 with a 7 MHz input. It's instructive to show the time domain output of Fig. 30.40 when clocked at 200 MHz, Fig. 30.58, and compare it to Fig. 30.41 (the output of the circuit of Fig. 30.40 clocked at 100 MHz). Note how the DAC output voltage step size has decreased in Fig. 30.58 when compared to Fig. 30.41, yet the quantization noise remains unchanged. This shows, once again, that we must look at the spectrum of a signal to determine the quantization noise voltage and that the "coarseness" of an output signal has nothing to do with quantization noise. ■

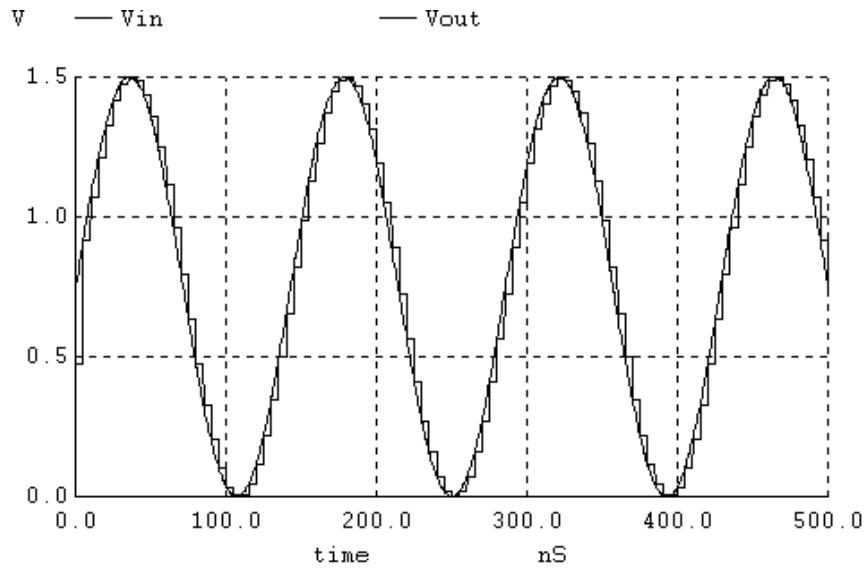


Figure 30.58 Output of the circuit shown in Fig. 30.40 with 7 MHz input and 200 MHz sampling clock. This figure should be compared to Fig. 30.41.

Reducing Quantization Noise Using Averaging

Consider the parallel combination of ADCs and DACs shown in Fig. 30.59. The top ADC and DAC are essentially the path we had back in Fig. 30.40 clocked at 100 MHz. The bottom path is a mirror image of the top except that its clock signal is inverted (delayed by 5 ns.) The two resistors are used to average the output of the DACs, or

$$V_{OUT} = \frac{V_{OUTA} + V_{OUTB}}{2} \quad (30.45)$$

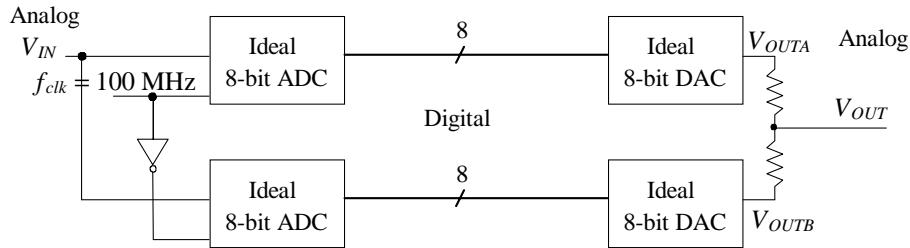


Figure 30.59 Using two paths to average the quantization noise.

Note that this configuration effectively samples the input at 200 MHz (200 Msamples/s [$2 \cdot f_s$]), as was accomplished in Fig. 30.58 except that now we are averaging consecutive samples. If we input a 7 MHz sinewave into this configuration, the same signal used in Fig. 30.41 or Fig. 30.58, we get the output shown in Fig. 30.60. Note the resemblance to Fig. 30.58. Also note the additional phase shift. The RMS quantization noise voltage now, however, has dropped from 1.68 mV to approximately 1.18 mV.

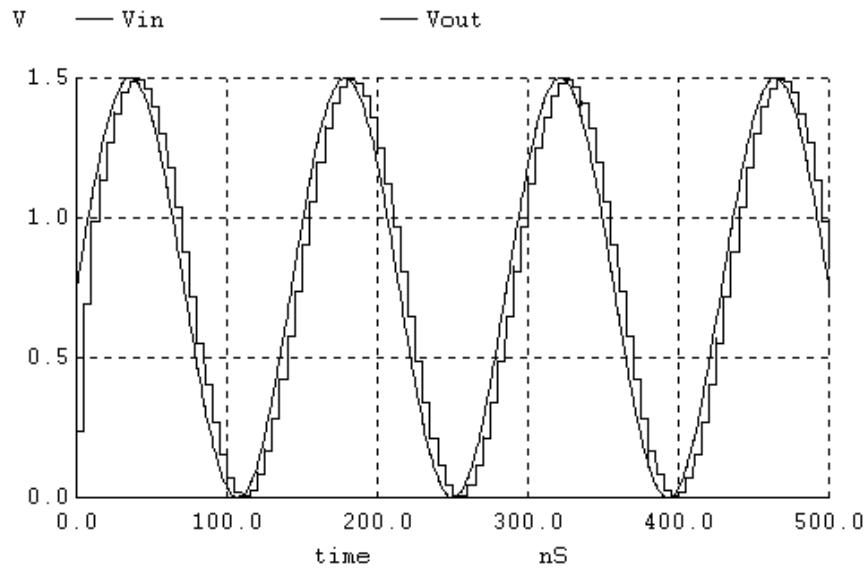


Figure 30.60 Output of the circuit of Fig. 30.59 with a 7 MHz input sinewave.

The Noise Spectral Density View of Averaging

In Fig. 30.59 we effectively doubled the sampling frequency. We can redraw Fig. 30.57 to show the effects of averaging by changing the amplitude in this figure from $V_{LSB}/\sqrt{12(f_s)}$ to $V_{LSB}/\sqrt{12 \cdot (2f_s)}$ and by increasing the frequency spectrum range as seen in Fig. 30.61. Assuming that we are still interested in the spectrum up to $f_s/2$, the RMS quantization noise can be calculated using

$$V_{Qe,RMS}^2 = 2 \int_0^{f_s/2} \frac{1}{2} \cdot \frac{V_{LSB}^2}{12f_s} \cdot df \quad (30.46)$$

or

$$V_{Qe,RMS} = \frac{1}{\sqrt{2}} \cdot \frac{V_{LSB}}{\sqrt{12}} \quad (30.47)$$

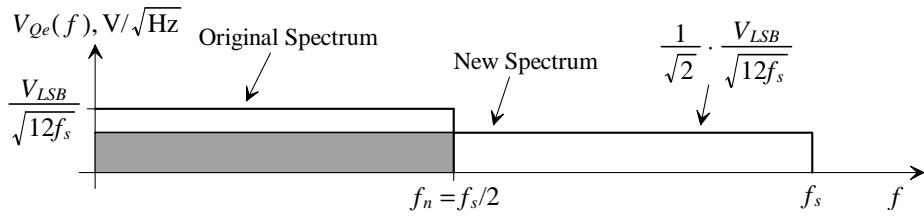


Figure 30.61 Quantization noise spectral density with two-sample averaging.
The sampling rate is effectively doubled.

In general, averaging K samples results in an RMS quantization noise voltage of

$$V_{Qe,RMS} = \frac{1}{\sqrt{K}} \cdot \frac{V_{LSB}}{\sqrt{12}} \quad (30.48)$$

The nonaveraged noise, $V_{LSB}/\sqrt{12}$, is reduced by the root of the averaging factor K . We know that the simulated $V_{Qe,RMS}$ in Ex. 30.17 was 1.68 mV. We simulated this circuit, again, using an averaging of two (Figs. 30.59 and 30.60) which resulted in a simulated $V_{Qe,RMS}$ of 1.18 mV. We could have estimated this RMS Quantization Noise beforehand using Eq. (30.48) as $(1.68 \text{ mV})/\sqrt{2} = 1.18 \text{ mV}$, which is, of course, the simulated result.

An Important Note

For averaging to effectively reduce the RMS quantization noise, the ADC and DAC must be linear (how linear will be answered in the next chapter). Examine Fig. 30.62. In the ideal situation, two adjacent codes are averaged to give an output that falls exactly in between the outputs of the data converter. In the case where the data converter has a nonlinearity, the averaged point doesn't necessarily provide an output that is much different from the data converter outputs themselves. If the data converter contains a

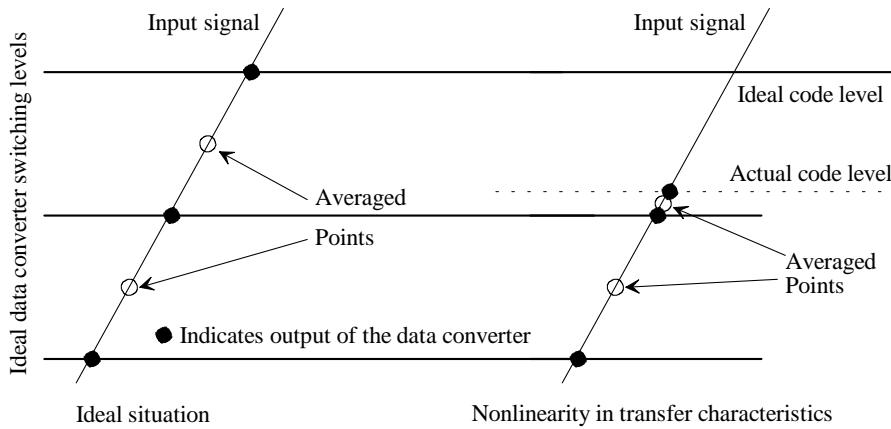


Figure 30.62 How ADC or DAC linearity affects averaging.

missing code (an input difference between two inputs at consecutive sampling times of 1 LSB results in the same output), then the averaging does nothing. If the data converter is nonmonotonic (an increase in the data converter's input doesn't result in an increase in its output) then the averaged value is meaningless. Finally, note that an input DC value (a digital code that isn't changing for the DAC, or an analog voltage that isn't changing for the ADC) or a value that isn't "busy" (not changing by at least 1 LSB in between sampling instances) will not benefit from averaging. These topics are discussed further in the next chapter.

Practical Implementation of Averaging in ADCs

The averaging topology shown in Fig. 30.59 is not practical in most situations. The silicon area required to implement the extra ADC and DAC generally costs more than is gained by the reduction in quantization noise. Also, as we'll see later, there are other techniques for averaging that provide a more efficient way to reduce quantization noise. Having said this and knowing that there are better ways, we will answer the question "How do we implement an ADC using averaging?"

Figure 30.63 shows how we can add a digital averaging filter to the output of the ADC to reduce quantization noise. The ADC and digital averaging filter are clocked at a rate of f_{clk} . If $K = 2$, for example, then the filter will sum its previous two inputs and output the result at a rate of f_{clk} . This filter could be implemented with a register and an adder. Note that the output word size increases when using the digital filter (it had better if we are reducing the quantization noise!). For example, if the output of the ADC is an 8-bit word, then the running sum coming out of the filter, again assuming $K = 2$, will be 9-bits.

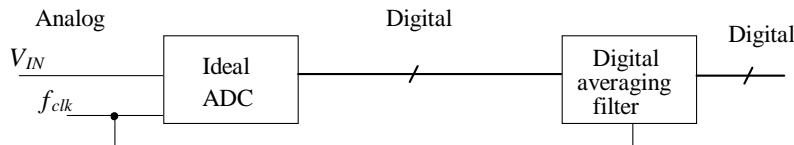


Figure 30.63 Using a digital averaging filter to reduce quantization noise.

We might, at this point, assume that we can use a low-resolution ADC, say 6-bits, with a significant amount of averaging to attain large resolutions (again the ADC must be linear). Assuming the input to the ADC is busy and we place restrictions on the bandwidth of the signals coming into the ADC then we can increase the resolution by averaging. We have to place restrictions on the bandwidth of the signal coming into the ADC because, unlike Fig. 30.59, we haven't increased the sampling rate of the signals. Therefore, the amplitude of the spectral density remains unchanged. For an averaging of two, we would have to limit our desired input signal bandwidth to $f_s/4$. If this wasn't the case, then an input sinewave at $f_s/2$ would average to zero. Again, these topics will be discussed in greater detail in Ch. 31.

REFERENCES

- [1] R. J. Baker, H. W. Li, and D. E. Boyce, *CMOS: Circuit Design, Layout, and Simulation*, Wiley-IEEE, 1998. ISBN 0-7803-3416-7
- [2] L. W. Couch, *Modern Communication Systems: Principles and Applications*, Prentice-Hall, 1995. ISBN 0-02325286-3
- [3] S. Haykin, *An Introduction to Analog and Digital Communications*, John Wiley and Sons, 1989. ISBN 0-471-85978-8
- [4] P. A. Lynn and W. Fuerst, *Introductory Digital Signal Processing*, Second Edition, John Wiley and Sons, 1998. ISBN 0-471-97631-8
- [5] E. P. Cunningham, *Digital Filtering: An Introduction*, John Wiley and Sons, 1995. ISBN 0-471-12475-3
- [6] R. K. Hester, *Introduction to Oversampled Data Conversion*, Notes from a tutorial at the 1995 International Solid-State Circuits Conference (ISSCC-95).
- [7] W. R. Bennett, "Spectra of Quantized Signals," *Bell System Technical Journal*, Vol. 27, pp. 446-472, July 1948.
- [8] J. C. Candy and G. C. Temes (eds.), *Oversampling Delta-Sigma Data Converters*, IEEE Press, 1992. ISBN 0-87942-285-8
- [9] S. R. Norsworthy, R. Schreier, and G. C. Temes (eds.), *Delta-Sigma Data Converters: Theory, Design, and Simulation*, IEEE Press, 1996. ISBN 0-7803-1045-4

LIST OF SYMBOLS/ACRONYMS

AAF - Antialiasing Filter

ADC - Analog-to-Digital Converter

C_H - Hold capacitor in a S/H

DAC - Digital-to-Analog Converter

DFT - Discrete Fourier Transform

DSP - Digital Signal Processing

Δt - Time difference between points used in a DFT

ϕ - Clock signal

f_{clk} - Frequency of the input clock signal

f_{in} - Input sinewave frequency

f_n - Nyquist frequency ($f_n = f_s/2$) which is 50 MHz in this chapter. Sometimes also called the folding frequency

f_{res} - Resolution of an DFT

f_s - Sampling frequency ($T_s = 1/f_s$), which is 100 MHz in this chapter. Sometimes also called the Nyquist rate

$H(j\omega)$ - Transfer function

$H_{RCFSH}(f)$ = Portion of the ideal S/H reconstruction filter

k - Counting index

K - Averaging factor or oversampling ratio

LPF - Lowpass Filter

$$\text{LSB} = V_{LSB} = \frac{V_{REF+} - V_{REF-}}{2^N} = \text{Least Significant Bit}$$

M - Number of points in the output of an DFT (or order of modulator, see Ch. 32)

n - Counting index

N - Number of bits in a data converter or the number of time steps in an DFT

OTA - Operational Transconductance Amplifier

Qe - Quantization error

RCF - ReConstruction Filter

RZ - Return-to-Zero format

S/H - Sample and Hold

SPICE - Simulation Program with Integrated Circuit Emphasis

ρ - Probability density function

$$Sinc(x) = Sin(x)/x$$

θ - Phase of a function

T - period of a periodic waveform

t_o - A time delay

T_s - Sampling interval ($T_s = 1/f_s$)

T_{stop} - Final simulation time

$V(f)$ - Spectral density, V/ $\sqrt{\text{Hz}}$

V_{CM} - Common-mode voltage, which is 0.75 V in this book

VDD - positive power supply voltage which is 1.5 V in this chapter

$V_{DFT}(n)$ - Discrete Fourier Transform of V

V_{OUTD} - Difference between an analog input and a digitized output, see Fig. 30.45.

V_{OUTdB} - Output signal in decibels

V_{INdB} - Input signal in decibels

V_{OUTSH} - Output voltage of a S/H

$V_{Qe}(f)$ - Quantization Error Spectral Density, V/ $\sqrt{\text{Hz}}$

$v_{in}(t)$ - Time domain input voltage

V_{inbuf} - Input signal after buffering

V_{ins} - Input signal after sampling

V_{LSB} - See LSB

V_{out} - Output voltage

V_p - Peak sinewave amplitude

$V_{Qe,RMS}$ - RMS quantization noise voltage

V_{REF+} - Positive reference used in an ADC or DAC, which is 1.5 V in this chapter

V_{REF-} - Negative reference used in an ADC or DAC which is ground in this chapter

VSS - negative power supply voltage which is 0 V in this chapter

$$z = e^{j2\pi f T_s} = e^{j2\pi \frac{f}{f_s}}$$

QUESTIONS

- 30.1** Qualitatively, using figures, show how impulse sampling a sinewave can result in an alias of the sampled sinewave at a different frequency.
- 30.2** What does linear phase indicate?
- 30.3** What does multiplying a signal by $e^{j2\pi f(-t_d)}$ indicate? How does the magnitude of the resulting signal change? How does the phase change?
- 30.4** Show, in the time domain, the input/output of the transmission line, and output of the comb filter in Fig. 30.11 if the input signal is a sinewave with a peak amplitude of 1 V and a frequency of 100 MHz. Show the two 500 Ω resistors average the input signal and the output signal of the delay line (transmission line).
- 30.5** Regenerate Fig. 30.19 if the switches are closed for 5 ns instead of 100 ps.
- 30.6** What sets the minimum resolution of a DFT in a SPICE spectral analysis?
- 30.7** Explain why the sinewave in Fig. 30.19 is "double sampled."
- 30.8** Explain why z is used in signal processing. What does multiplying a signal by z^{-1} do to the signal?

- 30.9** Sketch the implementation of a circuit that will multiply a digital signal by z^{-1} .
- 30.10** Sketch the time domain representation of the five signals shown in Fig. 30.29 on different plots. Regenerate Fig. 30.29 if the input signal is a 1 V peak sinewave at 5 MHz and zero offset. Explain the resulting plot.
- 30.11** Sketch the input and output spectrum for the following block diagram. Assume the DC component of the input is 0.75 V while the AC component is a sinewave at 4 MHz with a peak amplitude of 1 V. Assume the clock frequency is 100 MHz.

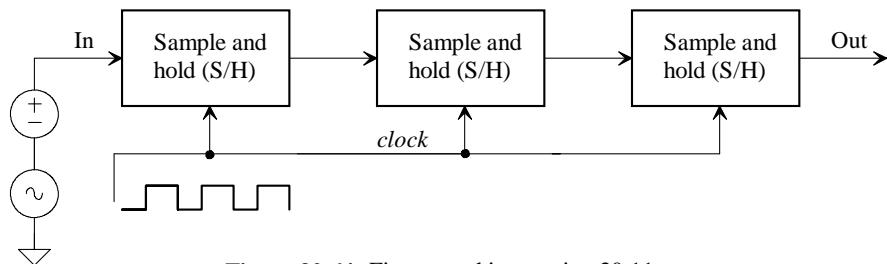


Figure 30.64 Figure used in question 30.11.

- 30.12** Using the models developed in the chapter design a SPICE model for the S/H of Fig. 30.31. Use the model to regenerate Fig. 30.29.
- 30.13** If $V_{REF+} = 1.5$ V and $V_{REF-} = 0$ regenerate Fig. 30.33 using SPICE. (Design a 3-bit ideal DAC model in SPICE.) The y-axis will be voltages in decimal form.
- 30.14** If, again, $V_{REF+} = 1.5$ V and $V_{REF-} = 0$, sketch Fig. 30.33 for a 1-bit DAC. Note that the digital input code will either be a 0 or a 1 and the analog voltage out of the DAC will be either 0 or 1.5 V. Using Eq. (30.23) what is the voltage value of 1 LSB? How does this compare to the value of 1 LSB we get from the sketch? Is Eq. (30.23) valid for a 1-bit DAC? Why? The 1-bit DAC will be a ubiquitous component in our noise-shaping modulators in Ch. 32 (see Fig. 32.28).
- 30.15** Using SPICE, implement an ideal 4-bit DAC and regenerate Fig. 30.36.
- 30.16** Why do the transfer curves of Fig. 30.37 show a shift of 1/2 LSB to the left? How do we implement this shift in SPICE?
- 30.17** Repeat question 30.16 for an ADC.
- 30.18** Using the models developed in questions 30.15 and 30.17 with a clock frequency of 100 MHz apply an input sinewave that has an amplitude of 750 mV peak centered around 750 mV DC and a frequency of 5 MHz to the input of the 4-bit ADC. If the ideal 4-bit DAC is connected to the digital outputs of the ADC, also show the DAC's analog output.
- 30.19** Using SPICE generate the spectrums of the input and output of the signals in question 30.18.

- 30.20** Does an ideal S/H introduce amplitude quantization noise into an input waveform? Why or why not?
- 30.21** Why are the amplitudes of the mirror images decreasing with an increase in frequency in Fig. 30.44b?
- 30.22** Show the details of the integration in Eq. (30.30).
- 30.23** How are voltage spectral density, power spectral density (PSD), average power, and RMS voltage related for a random signal? What are the units of each? Provide answers for both continuous signals and signals that are only defined at discrete frequencies.
- 30.24** How would we convert the voltage spectral density of Fig. 30.48 into a power spectral density plot? What term in Eq. (30.33) is the PSD? How would we rewrite Eq. (30.33) to give the average power of the quantization error?
- 30.25** Repeat Ex. 30.10 if we want to determine the quantization noise power. Show the simulation results and the commands used to determine this power.
- 30.26** Derive Eq. (30.43).
- 30.27** What term is the PSD in Eqs. (30.42) and (30.44)? What are its units?
- 30.28** Verify, with simulations, Ex. 30.16.
- 30.29** Verify, using simple circuit analysis, that resistors can be used to implement averaging as seen in Fig. 30.59 and Eq. (30.45).
- 30.30** How does averaging K samples of a random voltage variable reduce its RMS value? How does the power contained in the same variable get reduced by averaging?

Chapter

31

Data Converter SNR

In the last chapter we developed the idea of treating an analog-to-digital converter (ADC) as a noisy circuit block where the output of the ADC is the sum of quantization noise and the input signal. Logically, the next step in our development of concepts is to characterize a system using ADCs and DACs in terms of the signal-to-noise ratio (SNR). The ideal SNR for a data converter was developed back in Ch. 28 and is repeated here for convenience.

If we apply a sinewave with an amplitude of V_p (and thus an RMS value of $V_p/\sqrt{2}$) to an ADC input then, knowing the RMS quantization noise added to a busy ADC input signal is $V_{LSB}/\sqrt{12}$ (see Eqs. [30.30] and [30.32]), the resulting SNR for the ADC is given by

$$\text{SNR}_{ideal} = 20 \cdot \log \frac{V_p/\sqrt{2}}{V_{LSB}/\sqrt{12}} \quad (31.1)$$

If we remember that

$$V_{LSB} = 1 \text{ LSB} = \frac{V_{REF+} - V_{REF-}}{2^N} \quad (31.2)$$

and we assume that the largest possible amplitude sinewave is the ADC input (to maximize the SNR), that is,

$$2V_p = V_{REF+} - V_{REF-} \quad (31.3)$$

then Eq. (31.1) can be rewritten as

$$\text{SNR}_{ideal} = 20 \cdot \log \frac{2^N \sqrt{12}}{2\sqrt{2}} = 6.02N + 1.76 \text{ (in dB)} \quad (31.4)$$

Our goal in this chapter is to discuss how to determine the actual SNR of a data conversion system and to present topologies for improving data conversion system SNR (e.g., averaging, noise shaping, and others).

31.1 Data Converter SNR: An Overview

In this section we describe data converter performance, defining terms such as effective number of bits, dynamic range, signal-to-noise plus distortion ratio (SNDR), and spurious free dynamic range (SFDR). This discussion will be useful in understanding where problems or limitations with data converter performance originate.

31.1.1 Effective Number of Bits

Equation (31.4) relates the number of bits used in a data converter to the ideal SNR when the input signal is a sinewave that ranges from V_{REF+} to V_{REF-} . In reality the measured SNR, in most cases, will be different from the ideal value calculated using this equation. When an SNR is measured, we relate it to the *effective number of bits* using

$$N_{\text{eff}} = \frac{\text{SNR}_{\text{meas}} - 1.76}{6.02} \quad (31.5)$$

where the measured SNR (SNR_{meas}) is specified in dB.

Example 31.1

Determine the effective number of bits for an ADC with $V_{REF+} = 1.5$, $V_{REF-} = 0$, and a measured $V_{Qe,RMS}$ of 2 mV.

If we assume that the input peak amplitude, V_p , is $0.5 \cdot (V_{REF+} - V_{REF-})$ or 0.75 V, then the measured SNR is given by

$$\text{SNR}_{\text{meas}} = \frac{0.75/\sqrt{2}}{2 \text{ mV}} = 265 = 48.5 \text{ dB}$$

The effective number of bits, N_{eff} , is (from Eq. (31.5)) 7.76 bits. ■

Normally, the measured SNR (SNR_{meas}) is determined from the RMS quantization noise, which is determined using Eq. (30.33) with measured data. The amplitude and frequency of the input sinewave can be selected to maximize the SNR.

Example 31.2

Using the ideal 8-bit ADC and DAC shown in Fig. 31.1, which were developed in the last chapter, and a sampling frequency of 100 MHz ($=f_s$) show, using SPICE, that applying a full-scale sinewave at 24 MHz to this configuration will cause the resulting SNR to approach the ideal value given by Eq. (31.4).

Let's begin by calculating $\text{SNR}_{\text{ideal}}$. From Eq. (31.4), $\text{SNR}_{\text{ideal}}$ is roughly 50 dB, as the data converters have 8-bit resolution.

The time domain input and output of the circuit shown in Fig. 31.1, and the corresponding DAC output spectrum, are shown in Fig. 31.2. The input to the

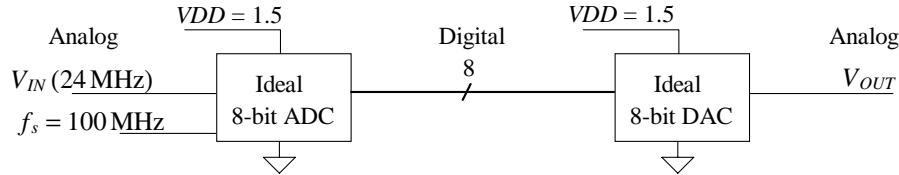


Figure 31.1 Test setup used in Ex. 31.2 to show deviation from ideal SNR.

ADC in Fig. 31.1 is a 24 MHz sinewave with a peak amplitude of 0.75 V centered on a DC voltage of 0.75 V (the peak-to-peak voltage of the input waveform is 1.5 V). The $V_{Qe,RMS}$ measured with SPICE, remembering to zero out the wanted signals at DC and 24 MHz and the images at 76 MHz, 124 MHz, and 176 MHz before calculating the noise (see Ex. 30.15), is 1.497 mV. The simulated SNR is $(0.75/\sqrt{2})/1.497$ mV or 354 (51 dB), which is very close to the value calculated at the beginning of the example. ■

It's important to understand that poor selection of the input frequency (or windowing function) can result in an SNR that is different from the ideal value calculated using Eq. (31.4). Selecting an input sinewave frequency, f_{in} , such that f_s/f_{in} is a whole number creates a condition where an integral number of input sinewave cycles fit perfectly into the sampling window (coherent sampling). This results in an output spectrum that contains isolated tones (helping to reduce the effects of spectral leakage on the SNR).

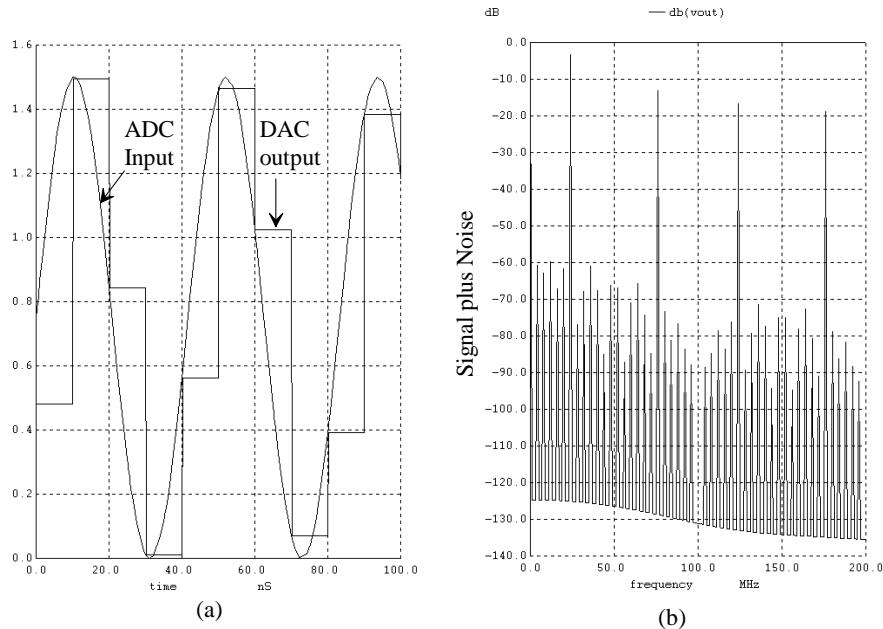


Figure 31.2 Example 31.2 (a) time domain input and output, and (b) spectrum of DAC output.

An additional effect to consider occurs when f_{in} is comparable to f_s . The input tone at f_{in} undergoes amplitude attenuation (-3.9 dB at f_n , Fig. 30.29).

Example 31.3

Repeat Ex. 31.2 if the input sinewave frequency is increased to 45 MHz.

The results of increasing the ADC input sinewave frequency in Fig. 31.1 to 45 MHz are shown in Fig. 31.3. Note how the DAC output contains tones at 5 MHz, 10 MHz, 15 MHz, etc., in addition to the desired tone at 45 MHz. The simulated $V_{Qe,RMS}$ is 2.26 mV. The input amplitude of the 45 MHz sinewave is 0.75 V (-2.5 dB). The simulated peak output amplitude at 45 MHz is 0.53 V (-5.5 dB). The simulated SNR can be calculated

$$\text{SNR} = 20 \log \frac{0.53/\sqrt{2}}{2.26 \text{ mV}} = 44.4 \text{ dB}$$

or 5.6 dB below the ideal value of 50 dB calculated using Eq. (31.4) for a data converter with a resolution of 8 bits. ■

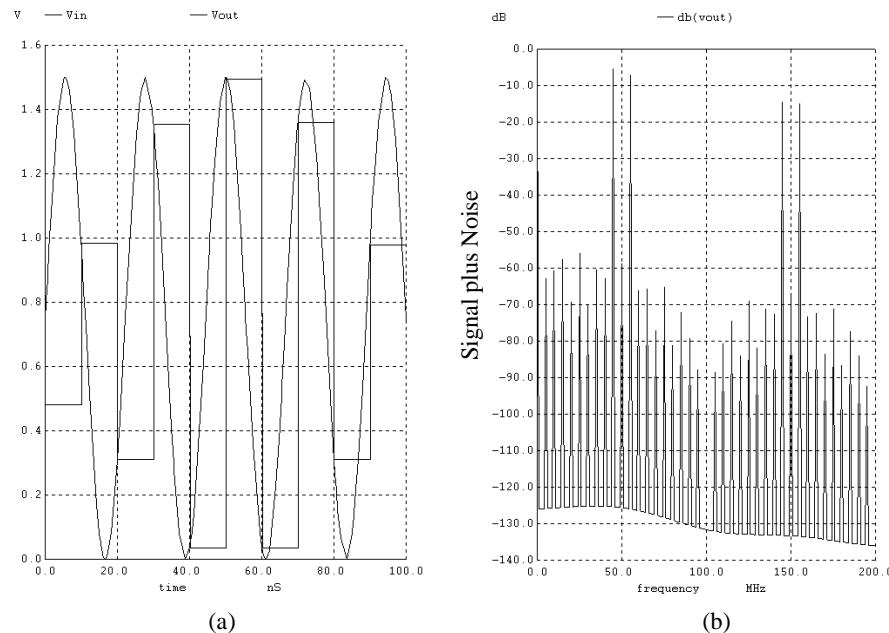


Figure 31.3 (a) DAC output with 45 MHz input and (b) the DAC output spectrum.

Signal-to-Noise Plus Distortion Ratio

In a practical data converter the output spectrum contains not only quantization noise but distortion resulting from nonlinearities and mismatch in the data converter circuitry. When

we calculate the RMS quantization noise voltage using Eq. (30.33) and nonideal components, we are actually calculating the noise plus the distortion in the spectrum. Up to this point we have only used ideal components, so that distortion in the output spectrums was absent. We can rewrite Eq. (30.33) to indicate that when it is used with a measured spectrum, both noise and distortion are included in the result as

$$V_{Qe+D,RMS} = \frac{1}{\sqrt{2}} \sqrt{\sum_{k=0}^{M-1} V_{DFT}^2(k \cdot f_{RES})} \quad \text{where } M = \#\text{DFT points} \quad (31.6)$$

The *signal-to-noise plus distortion ratio* is then given by

$$\text{SNDR} = 20 \log \frac{V_p/\sqrt{2}}{V_{Qe+D,RMS}} \quad (31.7)$$

The effective number of bits, from Eq. (31.5), can then be calculated using

$$N_{eff} = \frac{\text{SNDR} - 1.76}{6.02} \quad (31.8)$$

Example 31.4

Suppose that the test setup shown in Fig. 31.1 is used with an input sinewave having a frequency of 7 MHz, a peak amplitude of 0.75 V, and centered around 0.75 V (so that, once again, the sinewave swings from 0 V to 1.5 V.) Using SPICE simulation, determine the SNDR if there is a gain error in the ideal ADC in Fig. 31.1 (it's no longer ideal) so that each stage in the pipeline algorithm used to implement the ideal SPICE ADC has a gain of 2.1 instead of the ideal 2.0.

The resulting DAC output spectrum is shown in Fig. 31.4. The RMS noise plus distortion voltage, $V_{Qe+D,RMS}$, is calculated to be 16.9 mV, using SPICE and remembering to zero out the desired terms at DC and 7 MHz as well as the undesired images at 93 MHz, 107 MHz, and 193 MHz. The SNDR is then

$$\text{SNDR} = 20 \log \frac{0.75/\sqrt{2}}{16.9 \text{ mV}} = 30 \text{ dB}$$

and the effective number of bits is 4.7. In other words, a 5% gain error in the ADC amplifiers results in an effective resolution of almost half the ideal, 8-bit value. ■

Measuring SNDR requires a spectrum analyzer, when looking at the output of a DAC, or loading digital data (most often in decimal form) into a program that can perform a DFT (such as WinSPICE [utilizing the load command] or Matlab), when looking at the output of an ADC. Trying to measure SNDR using a time domain instrument, such as an oscilloscope, is usually a waste of time because the dynamic range of the instrument is comparable to the dynamic range of the data converter under test. Spectrum analyzers utilize narrow band filtering on their input to reduce the inherent noise measured in a circuit and can have dynamic ranges in excess of 120 dB over a very wide frequency spectrum. Also note that the SNDR is sometimes abbreviated as *SINAD* (signal-to-noise and distortion.)

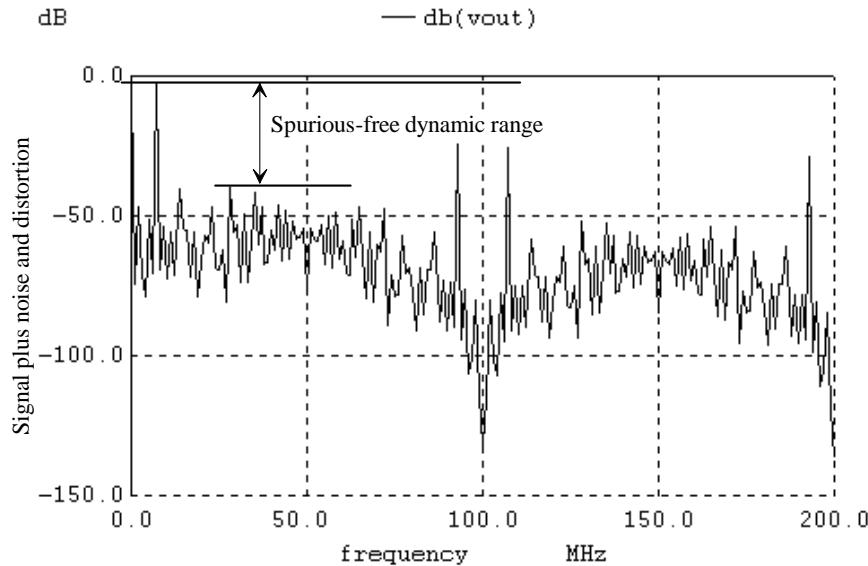


Figure 31.4 Output spectrum with ADC gain error (see Ex. 31.4).

Spurious Free Dynamic Range

Another specification of interest is the data converter's spurious free dynamic range. This term relates the peak signal in the output spectrum (the input sinewave or carrier) to the largest spike in the output spectrum up to the Nyquist frequency. This can be written using

$$\text{SFDR(dBc)} = \text{input carrier(dB)} - \text{unwanted tone(dB)} \quad (31.9)$$

For the spectrum shown in Fig. 31.4, the input sinewave (carrier) has an amplitude of 0.75 V (-2.5 dB), while the largest unwanted tone has an amplitude of -39.5 dB. The SFDR of this data converter is then 37 dBc.

Dynamic Range

The dynamic range of a data converter can be specified in several ways. We defined dynamic range back in Ch. 28 as the ratio of the largest output signal change (e.g., $[V_{REF+} - 1 \text{ LSB}] - V_{REF-}$) over the smallest output signal change (1 LSB). Remembering $1 \text{ LSB} = (V_{REF+} - V_{REF-})/2^N$ the dynamic range (DR) can be written as

$$\text{DR} = 20 \log \frac{V_{REF+} - (V_{REF+} - V_{REF-})/2^N - V_{REF-}}{(V_{REF+} - V_{REF-})/2^N} = 20 \log 2^N = 6.02N \quad (31.10)$$

If a 1,000 to 1 dynamic range (60 dB) is required, then a data converter with at least 10 bits is needed.

Another way to specify DR is as the ratio of the RMS full-scale input sinusoid amplitude, $V_p/\sqrt{2}$, to the input sinusoid amplitude (RMS) that results in an SNDR of 0 dB. (The RMS amplitude of the input signal is equal to the RMS quantization noise plus distortion, $V_{Qe+N,RMS}$, when the SNDR is 0 dB.) This is nothing more than saying that the SNDR can be used to specify DR.

Example 31.5

Determine the DR for the ideal ADC in Ex. 31.2 using Eq. (31.10). Compare the result to the SNDR calculated in Ex. 31.4.

Using Eq. (31.10), the DR is 48.16 dB (the ideal value). The SNDR calculated in Ex. 31.4 was 30 dB. Clearly, the SNDR is a better indication of DR than is the value obtained using Eq. (31.10). ■

Specifying SNR and SNDR

The SNR and the SNDR are usually specified as a function of input sinewave amplitude at a fixed frequency, Fig. 31.5. The x-axis in Fig. 31.5 is normalized so that an input sinewave with a peak-to-peak amplitude of $V_{REF+} - V_{REF-}$ corresponds to 0 dB. We might be wondering how we differentiate between SNR and SNDR as both, up to this point, have been calculated in the same way (Eqs. [31.6] and [30.33]). We continue to calculate SNDR using a data converter output spectrum, remembering to zero out the desired tones and images, and Eq. (31.6) as was done in Ex. 31.4. When we calculate the SNR, we follow the same procedure except that now we also zero out any *spikes* or *spurs* (spurious responses) in the spectrum that are "sticking up" above the noise floor in the spectrum. These spikes come from imperfections in the data converter and result in distortion in the output waveform. Note in Fig. 31.5 how the SNR and the SNDR coincide until the input signal amplitude gets reasonably large (so the distortion tones increase in amplitude above the quantization noise).

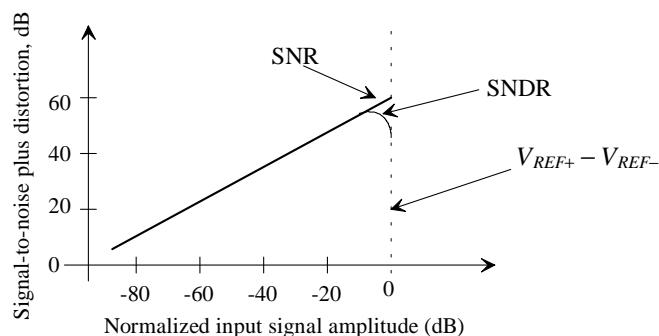


Figure 31.5 Specifying SNR and SNDR for a data converter.

31.1.2 Clock Jitter

We might assume that using the ideal data converters developed in the last chapter in a system with "real world" input and clock signals would give us a resolution (number of bits) set by the resolution of the ideal data converters used. However, if the clock signal used isn't ideal, the resolution will be less than ideal. We discussed this problem, aperture jitter, back in Ch. 28. Here we want to relate the amount of clock jitter to the data converters SNR and thus the effective number of bits. Clock jitter is the variation in the period of the clock signal around the ideal value

Figure 31.6 shows the basic problem. In this figure we have assumed the input sinewave frequency is running at the Nyquist frequency $f_n (= f_s/2)$ so that the sampling point (when the sinewave crosses zero in this figure) is seeing the fastest transition in the input signal. We assume the peak amount of jitter in the clock signal is ΔT_s . For example, if the sampling clock frequency is 100 MHz ($T_s = 10 \text{ ns}$) and the peak-to-peak clock jitter is 50 ps ($= \Delta T_s$), then the specification of the sampling clock stability is 5,000 ppm (where parts per million [ppm] $= 10^{-6}$ and $\Delta T_s = (\text{stability, ppm}) \cdot (1/f_s)$).

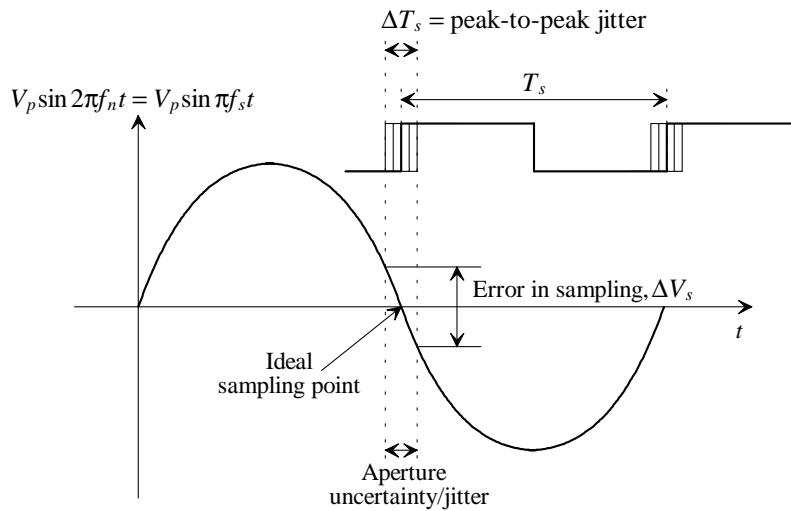


Figure 31.6 Data converter input signal and clock jitter.

The slew rate of the signal in Fig. 31.6, at the sampling point (when the clock signal transitions high), is given by

$$\frac{d}{dt}(V_p \sin \pi f_s t) = \pi f_s V_p \overbrace{\cos \pi f_s t}^{=1} = \pi f_s V_p \quad (31.11)$$

We can relate the uncertainty in the sampling instant, ΔT_s , to the uncertainty in the sampled voltage, ΔV_s , using

$$\frac{\Delta V_s}{\Delta T_s} = \pi f_s V_p, \text{ or } \Delta V_s = \Delta T_s \cdot \pi f_s V_p \quad (31.12)$$

If we require the uncertainty in the sampled voltage, ΔV_s , to be at most 0.5 LSB = $(V_{REF+} - V_{REF-})/2^{N+1}$ and we remember $V_p = (V_{REF+} - V_{REF-})/2$, then our maximum allowable peak-to-peak clock jitter can be determined for a particular data converter using

$$\Delta T_s \leq \frac{1}{2^N} \cdot \frac{1}{\pi f_s} \quad (31.13)$$

or in terms of the sampling clock stability

$$\text{Stability, ppm} = \Delta T_s \cdot f_s = \frac{\Delta T_s}{T_s} = \frac{1}{\pi \cdot 2^N} \quad (31.14)$$

Table 31.1 relates the stability requirements placed on a sampling clock for a data converter resolution, N , if less than 0.5 LSBs aperture error or sampling voltage uncertainty is required of the data converter.

Resolution, N	Stability, ppm	$\Delta T_s(\max), \text{ps}$	$\Delta T_s(\max), \text{ns}$
		If $f_s = 100 \text{ MHz}$	If $f_s = 44.1 \text{ kHz}$
6	5,000	50	113.4
8	1,250	12.5	28.3
10	312.5	3.125	7.1
12	78.1	0.78	1.77
14	19.5	0.195	0.443
16	4.9	0.05	0.111

Table 31.1 Maximum jitter, ΔT_s , for 0.5 LSB sampling uncertainty.

Example 31.6

Suppose a phase-locked loop (PLL) is used to generate a clock signal for a data converter. If the resolution of the data converter is 10 bits and the frequency of the sampling clock coming from the PLL is 900 MHz, then specify the maximum jitter allowed in the output of the PLL. Assume that the maximum sampling error allowed is 0.5 LSB and that the data converter is sampling a sinewave with a frequency of 100 MHz.

Because the sinewave being sampled has a frequency below the Nyquist value, Eq. (31.13) cannot be used directly. Instead, after reviewing the derivation of this equation, we can rewrite it in terms of any input signal frequency, f_{in} , as

$$\Delta T_s \leq \frac{1}{2^N} \cdot \frac{1}{2\pi \cdot f_{in}} \quad (31.15)$$

noting that when $f_{in} = f_n = f_s/2$, Eq. (31.15) reduces to Eq. (31.13). Using Eq. (31.13) with the numbers in this problem results in a peak-to-peak jitter of 1.56 ps! The reader familiar with PLL design will recognize that this is a very challenging requirement when designing a PLL (that is, to design a PLL with an output frequency of 900 MHz and an output jitter of 1.56 ps). ■

We're now in a position to answer how, given the peak-to-peak clock jitter ΔT_s , the SNR of a data converter is degraded from the ideal value (given in Eq. [31.4]) when the input sampling clock isn't ideal. Rewriting Eq. (31.15) and assuming

$$\Delta T_s \geq \frac{1}{2^N} \cdot \frac{1}{2\pi \cdot f_{in}} \quad (31.16)$$

(a resolution loss ≥ 0.5 LSB), we get

$$\Delta T_s = \frac{1}{2^{N-N_{Loss}}} \cdot \frac{1}{2\pi \cdot f_{in}} \quad (31.17)$$

where f_{in} is, once again, the frequency of the input sinewave, and N_{Loss} is the number of bits lost due to the excess jitter. Assuming Eq. (31.16) is valid, then when N_{Loss} is zero, the loss in resolution is 0.5 LSB and Eq. (31.17) reduces to the equality condition in Eqs. (31.15) or (31.16). The ideal data converter's SNR, assuming the only nonideal factor in the system is clock jitter, can be written as

$$\text{SNR} = 6.02(\overbrace{N - N_{Loss} - 0.5}^{\text{effective bits, } N_{eff}}) + 1.76 \text{ (in dB)} \quad (31.18)$$

Example 31.7

For an ideal 8-bit ADC clocked at 100 MHz, determine the SNR of the data converter with 100 ps of peak-to-peak jitter in the input sampling clock, ΔT_s , assuming the ADC's input is a full-scale sinewave at 25 MHz.

We can write the number of bits lost by solving Eq. (31.17) as a function of peak-to-peak jitter as

$$N_{Loss} = N + 3.32 \cdot \log(2\pi \cdot f_{in} \cdot \Delta T_s) \text{ assuming } \Delta T_s \geq \frac{1}{2^N} \cdot \frac{1}{2\pi \cdot f_{in}} \quad (31.19)$$

or

$$N_{Loss} = 8 + 3.32 \cdot \log(2\pi \cdot 25\text{MHz} \cdot 100\text{ps}) = 2 \text{ bits}$$

The effective number of bits, N_{eff} , is then 5.5 and the SNR is 34.87 dB. ■

Using Oversampling to Reduce Sampling Clock Jitter Stability Requirements

Suppose we limit the maximum input frequency coming into an ADC to f_{in} , such that the sampling frequency is related to the maximum ADC input frequency by

$$\frac{f_s}{2} = K \cdot f_{in} = f_n \text{ or } f_{in} = \frac{f_s}{2K} \quad (31.20)$$

In other words, we are getting at least $2K$ samples for every cycle of the input sinewave. If we were sampling at twice the Nyquist frequency (f_s), where $K = 1$, then we would get two samples for every cycle of the input signal. Notice that Eq. (31.15) gives the maximum jitter specification for a given input frequency and data converter resolution, but it doesn't specify the sampling frequency, f_s , or the sampling frequency period, T_s .

For a given maximum jitter, ΔT_s , we can reduce the requirements placed on the stability of the oscillator by increasing the sampling frequency. This can be written as

$$\text{Stability(new), ppm} = [\text{stability(old), ppm}] \cdot K \quad (31.21)$$

If we were sampling at 1 MHz and the stability required was 10 ppm, then the jitter in the sampling clock would be at most 10 ps, peak-to-peak. Increasing the sampling rate to 100 MHz, with 10 ps jitter would require an oscillator stability of 1,000 ppm. If we were to increase the sampling clock frequency to 1 GHz, then the stability of the clock would be at least 10,000 ppm (the period of the sampling signal is 1 ns and the jitter is 10 ps or 1% [10,000 ppm] of the sampling period).

Note that the oversampling factor symbol, K , is the same symbol that indicates the number of samples averaged on the output of a data converter to reduce RMS quantization noise voltage (see Sec. 30.3.2). The choice of variable was made for a reason (which will be discussed further in the next section).

Example 31.8

In Table 31.1 we saw that the 16-bit data converter clocked at 44.1 kHz could have at most 111 ps peak-to-peak jitter to limit the sampling uncertainty to 0.5 LSB. We saw that the stability required of the oscillator under these circumstances was 5 ppm at 44.1 kHz. What would happen to the stability requirements of the oscillator generating the sampling clock if we increased the sampling clock frequency to $128 \cdot 44.1 \text{ kHz} = 5.645 \text{ MHz}$?

We know that the input bandwidth, prior to increasing the sampling frequency, is limited to $44.1 \text{ kHz}/2$ or 22.05 kHz ($= B$, the bandwidth of the input signal). We then assume the maximum input frequency, f_{in} , remains at or below 22.05 kHz even after we increase the sampling frequency. We can define the oversampling factor, in this example, as

$$K = \frac{f_s/2}{f_{in}} = \frac{2.822 \times 10^6}{22.05 \times 10^3} = 128$$

The jitter requirement remains 111 ps whether we use a sampling frequency of 44.1 kHz or 5.645 MHz. However, now that the clock frequency has increased to 5.645 MHz, the stability required of the oscillator has gone from approximately 5 ppm to 640 ppm. ■

It's important to note that the oversampling ratio, K , is given by

$$K = \frac{f_s/2}{B} = \frac{f_n}{B} \quad \text{for } f_{in} \leq B \quad (31.22)$$

If we desire less than 0.5 LSBs aperture error, and we are using oversampling, then we can use Eqs. (31.13) and (31.22) to write

$$\Delta T_s \leq \frac{1}{2^N} \cdot \frac{K}{\pi f_s} = \frac{1}{2^N} \cdot \frac{1}{2\pi B} \quad (31.23)$$

where, once again, B is the bandwidth of the input signal and K is the oversampling ratio. As shown by this equation and in Eq. (31.21), using oversampling reduces the requirements placed on the stability of the sampling clock.

A Practical Note

We need to point out that the effects of clock jitter are possible even if the clock is perfectly stable because of the clock's finite transition times (rise and fall times). If the rise time of the clock signal in Fig. 31.6 is finite, say 50 ps, then the same derivations and discussions concerning jitter in the previous section can be applied to determine how the SNR of the data converter is affected. We would assume the aperture window is a function of the transition times of the sampling clock signal. The slower the transition times, the larger the sampling uncertainty. In any practical data converter the SNR, and thus the effective number of bits, will be reduced because of the clock jitter and finite transition times as the input signal frequency increases.

Modeling Clock Jitter with SPICE

It's useful in many situations to determine how clock jitter affects a data converter's performance. Consider the block diagram shown in Fig. 31.7. This is our basic configuration, used previously, to show data converter operation. Now, however, we have changed the sampling clock from an ideal pulse source to a source that contains jitter. The questions we want to answer in this section are "How do we use SPICE to model a jittery clock source?" and "How does the jitter affect the SNDR of the data converter?"

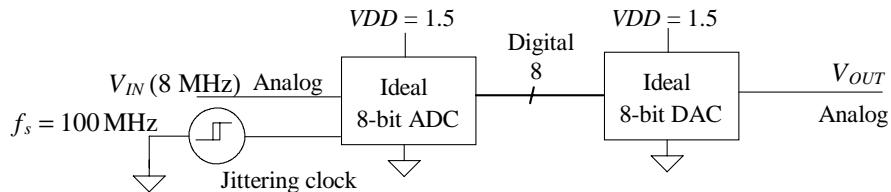


Figure 31.7 Simulating ideal data converters using a sampling clock with jitter.

To begin, let's consider the single frequency frequency modulation (SFFM) source available in SPICE. This source generates a frequency modulated (FM) sinewave using the following syntax

SFFM(VO VA FC MDI FS)

where the parameters describe the following function

$$V_{SFFM}(t) = VO + VA \cdot \sin([2\pi \cdot FC \cdot t] + [MDI \cdot \sin(2\pi \cdot FS \cdot t)]) \quad (31.24)$$

The modulation index (MDI) will set the peak-to-peak jitter time in the waveform while the signal frequency (FS) describes the rate at which this jitter varies. The carrier frequency (FC) will set the clock frequency (FC = 100 MHz in Fig. 31.7). The term VO is available to add a DC offset to the signal (which we will assume is zero in our discussion), and the value VA sets the amplitude of the frequency modulated sinewave. The question we need to answer now is, "How do we convert the FM sinewave generated using Eq. (31.24) into a squarewave suitable for driving our ADC?"

Figure 31.8 shows that a switch and the SFFM source can be used to generate the sampling clock with jitter. When the FM source transitions above zero, the top switch closes and the clock output goes high. When the source transitions below zero, the bottom switch closes and the clock output goes low.

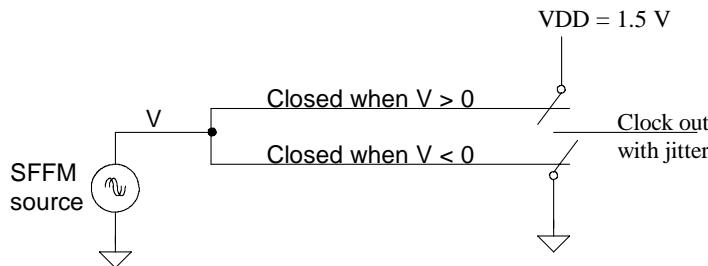


Figure 31.8 Generating a clock with jitter using a switch and an SFFM source.

The average period of the sampling clock generated by the SFFM source is given by

$$T_s = \frac{1}{FC} \text{ or } f_s = FC \quad (31.25)$$

The peak phase excursion of the clock signal is set by MDI. The peak phase excursion can be related to the sampling frequency using

$$2 \cdot MDI = 2\pi \cdot \frac{\Delta T_s}{T_s} \quad (31.26)$$

or the peak-to-peak jitter, ΔT_s , is given by

$$\Delta T_s = \frac{1}{2\pi} \cdot \frac{2MDI}{FC} = \frac{MDI}{\pi f_s} \quad (31.27)$$

Example 31.9

Using SPICE, generate the output spectrum of an oscillator assuming the oscillator frequency is 100 MHz and the peak-to-peak jitter is 100 ps. Assume FS = 1 MEG.

We can begin this example by noting FC = 100 MHz and the modulation index is $\pi \cdot 100\text{ps} \cdot 100\text{MEG} = 0.0314$. The SPICE netlist and the resulting spectrum are shown as follows and in Fig. 31.9, respectively.

```

* Figure 31.9 CMOS2: Mixed-Signal Circuit Design *
.tran .2n 1000n 0 .2n UIC
*WinSPICE command scripts
*#destroy all
*#run
*#plot clock xlimit 0 100n
*#plot vclock xlimit 0 100n
*#linearize clock
*#spec 90MEG 110MEG 200k clock
*#plot db(clock)
VDD VDD 0 DC 1.5

Vclock Vclock 0 DC 0 SFFM 0 1 100MEG 0.0628 1MEG
Rclock Vclock 0 100MEG

SH      VDD      clock   Vclock  0      Switmod
SL      0       clock   0      Vclock  Switmod
Rload   clock   0      100MEG

.model switmod SW
.end

```

Note that the resolution of the DFT was 200 kHz (set by Eq. [30.35] and that the signal frequency, FS, was 1 MEG (which sets the spacing between the tones in Fig. 31.9). The period of the 100 MHz clock, in this example, varies sinusoidally from 9.95 ns to 10.05 ns over a time frame of 1 μ s (1/FS.) While this spectrum is interesting, it isn't representative of an actual oscillator where the noise is random.

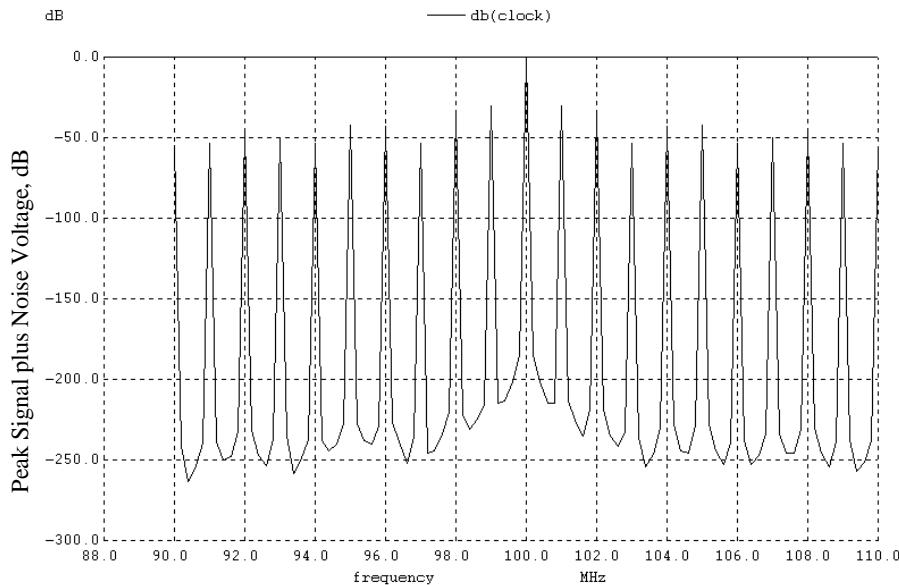


Figure 31.9 Spectrum of a sampling clock signal with noise.

While our simple model will never be capable of generating truly random noise, we can, for a given simulation time, make the simulated spectrum of the oscillator approach something that looks more realistic (and thus more random over a given simulation time). Toward this goal, let's attempt to make the oscillator spectrum more continuous. We can do this by requiring

$$FS = \frac{1}{\text{simulation time}} \quad (31.28)$$

We apply this result in the following example. ■

Example 31.10

Repeat Ex. 31.9 if the FS is set using Eq. (31.28).

Using Eq. (31.28) and a simulation time of 10,000 ns, we get an FS = 100 kHz. Resimulating, using this value of FS, gives the results shown in Fig. 31.10. The amplitude of the square wave varied from 0 to 1.5 V (varied from 0 to A). Remembering that the harmonics of the clock (a square wave) have a value of A/2 at DC and a value of $2A/n\pi$, where $n = 1, 3, 5, \dots$ at the other harmonics, we can calculate the peak amplitude of the fundamental tone in Fig. 31.10 (or Fig. 31.9) as $(2 \cdot 1.5)/\pi = 0.955$ V or -0.4 dB. ■

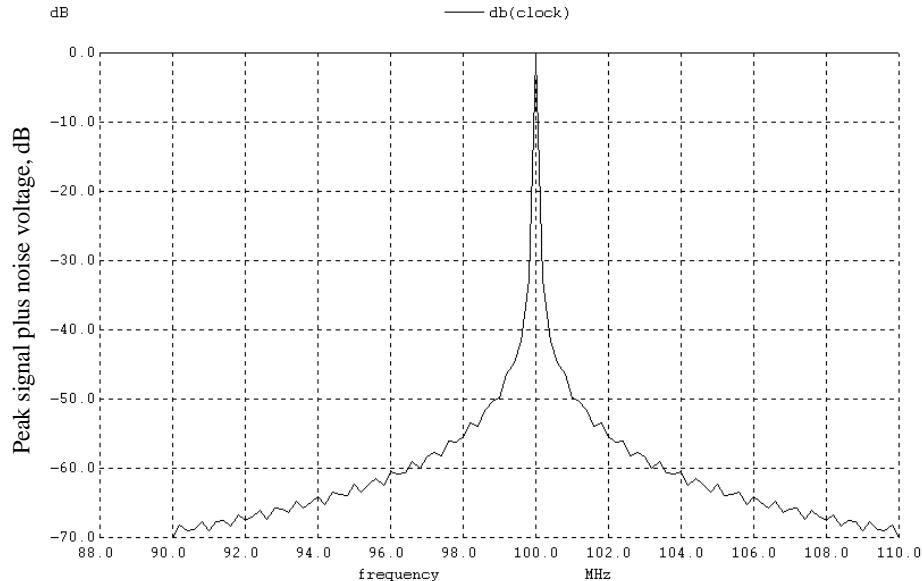


Figure 31.10 Oscillator spectrum using Eq. (31.28) to set phase variation time.

Note that a measured spectrum with anomalous spikes would generally indicate that the noise (the spikes) is not random and could be the result of coupling from an adjacent circuit. The coupling could be through the substrate or power connections, or it

could be capacitive. If the oscillator jitter is only due to MOSFET noise, in general, no unwanted spikes will exist in the oscillator's output spectrum.

In the frequency domain the jitter (which is called *phase noise*) is usually specified at some offset to the fundamental carrier and taken with reference to the carrier. For example, at a 1 MHz offset in Fig. 31.10 (at 99 MHz or 101 MHz), the spectrum has a peak amplitude of approximately -50 dB. The phase noise at a 1 MHz offset would then be given with reference to the carrier as $-50 \text{ dB} - (-0.4 \text{ dB})$ (the carrier amplitude) = -49.6 dBc. However, when talking about phase noise, we are generally referring to a single-tone sinusoid (not a square wave with odd harmonics).

Using Our SPICE Jitter Model

The model we've just developed is difficult to use in a practical simulation because of the finite step time used by SPICE. For example, if we are trying to model the effects of 100 ps of clock jitter on a data converter's performance, then our step size in the simulation should be much smaller than 100 ps. This requirement can lead to very long simulation times or, if the step size is comparable to the simulated jitter, questionable results (see the example below). Nevertheless, the model is useful in many situations.

Example 31.11

Suppose a 100 MHz sampling clock has 500 ps of jitter. Determine how the SNR of an ideal data converter will be affected when clocked with this signal. Assume the topology and input signal of Fig. 31.1 are used.

We begin by using Eq. (31.19) to calculate the number of bits lost

$$N_{Loss} = 8 + 3.33 \log(2\pi \cdot 25\text{MEG} \cdot 500\text{p}) = 4.3 \text{ bits}$$

The SNR of the 8-bit system is determined using Eq. (31.18)

$$\text{SNR} = 6.02(8 - 4.3 - 0.5) + 1.76 = 21 \text{ dB}$$

The next factor that we need to determine is the modulation index, MDI, which simulates 500 ps of jitter. Using Eq. (31.27) we get

$$\text{MDI} = 2\pi \cdot 100\text{MEG} \cdot 500\text{p} = 0.314$$

To keep the simulation time relatively short, we'll simulate for 2,000 ns with a step size of 100 ps. The resolution of the DFT is 1 MHz (set by Eq. [30.35]). The rate at which the jitter varies is set by Eq. (31.27) and is 500 kHz. The DAC output spectrum is shown in Fig. 31.11. The simulated $V_{Qe+N,RMS}$ is 93 mV. The SNR is then given by $20 \cdot \log([0.75/\sqrt{2}] / 93\text{mV}) = 15 \text{ dB}$ (practically worthless). We see the simulated SNR is fully 6 dB below the calculated SNR. We may speculate that this is due to both the jitter we purposely introduced into the clock and the jitter introduced by the varying step size in the SPICE simulation. When we used a pulse source to clock our ADC in previous simulations, the jitter was absent because of the exact timing of the pulse statement and the fact that the rise and fall times were set by the simulation step size. ■

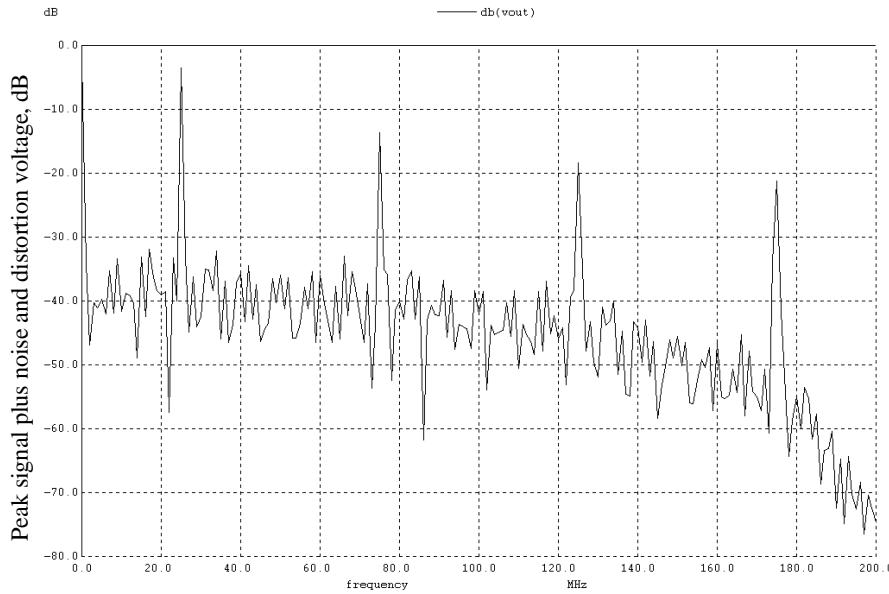


Figure 31.11 DAC output spectrum for Ex. 31.11.

31.1.3 A Tool: The Spectral Density

The observant reader may have noticed, in the last section, that we only discussed the peak-to-peak jitter, ΔT_s , and how it affects the data converter's performance. It is very useful, in many situations, to also have an idea of how the spectrum or spectral characteristics of the data converter's output change as a function of the random sampling jitter or a random variable such as noise. In this section we discuss tools useful in describing the spectrum of a random signal.

The Spectral Density of Deterministic Signals: An Overview

Consider the simple sinewave signal of the form

$$V_{in}(t) = V_p \sin 2\pi f_{in} t \quad (\text{units, V}) \quad (31.29)$$

This signal is termed "deterministic" because the signal has a well-defined shape whether it is continuous or sampled. We can find the average power of this signal, as a function of time $R_{in}(t)$, using the *autocorrelation function* (ACF) for continuous signals given by

$$R_{in}(t) = \lim_{T_0 \rightarrow \infty} \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} V_{in}(\tau) \cdot V_{in}(\tau + t) \cdot d\tau \quad (\text{units, V}^2) \quad (31.30)$$

The average value of Eq. (31.29) as a function of time is then

$$R_{in}(t) = \lim_{T_0 \rightarrow \infty} \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} [V_p \sin 2\pi f_{in} \tau] \cdot [V_p \sin 2\pi f_{in} (\tau + t)] d\tau \quad (31.31)$$

or knowing

$$\sin A \cdot \sin B = \frac{1}{2}[\cos(A - B) - \cos(A + B)] \quad (31.32)$$

we can write

$$V_p^2 \cdot \sin 2\pi f_{in}\tau \cdot \sin 2\pi f_{in}(t+\tau) = \frac{V_p^2}{2}[\cos 2\pi f_{in}t - \cos 2\pi f_{in}(t+2\tau)] \quad (31.33)$$

When we integrate this result, the term $\cos[2\pi f_{in}(t+2\tau)]$ represents a sinusoid with a frequency of $4\pi f_{in}$ (remembering our integration variable is τ) and a phase shift of $2\pi f_{in}t$. Over a long period of time this term averages to zero. Therefore, we can write the average value of Eq. (31.29) as a function of time (the autocorrelation function)

$$R_{in}(t) = \lim_{T_0 \rightarrow \infty} \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} \frac{V_p^2}{2} \cdot \cos 2\pi f_{in}t \cdot d\tau = \frac{V_p^2}{2} \cdot \cos 2\pi f_{in}t \quad (\text{units, V}^2) \quad (31.34)$$

The spectrum of the average value of a function can be found by taking the Fourier transform of the autocorrelation function. The result is called the *power spectral density* function (PSD) and is given by

$$P_{in}(f) = \int_{-\infty}^{\infty} R_{in}(t) \cdot e^{-j2\pi f t} \cdot dt \quad (\text{units, V}^2/\text{Hz or V}^2 \cdot \text{s}) \quad (31.35)$$

The power spectral density function of Eq. (31.29) is then, with the help of Eq. (31.34),

$$P_{in}(f) = \frac{V_p^2}{4} \cdot [\delta(f+f_{in}) + \delta(f-f_{in})] \quad (\text{units, V}^2/\text{Hz}) \quad (31.36)$$

This is simply two impulses in the frequency spectrum located at $\pm f_{in}$ with an amplitude of $V_p^2/4$ (V^2/Hz). The *total average power* of this signal is given by

$$P_{AVG} = \int_{-\infty}^{\infty} P_{in}(f) \cdot df = 2 \cdot \int_0^{\infty} P_{in}(f) \cdot df \quad (\text{units, V}^2/\Omega \text{ or watts}) \quad (31.37)$$

assuming a 1- Ω (normalized) load, which, for Eq. (31.29), is $V_p^2/2$ (V^2).

The *voltage spectral density*, with units of $\text{V}/\sqrt{\text{Hz}}$, is simply the square root of Eq. (31.35) (that is, the square root of the PSD [$= \sqrt{P_{in}(f)}$]). The *root mean square* (RMS) voltage of a signal is given by

$$V_{RMS} = \sqrt{P_{AVG}} = \sqrt{2 \int_0^{\infty} P_{in}(f) \cdot df} = \sqrt{2 \int_0^{\infty} (\text{voltage spectral density})^2 \cdot df} \quad (31.38)$$

The RMS value of Eq. (31.29) is simply, as one would expect for a sinewave, $V_p/\sqrt{2}$. Note the similarity between Eq. (31.38) and Eq. (31.6). The factor of root 2 in Eq. (31.6) is used because $V_{DFT}(f)$, the output of WinSPICE, is the peak voltage at a given (one-sided spectrum) frequency (and so dividing $V_{DFT}[f]$ by $\sqrt{2}$ results in RMS voltages as a function of frequency).

Example 31.12

Determine the ACF, PSD, average power, and RMS value of a signal $V(t)$ made up of three sine waves with peak amplitudes of V_1 , V_2 , and V_3 with frequencies of f_1 , f_2 , and f_3 .

Using Eqs. (31.30) and (31.34), the ACF is

$$R(t) = \frac{V_1^2}{2} \cos 2\pi f_1 t + \frac{V_2^2}{2} \cos 2\pi f_2 t + \frac{V_3^2}{2} \cos 2\pi f_3 t \quad (\text{units, V}^2)$$

The PSD (positive frequencies) is determined using Eqs. (31.35) and (31.36)

$$P(f) = \frac{V_1^2}{4} \cdot \delta(f - f_1) + \frac{V_2^2}{4} \cdot \delta(f - f_2) + \frac{V_3^2}{4} \cdot \delta(f - f_3) \quad (\text{units, V}^2/\text{Hz})$$

The average power, using Eq. (31.37), is

$$P_{AVG} = \frac{V_1^2 + V_2^2 + V_3^2}{2} \quad (\text{units, watts})$$

Finally, the RMS value of the signal is given by

$$V_{RMS} = \sqrt{\frac{V_1^2 + V_2^2 + V_3^2}{2}} \quad (\text{units, V})$$

Note that if we added phase shifts to our signals the results would be the same; the phase shift doesn't change the signal's average value, so we get the same results whether sines or cosines are used in our original spectrum. ■

Next, suppose that the sinewave specified by Eq. (31.29) is sampled at a rate of f_s

$$V_{in}(nT_s) = V_p \sin(2\pi f_{in} \cdot nT_s) \quad (31.39)$$

The ACF for a sampled signal can be written as

$$R_{in}(nT_s) = \lim_{N \rightarrow \infty} \frac{1}{(2N+1)} \sum_{k=-N}^N V_{in}(kT_s) \cdot V_{in}(kT_s + nT_s) \quad (31.40)$$

which results in

$$R_{in}(nT_s) = \frac{V_p^2}{2} \cos 2\pi f_{in} \cdot nT_s \quad (\text{units, V}^2) \quad (31.41)$$

The PSD is the Fourier transform of this equation (see Eq. [30.2] in the last chapter),

$$P_{in}(f) = \frac{V_p^2}{4T_s} \sum_{k=-\infty}^{\infty} [\delta(f - f_{in} + kf_s) + \delta(f + f_{in} + kf_s)] \quad (31.42)$$

The RMS value of the sampled sinewave, Eq. (31.39), assuming we have passed the signal through an ideal reconstruction filter (RCF) with a bandwidth of $f_s/2$, is simply, once again, $V_p/\sqrt{2}$. The PSD of the signal, after passing through the RCF, has an amplitude of $V_p^2/4$ at frequencies of $\pm f_{in}$.

The Spectral Density of Random Signals: An Overview

Let's use our jitter discussion of the last section to illustrate how to look at the spectrum of a random signal. We'll do this in two parts: (1) we'll begin by assuming the jitter is a random variable that falls between two limits and has equal probability of lying anywhere in the region (just as was assumed for the quantization error probability density function when calculating the RMS quantization noise voltage in the last chapter), and (2) then assume the jitter has a Gaussian distribution around some average value (the more practical and realistic situation) and determine how the output of the ADC is affected.

Consider the representations of clock jitter shown in Fig. 31.12. Trace 1 in this figure shows the ideal position of the rising edge of a clock signal. This point is represented on the probability density function (PDF), $\rho(t)$, at time zero. On the next rising edge of the clock, trace 2, the edge is a little too early and is represented on the PDF as shown. We are assuming, probably incorrectly for most practical situations, that the rising edge of the clock is falling within the peak-to-peak boundaries with the equal probability of being in the correct position (as shown in trace 1) or at the edge of a boundary (as shown in trace 4). We also know that the area under the PDF curve in Fig. 31.12 must equal unity, and the average value (also known as the mean or the expected value and denoted by $\langle y \rangle$ or \bar{y}) of a PDF is given by

$$\text{Average value, } \bar{y}, = \int_{-\infty}^{\infty} t \cdot \rho(t) \cdot dt \quad (31.43)$$

Example 31.13

Determine the average value of the jitter with the PDF shown in Fig. 31.12.

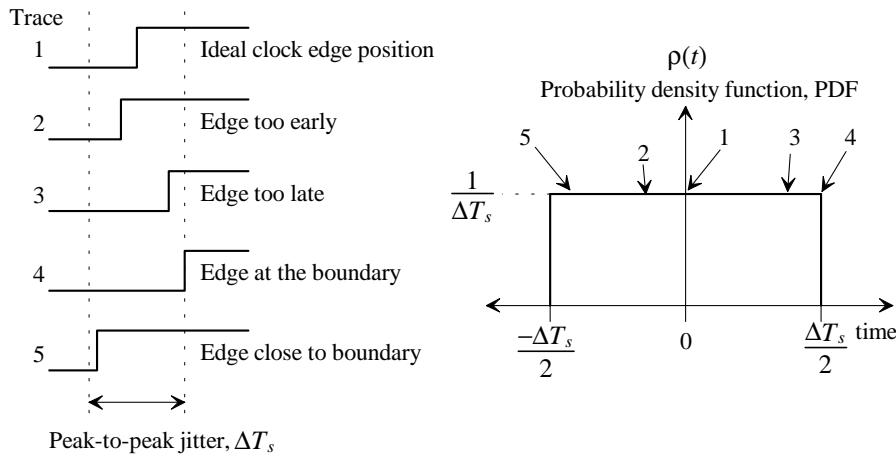


Figure 31.12 Clock jitter assuming the edge falls with the same probability anywhere within the peak-to-peak limits.

We can use Eq. (31.43) to determine the average value of any PDF. Applying this equation to the PDF shown in Fig. 31.12 results in

$$\text{Average value, } \bar{y}, = \int_{-\Delta T_s/2}^{\Delta T_s/2} t \cdot \frac{1}{\Delta T_s} \cdot dt = 0$$

This somewhat obvious result means that the average position of the clock rising edge is the ideal position indicated by trace 1 in Fig. 31.12. Any PDF that is symmetrical about some center point will have an average equal to the center point. ■

The variance of the PDF is defined as the average of the square of the signal's departure from its average value. For a random signal this can be written as

$$\sigma^2 = \overline{(y - \bar{y})^2} = \int_{-\infty}^{\infty} (y - \bar{y})^2 \cdot \rho(y) \cdot dy \quad (31.44)$$

where σ is the standard deviation of the PDF (the square root of Eq. [31.44]). For our purposes, in this book, we can think of variance as the average power of a random (voltage) signal and the standard deviation as the RMS value of the signal (see Eqs. [31.37] and [31.38]). Example random signals include the time difference between the actual edge of a clock and the ideal edge location (jitter), the voltage difference between the input of an ADC and the ADC's reconstructed output (quantization noise), and the random fluctuations of electrons due to thermal motion in a resistor (thermal noise).

Example 31.14

Determine the RMS value of the jitter when the jitter has a probability density function, PDF, as shown in Fig. 31.12.

Using Eq. (31.44) the variance of the jitter PDF is

$$\sigma^2 = \int_{-\Delta T_s/2}^{\Delta T_s/2} t^2 \cdot \frac{1}{\Delta T_s} \cdot dt = \frac{1}{3 \cdot \Delta T_s} \cdot t^3 \Big|_{-\Delta T_s/2}^{\Delta T_s/2} = \frac{(\Delta T_s)^2}{12} \text{ (seconds}^2\text{)}$$

and thus the RMS jitter is

$$\text{RMS jitter, } \sigma = \frac{\Delta T_s}{\sqrt{12}} \text{ (seconds)}$$

where ΔT_s is the peak-to-peak jitter in the sampling clock rising edge. Note the similarity to the derivation of $V_{Qe,RMS}$ in the last chapter. ■

A more useful discussion of jitter can be constructed if we assume the jitter has a Gaussian PDF, as shown in Fig. 31.13, and attempt to describe how the jitter in the sampling clock affects an ADC output spectrum with a single-tone input. Using Eqs. (31.11), (31.12), and (31.15), we can write the sampling error voltage (review Fig. 31.6), at a given time, as

$$\Delta V_s(t) = \delta T_s(t) \cdot V_p \cdot 2\pi f_{in} \cdot \cos 2\pi f_{int} t \quad (31.45)$$

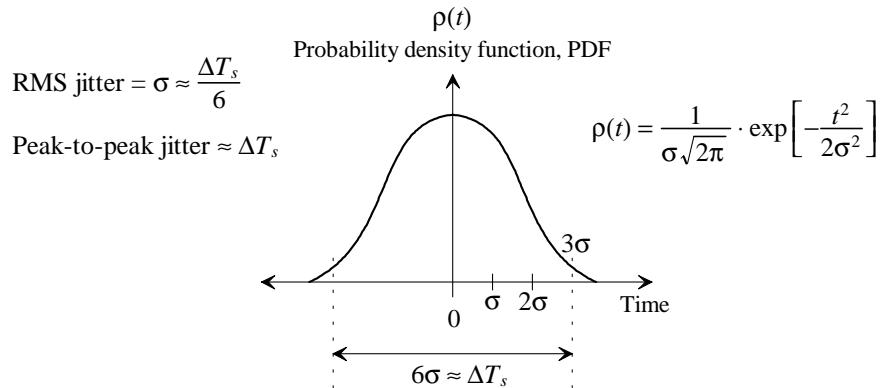


Figure 31.13 Sampling jitter with a Gaussian probability distribution.

where $\delta T_s(t)$ is a random variable indicating the jitter in the sampling clock at a given time. (The variable $\delta T_s(t)$ is the time difference between the actual clock transition time and the expected transition times that are spaced by T_s [see Fig. 31.12].) The peak-to-peak value of $\delta T_s(t)$ is ΔT_s , while its average value is zero. Again, we assume that the jitter probability distribution function is Gaussian, as seen in Fig. 31.13.

Rewriting Eq. (31.45) using a discrete time step nT_s , the sampling error can be written as

$$\Delta V_s(nT_s) = \overbrace{\delta T_s(nT_s) \cdot V_p \cdot 2\pi f_m}^{\text{Sampling error amplitude}} \cdot \overbrace{\cos 2\pi f_m nT_s}^{\text{Carrier term}} \quad (31.46)$$

We're interested in the spectrum of this error signal as it will add to our RMS quantization noise plus distortion voltage, effectively lowering the data converter's SNDR. Notice that the spectrum of Eq. (31.46) will have aliased components (and so will the sampled signal) so we need to filter out these components above $f_s/2$ (with the reconstruction filter.) Also note that multiplying the sampling error by the cosine term in Eq. (31.46) simply shifts the error spectrum to a frequency f_m . The cosine terms acts like a carrier in an amplitude-modulated signal. This is illustrated in Fig. 31.14.

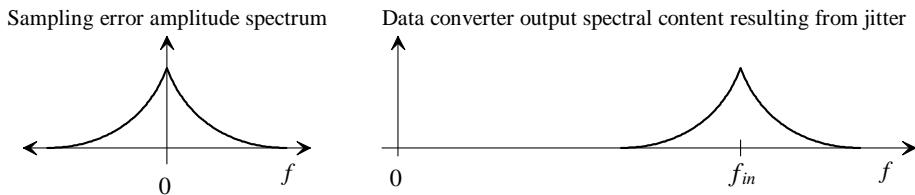


Figure 31.14 Modulating sampling error with an input sinewave frequency.

Example 31.15

Repeat Ex. 31.7 assuming the clock jitter has a Gaussian PDF.

In this example the peak amplitude of the input signal, V_p , is 0.75 V, the input frequency, f_{in} , is 25 MHz, and the peak-to-peak jitter is 100 ps. The average power in the sampling error amplitude spectrum is

$$P_{AVG,jitter} = \sigma^2 \cdot \frac{(V_p \cdot 2\pi f_{in})^2}{2} = \left(\frac{\Delta T_s}{6}\right)^2 \cdot \frac{(V_p \cdot 2\pi f_{in})^2}{2} \quad (31.47)$$

or

$$P_{AVG,jitter} = \left[\frac{100 \text{ ps}}{6}\right]^2 \cdot \frac{(0.75 \cdot 2\pi \cdot 25 \text{ MHz})^2}{2} = 1.93 \times 10^{-6} \text{ V}^2$$

while the RMS voltage associated with this error is 1.39 mV. The quantization noise associated with this 8-bit data converter is

$$V_{Qe,RMS} = \frac{V_{LSB}}{\sqrt{12}} = \frac{V_{REF+} - V_{REF-}}{2^N \sqrt{12}} = 1.69 \text{ mV}$$

The RMS noise voltage due to clock jitter and quantization effects is then given by

$$\sqrt{1.39^2 + 1.69^2} \text{ mV} = 2.1 \text{ mV}$$

We can calculate the SNR using

$$\text{SNR} = 20 \cdot \log \frac{0.75/\sqrt{2}}{2.1 \text{ mV}} = 48.1 \text{ dB}$$

giving an effective number of bits, from Eq. (31.5), equal to 7.7. Note that this is a *significant* improvement over what was calculated in Ex. 31.7, where the jitter variation was always the peak-to-peak value. ■

The PSD of the sampling error amplitude, described by Eq. (31.46), can be determined with the help of Eq. (31.37)

$$\sigma^2 \cdot \frac{(V_p \cdot 2\pi f_{in})^2}{2} = 2 \int_0^\infty P_{jitter}(f) \cdot df \quad (31.48)$$

If the spectrum of the phase noise due to jitter is narrow, as seen in Fig. 31.14, then the spectral density of the sampling error, $P_{jitter}(f)$, is concentrated around the frequency of the input sinusoid. However, if we assume the phase noise spectrum is white and evenly distributed throughout the base spectrum (so that we integrate Eq. [31.48] from DC to $f_s/2$), we can write

$$P_{jitter}(f) = \frac{\sigma^2}{f_s} \cdot \frac{(V_p \cdot 2\pi f_{in})^2}{2} \quad (31.49)$$

The power spectral density of the sampling error voltage, assuming even distribution of the noise throughout the base spectrum, is shown in Fig. 31.15.

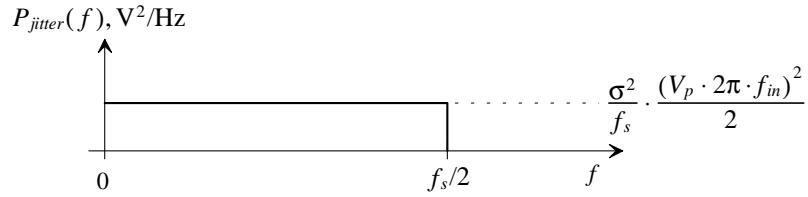


Figure 31.15 Sampling amplitude error PSD assuming sampling error spectrum is white.

Specifying Phase Noise from Measured Data

It's important to note that we have been discussing clock signals that are square waves (that is, have odd order harmonics) and so discussing jitter (a time-domain term) is, generally, more appropriate than discussing phase noise (a frequency domain term). However, because the terms are both widely used to indicate the same, basic, effect (a variation in the period of a periodic waveform), we will briefly discuss phase noise specification from measured oscillator data.

Consider the representation of a measured oscillator spectrum (power spectral density) shown in Fig. 31.16. In general, oscillator noise is specified in terms of the carrier voltage (or power) with units of dBc (decibels with respect to the carrier). The ratio of the power of the fundamental (called the carrier or sampling clock) at f_s is taken to the noise power in a bandwidth at some offset from the fundamental

$$\text{Phase noise, dBc/Hz} = \underbrace{10 \cdot \log \left[\int_{f_{L1}}^{f_{H1}} P_{osc}(f) \cdot df \right]}_{10 \cdot \log(V^2)} - \underbrace{10 \cdot \log P(f_s)}_{10 \cdot \log(P/V^2/\text{Hz})} \quad (31.50)$$

where the first term is the noise power at an offset from f_s .

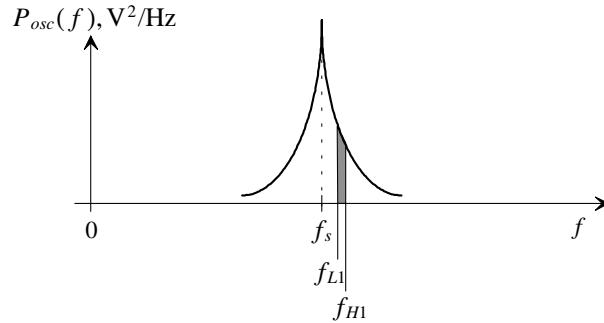


Figure 31.16 Measured oscillator spectrum.

31.2 Improving SNR using Averaging

We first introduced the concept of averaging back in Sec. 30.3.2 to reduce the RMS quantization noise voltage. In this section we'll continue this discussion showing that the SNR of a data converter can be improved by using averaging provided the data converter is linear to within the resolution of the improvement, the input signal is bandlimited, and the input signal is busy (not a DC signal).

31.2.1 Using Averaging to Improve SNR

Recalling that averaging K outputs from a data converter with a busy input results in an RMS quantization noise voltage (see Eq. [30.48]) of

$$V_{Qe,RMS} = \frac{1}{\sqrt{K}} \cdot \frac{V_{LSB}}{\sqrt{12}} \quad (31.51)$$

we can now, with the help of Eq. (31.1), write the ideal signal-to-noise ratio using averaging as

$$\text{SNR}_{ideal} = 6.02N + 1.76 + 10 \log K \quad (31.52)$$

where N is the number of bits (the resolution) of the data converter whose output is being averaged. Using no averaging, that is $K = 1$, results in Eq. (31.52) simplifying to Eq. (31.4). Averaging two samples causes the SNR_{ideal} to increase by 3 dB or the effective resolution of the data converter to increase by 0.5 bits. The increase in resolution due to averaging can be written as

$$\text{Increased resolution, } N_{Inc} = \frac{10 \log K}{6.02} \quad (31.53)$$

Figure 31.17 shows how averaging the output of a data converter changes the effective resolution of the data converter. Note that the increase in resolution is based on the following assumptions: a busy input signal, the input signal is bandlimited, and the data converter is linear to the final resolution (data converter resolution, N , + improvement in resolution, N_{Inc}) coming out of the averaging circuit.

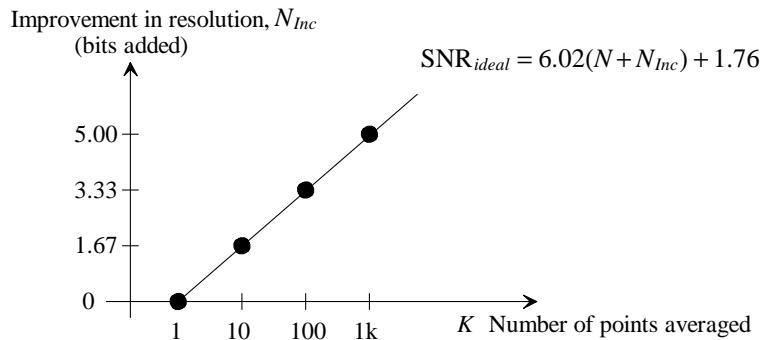


Figure 31.17 Using averaging to improve data converter resolution.

Spectral Density View of Averaging Revisited

We know, from our previous discussions, that if the quantization noise is random, we can determine its spectral density using

$$\sigma = \frac{V_{LSB}}{\sqrt{12}} = \sqrt{2 \int_0^{\infty} P_{Qe}(f) \cdot df} \quad (31.54)$$

where $P_{Qe}(f)$ is the quantization noise power spectral density. Also, from the last chapter we can write

$$P_{Qe}(f) = [V_{Qe}(f)]^2 \quad (31.55)$$

We might think that if the quantization noise is white (Bennett's criteria hold, so there is no correlation from one data converter output sample to the next) then the spectral content of the noise is spread evenly in frequency from zero to infinity ($P_{Qe}[f]$ is a constant with frequency). This would also mean that $P_{Qe}(f)$ approaches zero, from Eq. (31.54), in order to make the average power, that is, the variance (σ^2), of the quantization noise equal to $(V_{LSB})^2/12$.

Before we address this concern (our spectral density approaching zero), let's review how we calculated the RMS quantization noise voltage, $V_{Qe,RMS}$, from the spectrum given in Fig. 30.48 back in Ch. 30. In this figure we looked at the entire spectrum (or most of the spectrum, up to 200 MHz or $2f_s$, where significant spectral content is found) to determine $V_{Qe,RMS}$ (see Ex. 30.11 and the discussion concerning the figure). We know that the quantization noise doesn't experience aliasing since quantization occurs after sampling. So while it is correct to look at a wide spectrum to calculate noise, it would be more useful to limit our view of the spectrum to frequencies up to the Nyquist frequency ($= f_n = f_s/2$), where our desired signal spectrum should reside. We can do this by *assuming* the entire quantization noise power lies in the base spectrum or

$$\frac{V_{LSB}^2}{12} = 2 \int_0^{f_s/2} P_{Qe}(f) \cdot df \quad (31.56)$$

or

$$P_{Qe}(f) = \frac{1}{f_s} \cdot \frac{V_{LSB}^2}{12} \quad (31.57)$$

The PSD of the quantization noise is plotted in Fig. 31.18. Note the similarity to Fig. 30.57 (the voltage spectral density of the quantization error).

Consider the result of adding two consecutive ADC outputs as shown in Fig. 31.19. A simple sum will be considered the average of the two consecutive ADC output signals. The finite digital output word length, in this case 8 bits, can limit the resolution of the resulting sum. In the cases where we do need to do a division by two we could simply use the top eight bits of the sum (a shift-right operation). In most of the discussions related to digital words in this book, *averaging will be equivalent to addition*. The current

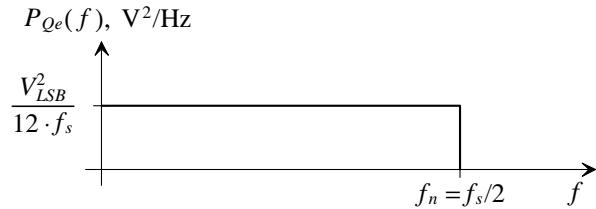


Figure 31.18 Quantization noise power spectral density.

time sample coming out of the ADC is labeled $x(nT_s)$, while the previous ADC output is $x[(n-1)T_s]$. The output of the simple digital averager is

$$y(nT_s) = x(nT_s) + x[(n-1)T_s] \quad (31.58)$$

remembering that Bennett's criteria must be valid for averaging to effectively reduce the quantization noise. For example, applying a DC input signal to the circuit of Fig. 31.19 will not result in higher accuracy (the output of the averager will remain the same as the output of the ADC [actually the averaged output is twice the ADC output]). We'll discuss this restriction in more detail in a moment when we discuss adding a dither or pseudo random noise signal to the input to randomize the quantization noise (make its spectrum white). Also note that there are restrictions on the allowable range of input frequencies when using this configuration to avoid amplitude distortion. For example, if f_{in} is 50 MHz (with $f_s = 100$ MHz), then it's easy to show that the resulting digital averager output is zero (see Fig. 31.20).

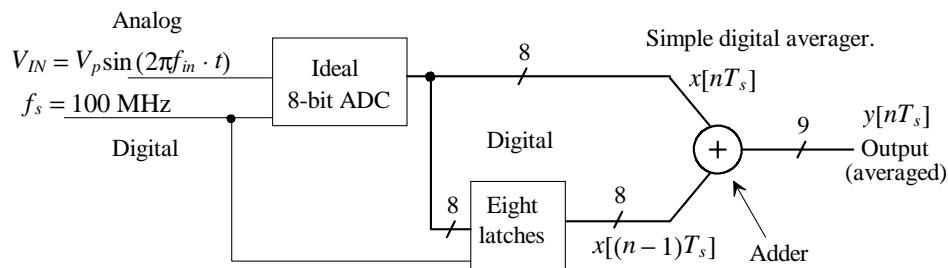


Figure 31.19 Using two paths to average the quantization noise.

Let's show that the digital averager of Fig. 31.19 can be thought of as a filter and look at how passing the ADC output through the averager affects the ADC's signal plus quantization noise and distortion output spectrum. This will also tell us how we have to restrict the input frequencies applied to the ADC to avoid amplitude distortion or something similar to what's shown in Fig. 31.20.

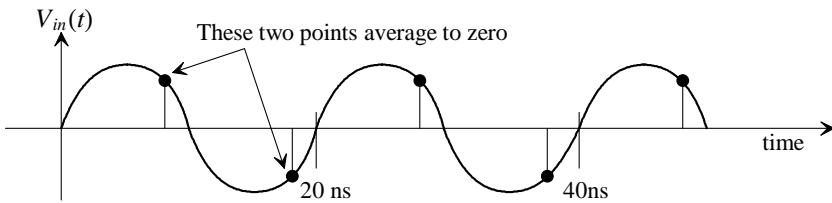


Figure 31.20 The limitations placed on ADC input frequency when using averaging.

Consider the redrawn (Fig. 31.21) z-domain version of the digital averaging filter of Fig. 31.19. The filter's transfer function can be found directly from Fig. 31.21 or by taking the z-transform of Eq. (31.58) as

$$H(z) = \frac{Y(z)}{X(z)} = 1 + z^{-1} \quad (31.59)$$

Remembering from Eq. (30.12) that $z = e^{j2\pi f_s t}$ we can write

$$H(z) = 1 + e^{-j2\pi \frac{f_{in}}{f_s}} = \underbrace{1 + \cos\left(-2\pi \frac{f_{in}}{f_s}\right)}_{\text{real}} + j \cdot \underbrace{\sin\left(-2\pi \frac{f_{in}}{f_s}\right)}_{\text{imaginary}} \quad (31.60)$$

Taking the magnitude of this equation results in

$$|H(z)| = \sqrt{\left(1 + \cos\left[2\pi \frac{f_{in}}{f_s}\right]\right)^2 + \left(\sin\left[2\pi \frac{f_{in}}{f_s}\right]\right)^2} \quad (31.61)$$

or simplifying and changing the notation to $H(f)$ with $f = f_{in}$

$$|H(f)| = \sqrt{2\left(1 + \cos\left[2\pi \frac{f}{f_s}\right]\right)} \quad (31.62)$$

and the phase is given by

$$\angle H(f) = \tan^{-1} \left[\frac{-\sin\left(2\pi \frac{f}{f_s}\right)}{1 + \cos\left(2\pi \frac{f}{f_s}\right)} \right] \quad (31.63)$$

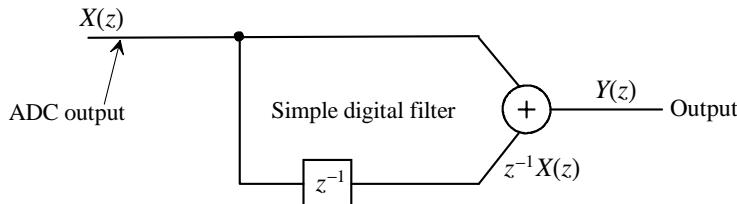


Figure 31.21 Z-Domain representation of the averager shown in Fig. 31.19.

Referring to Ex. 30.4 of the last chapter the phase can be written as

$$\angle H(f) = -\pi \cdot \frac{f}{f_s} \text{ (units, radians) for } f < f_s/2 \quad (31.64)$$

The magnitude and phase responses of this simple digital filter are shown in Fig. 31.22. Note that this is the discrete version of the comb filter discussed in Ex. 30.4. Also note that (1) the phase response is linear, (2) the response is periodic (as is the response of any digital filter), and (3) at an input frequency of half the Nyquist frequency, $f_s/4$, the magnitude response is $\sqrt{2}$ (3 dB down from the DC gain of two).

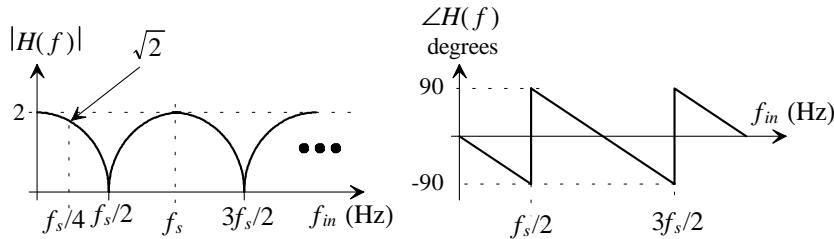


Figure 31.22 Magnitude and phase response for the simple digital filter of Fig. 31.21.

We can also see that (1) averaging results in an attenuation of many of the input signal frequencies (as shown in Fig. 31.22) and (2) indeed the average of the input signal goes to zero, as was shown in Fig. 31.20, when the input signal frequency is $f_s/2$.

If we assume the output quantization noise power spectral density, $P_{Qe}(f)$, for the ADC shown in Fig. 31.19 is white then the output of the simple digital filter has the PSD shown in Fig. 31.23 (the product of the filter response squared with the noise PSD). The average power contained in this PSD is

$$P_{AVG} = 2 \int_0^{f_s/2} \frac{V_{LSB}^2}{12 \cdot f_s} \cdot 2 \left(1 + \cos \left[2\pi \frac{f}{f_s} \right] \right) df \quad (31.65)$$

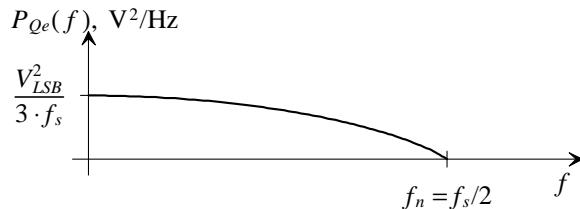


Figure 31.23 Quantization noise power spectral density after averaging two samples.

or

$$P_{AVG} = \frac{V_{LSB}^2}{6} + \frac{V_{LSB}^2}{6\pi} \left[\sin 2\pi \frac{f}{f_s} \right]_{f=0}^{f=f_s/2} = \frac{V_{LSB}^2}{6} \quad (31.66)$$

The power in an input sinewave before averaging is $V_p^2/2$ (the RMS voltage of the sinewave is $V_p/\sqrt{2}$). Averaging (adding) two samples results in an increase in the desired signal amplitude by two and so the power increases to $2V_p^2$ (the RMS voltage increases to $(2V_p)/\sqrt{2}$). This is important because now the SNR, on the output of the digital averaging filter, is

$$\text{SNR} = 20 \log \frac{2V_p/\sqrt{2}}{V_{LSB}/\sqrt{6}} = 20 \log \frac{V_p/\sqrt{2}}{V_{LSB}/\sqrt{24}} \quad (31.67)$$

or in terms of a generic averaging constant K (see Eqs. [31.51] or [30.48]), the effective RMS quantization noise voltage is

$$V_{Qe,RMS} = \frac{1}{\sqrt{K}} \cdot \frac{V_{LSB}}{\sqrt{12}} \quad (31.68).$$

Without rederiving the equations presented at the beginning of this section, we should see how averaging affects a data converter's SNR.

An Important Observation

Equation (31.68) assumes the averaging filter does not attenuate the input signal. If, for example, the input frequency were $f_s/4$, then the RMS amplitude of the desired signal would change from $(2V_p)/\sqrt{2}$ to $(V_p \cdot \sqrt{2})/\sqrt{2}$ or simply V_p (because of the root two gain at $f_s/4$, as shown in Fig. 31.22) and the SNR would be the same as the output of the ADC in the nonaveraged circuit. If the input frequency were greater than $f_s/4$, then the SNR would actually be worse than the nonaveraged SNR! Therefore, we have to restrict the input frequency bandwidth, B , to frequencies less than $f_s/4$ when averaging two terms in order to avoid degrading the data converter's SNR. In general, for an arbitrary number of averages K , we can write the restrictions on the input bandwidth using

$$B = \frac{f_s/2}{K} = \frac{f_n}{K} \quad \text{and} \quad f_{in} \leq B \quad (31.69)$$

We have already presented this equation (Eq. [31.22]) when discussing how oversampling affects sampling clock jitter stability requirements. The averaging factor K is commonly called the *oversampling ratio* to denote the ratio of the Nyquist frequency to the input signal bandwidth. This can sometimes be confusing since, as we showed in the last chapter, oversampling an input waveform alone, without averaging, does not lower the amount of quantization noise in a data converter's output spectrum. Nevertheless, stating that a data converter is using oversampling is synonymous with stating the data converter employs an averaging filter. The averaging filter used on the output of an ADC is called a

decimating filter while the reverse averaging filter used on the input of a DAC is called an *interpolating filter*. We will discuss these filters in detail in the next sections.

Example 31.16

Suppose the input sinewave in Fig. 31.19 has a peak amplitude of 0.5 V and a frequency of 20 MHz. Determine the peak amplitude of the averager output and the delay through the circuit. Comment on any assumptions made.

Using Eq. (31.62) we get

$$|H(f)| = \sqrt{2\left(1 + \cos\left[2\pi \frac{20}{100}\right]\right)} = 1.62$$

and so the peak amplitude of the output sinewave is $0.5 \cdot 1.62 = 809$ mV. Ideally, the amplitude out of the averager is twice the input or, in this case, 1 V.

The delay through the filter is determined using Eq. (31.64) and knowing

$$\overbrace{2\pi \cdot f \cdot \Delta t}^{\text{Phase shift}} = -\pi \frac{f}{f_s}$$

The constant delay (knowing the minus sign indicates the output of the filter occurs after, or later in time than, the input signal) can then be written as

$$\Delta t = \frac{1}{2f_s} = \frac{T_s}{2} \quad (31.70)$$

and so, for this example, $\Delta t = 5$ ns.

Note that we are not discussing the effects of quantization noise, that is, the fundamental minimum voltage that can be resolved. We are assuming continuous amplitude signals throughout the system in order to simplify the filter calculations. This assumption falls apart if, for example, the peak-to-peak amplitude of the input sinewave is reduced to a value below one least significant bit. This will cause the circuit to function as if the input were a DC signal. ■

Jitter and Averaging

We can apply the averaging discussion just developed directly to the jitter discussion presented earlier in the chapter and answer the question, "How does averaging affect the sampling amplitude error power (resulting from jitter) in a data conversion system?" If we assume that the jitter has a Gaussian PDF, then the average power in the sampling error amplitude, from Ex. 31.15, is

$$P_{AVG,jitter} = \left[\sigma \cdot \frac{V_p}{\sqrt{2}} \cdot 2\pi f_{in} \right]^2 \quad (31.71)$$

where σ is the standard deviation of the jitter (see Fig. 31.13). It may be helpful to rewrite Eq. (31.68) in terms of the quantization error power as

$$P_{Qe,AVG} = (V_{Qe,RMS})^2 = \frac{1}{K} \cdot \frac{V_{LSB}^2}{12} \quad (31.72)$$

and apply the same derivation to Eq. (31.71) to give

$$P_{AVG,jitter} = \frac{1}{K} \cdot \left[\sigma \cdot \frac{V_p}{\sqrt{2}} \cdot 2\pi f_{in} \right]^2 \quad (31.73)$$

This equation shows that the sampling error amplitude power, $P_{AVG,jitter}$, introduced into the data converter's output spectrum decreases with averaging. Averaging two samples causes the sampling error amplitude power to decrease by 3 dB. This effectively reduces the jitter requirements placed on the sampling clock. While this may not appear to be very significant at first glance, consider what happens if, for example, 256 samples are averaged ($K = 256$). The sampling error power decreases by 24 dB, making clock jitter, when using a reasonably stable oscillator, almost not an issue. Also note that a doubling in the jitter's standard deviation, σ , results in a 6 dB increase in sampling error amplitude power.

Relaxed Requirements Placed on the Antialiasing Filter

The use of averaging will also lead to relaxed requirements of the antialiasing filter (AAF). Figure 31.24a shows the requirements placed on the AAF without averaging. As we saw in the last chapter, ideally, the transition from the 3 dB frequency to the "stop frequency" or Nyquist frequency should be infinitely sharp (the filter should abruptly change from a gain of unity to a gain of zero [something small]). When using averaging, Fig. 31.24b, we have to limit our desired input signal bandwidth to B ; see Eq. (31.69). The rolloff of the filter in part (b) of the figure can be much more gradual and in many cases a simple, single pole, RC filter is all that's needed for an AAF. Also, our averaging filter will attenuate the ADC output spectrum, as seen in Fig. 31.22, and help to remove input signal power above $f_s/2K$. The significance of this will be easier to see as the number of points averaged increases and our averaging filter's response gets sharper with more attenuation (as discussed in the next section). Of course, the penalty for the relaxed requirements of the AAF is reduced signal bandwidth for a fixed sampling frequency.

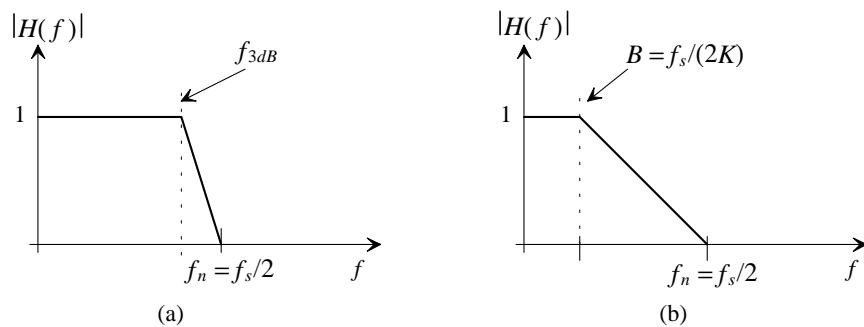


Figure 31.24 (a) AAF requirements without averaging, and (b) AAF requirements with averaging.

Data Converter Linearity Requirements

Consider the cases for averaging ADC outputs shown in Fig. 31.25. In part (a) we show the ideal situation where the black dots indicate two consecutive outputs spaced by one LSB (time is not shown in this figure). The ADC outputs in part (a) are located on the ideal levels, while the averaged output falls exactly in the middle of these levels (and hence our increased resolution).

Part (b) of this figure shows the situation where the ADC outputs are shifted downwards by 0.5 LSBs from their ideal levels. Following this offset, the averaged point shifts downwards as well. In part (c) the top output of the ADC (the top black dot) is shifted downwards by 0.5 LSBs and so the averaged point shows a 0.25 LSB offset from its ideal position. While we used a single LSB difference to show averaging, we could use any number of LSBs to show that the ADC accuracy must be equal to or better than the desired final digital filter output accuracy.

The number of bits in the ADC (its resolution) N , and the number of bits improvement in resolution after filtering, N_{inc} , are used with the final, total number of bits (the number of bits coming out of the digital filter) to give

$$N_{Final} = N + N_{Inc} \quad (31.74)$$

The ADC output should ideally change in increments of the exact LSB voltage. In reality, the changes will be different from the ideal output levels (as just discussed). In order to achieve an increase in the number of final bits, the output of the ADC must be accurate (its actual levels must be spaced from the ideal levels) to within

$$\pm \frac{V_{REF+} - V_{REF-}}{2^{N_{Final}+1}} = \pm (0.5 \text{ LSB}) \cdot \frac{1}{2^{N_{Inc}}} \quad (31.75)$$

where no averaging ($N_{inc} = 0$ and $K = 1$) means the ADC is at least 0.5 LSBs accurate. This is a *significant limitation* when using averaging to increase the resolution of an ADC.

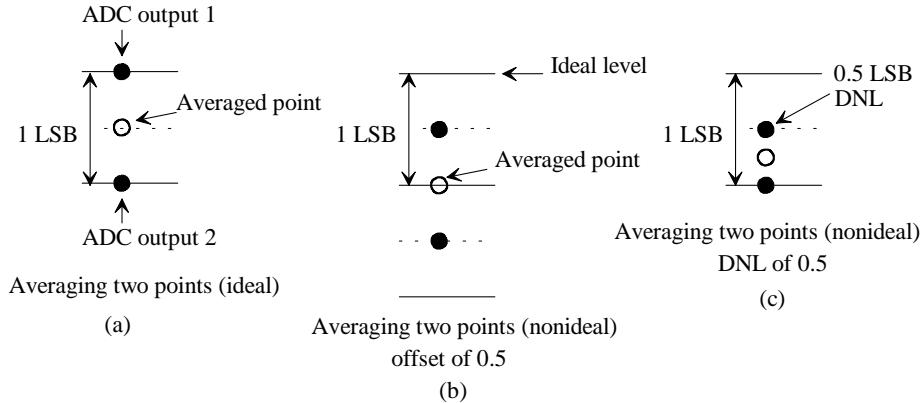


Figure 31.25 Linearity requirements when averaging.

This is especially true when a resolution greater than 10 bits is desired with INL and DNL less than ± 0.5 LSBs. Later in the chapter, and in the next chapter, we will look at feedback topologies that may relax the accuracy requirements placed on the ADC and allow averaging to more effectively remove quantization noise.

Example 31.17

To illustrate the requirements placed on the accuracy of the original ADC in more detail, consider averaging 16 consecutive ADC output samples: 15 at a digital code of zero and one at a digital code of 1 LSB. Determine the accuracy required of the ADC, the size of the word coming out of the averaging filter assuming the ADC is 8 bits, and the final word size after considering the increase in resolution.

The ideal output of the digital filter will be 1/16 of the original ADC's LSB. We can write this as

$$\text{Average} = \frac{\overbrace{1+0+0+\dots+0}^{\text{16 consecutive ADC outputs}}}{16} \cdot (\text{Original LSBs})$$

where 1 is a data converter output of 00000001 and 0 is 00000000. Averaging 16 sample points from Eq. (31.53) will give an increase in resolution of 2 bits. (Getting only a 2-bit increase may be tough to see using only a single LSB difference in the ADC outputs but may be easier to see if the codes are different by several LSBs.) Because the ADC is 8 bits, the final resolution will be 10 bits. (Note that an increase in resolution of 1.5 bits would also require a 10-bit word.) However, if we add 16 8-bit words (where the value of the words, in the general case, can range from 00000000 to 11111111), we end up with a 12-bit word size. To get to our desired 10-bit word size we throw out the lower 2 bits. Another way of stating this is that we divide the 12-bit word by four (shift right two times to remove the lower two bits in the word). Here, where only one ADC output of the 16 is 00000001, the output of the averaging filter would end up being 0000000000. We would need to see at least four ADC outputs of 00000001 in order to see the 10-bit filter output go to 0000000001. If the input is busy, we will unlikely have the case where only one of the 16 filter inputs is 00000001 and the other inputs are 00000000. (More on this below in the dither discussion.)

The INL and DNL of the ADC must be less than ± 0.125 LSBs (8-bit LSB), from Eq. (31.75), in order for the output of the filter to be ± 0.5 LSBs accurate (a 10-bit LSB).

Note that an ADC output of all ones, that is, 11111111 (255) corresponds to an analog voltage of $VDD - 1$ LSB where $1 \text{ LSB} = (V_{REF+} - V_{REF-})/256$ ($VDD - 1 \text{ LSB} = 255 \cdot 1 \text{ LSB}$). The maximum output of the filter, after averaging 16 ADC output samples, is $(16 \cdot 255)/4 = 111111100$ (1020) or, in terms of a voltage, $1020 \cdot 1 \text{ LSB}$ where $1 \text{ LSB} = (V_{REF+} - V_{REF-})/1024$. In both cases an output of all zeroes corresponds to V_{REF-} and, also in both cases, the maximum analog output voltage is $V_{out} = 0.9961 \cdot (V_{REF+} - V_{REF-})$. ■

Example 31.18

Specify the accuracy required of an 8-bit ADC if it is to be used with oversampling to attain 12 bits with INL and DNL of ± 0.5 LSBs.

The increase in the number of bits, N_{mc} , is 4. The accuracy required of the 8-bit ADC, from Eq. (31.75), is $\pm(1/32)$ of an LSB. If $V_{REF+}=1.5$ V and $V_{REF-}=0$ then the LSB of the ADC is 5.86 mV. The output of the ADC must be within $\pm 183 \mu\text{V}$ of the ideal ADC output levels in order to arrive at a final, after averaging, resolution of 12 bits (with a 12-bit accuracy of ± 0.5 LSBs). Also, according to Eq. (31.53), we will have to average 256 consecutive ADC outputs to get a 12-bit output. ■

Adding a Noise Dither to the ADC Input

Our assumption, when discussing the benefits of averaging or calculating the spectral density of the quantization noise, falls apart for DC or slow-moving signals (the ADC input is not "busy"). To solve this problem consider adding a noise signal to the ADC input that has a frequency content that falls within the range

$$\frac{f_s}{2K} \leq f < \frac{f_s}{2} \quad (31.76)$$

so that it can be filtered out with the averaging filter. This noise is often called *dither* (a state of indecision or agitation) because it helps to randomize the spectral content of the quantization noise making it white (a flat spectrum, see Fig. 31.18).

Figure 31.26 shows the basic idea. In part (a) a DC signal is applied to the ADC that falls halfway between two ADC transition codes spaced apart by 1 LSB. The output code of the ADC remains unchanged with time. In part (b) a noise signal is added to the DC input which has two benefits: (1) the quantization noise (the difference between the input signal and the reconstructed ADC output code) changes with time, and (2) the output of the ADC has some variation which makes it possible to determine the DC voltage after averaging.

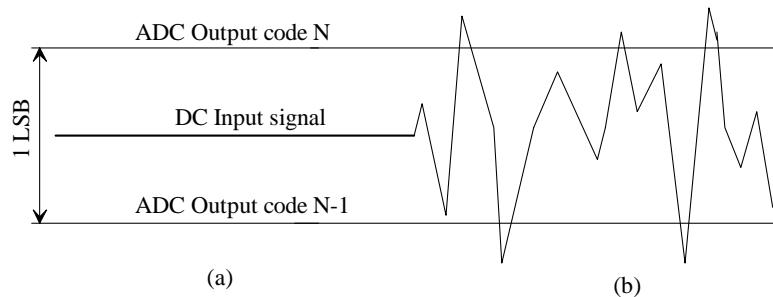


Figure 31.26 (a) DC input signal and (b) DC input signal with dither added.

We can add the noise signal to our desired input signal with a circuit similar to what's shown in Fig. 31.27. Simple resistors add and reduce the noise signal to the ADC input. The noise signal source is, most easily, derived from some sort of asynchronous logic circuit and has a peak amplitude (before reduction) of VDD ($= 1.5$ V in this chapter). In this figure note that we have indicated that the dither signal amplitude should be approximately 0.5 LSB RMS (remembering the signal is, ideally, random and bandlimited as specified by Eq. [31.76]). This number, 0.5 LSB RMS, is subjective, and no exact rules as to its selection can be given other than the desire that the peak-to-peak amplitude be greater than 1 LSB. One *disadvantage* of adding the dither is that the allowable range of input signals shrinks (a DC signal at $VDD - 1$ LSB will not benefit from dithering since the ADC will be at its full-scale output).

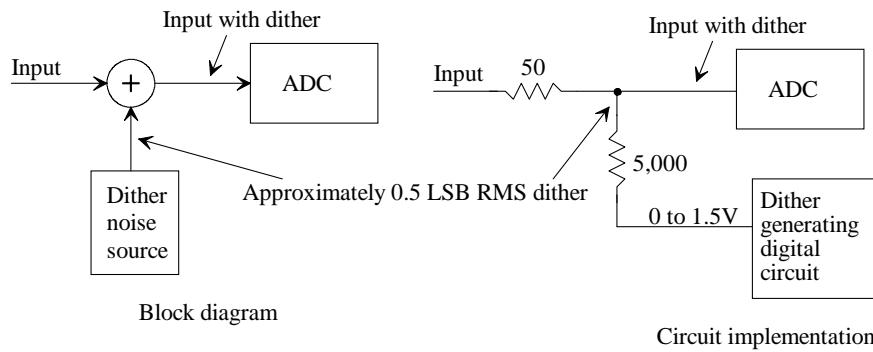


Figure 31.27 Adding dither to an ADC input signal.

Before we discuss the implementation of a dither source, consider one possibility (a Gaussian PDF) for the desired probability density function (PDF) of the dither signal and DC input shown in Fig. 31.28 (the input to the ADC). If we average this signal over a long time, we get the average or DC input signal since the dither averages to zero. This

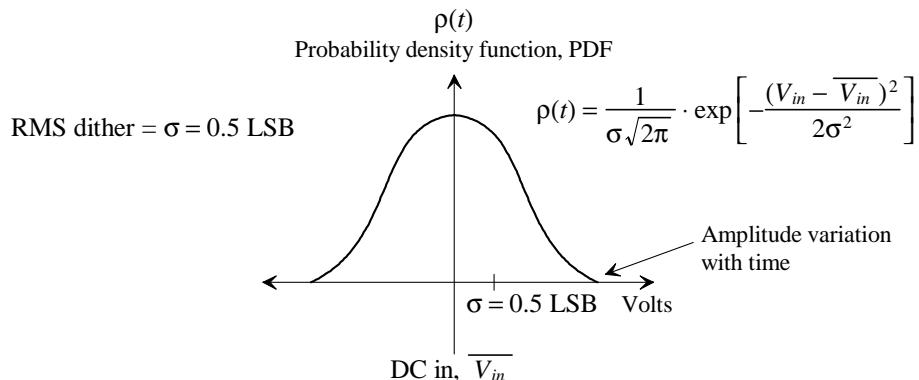


Figure 31.28 Input to the ADC, dither and DC, with a Gaussian probability distribution.

would also mean that we can have some dither spectral content below $f_s/2K$ as long as we average enough ADC output samples to make its contribution to the SNDR small. It is generally a good idea to use Eq. (31.76) as a guide for allowable dither spectral content. Finally, it's important that any dither signal we generate has a symmetrical PDF (the dither signal must average to $VDD/2$ before amplitude reduction). If not, an unknown DC offset (the known DC offset is the $VDD/2$ attenuated by the resistive divider in Fig. 31.27) in the data converter's (actually the filter's) output will result.

An example of an implementation of a dither noise source is shown in Fig. 31.29. The outputs of the rows of inverters, which are tied together, will occur asynchronously and fight against each other causing the amplitude of the dither signal to occupy levels other than the normal logic levels of VDD and ground for significant amounts of time. The dither signal can be made more random by adding more rows of inverters. The challenge to this design is setting the number of inverters used in each row so that the spectral content falls within the desired range (which may require a large number of inverters) and keeping the output of the dither circuit uncorrelated with the sampling clock. Other techniques for generating random noise, such as using linear serial feedback registers, can be found in most books covering communication systems.

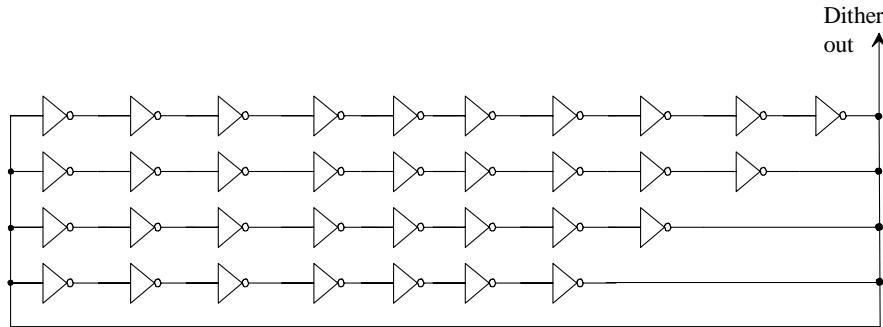


Figure 31.29 One possible implementation of a dither circuit.

The Z-plane

It will be very helpful in our discussion of mixed-signal circuits and systems to gain an intuitive feel for the frequency response of a discrete-time system by looking at the z-domain representation of the system. Toward this goal, consider the transfer function of the simple digital averager depicted in Figs. 31.19 and 31.21 with a z-transform of

$$H(z) = \frac{Y(z)}{X(z)} = 1 + z^{-1} = \frac{z+1}{z} \quad (31.77)$$

$Y(z)$ is the system's output, while $X(z)$ is the system's input. Note that the system is discrete in time but not necessarily in amplitude. We can apply the z-transform to switched-capacitor circuits with continuous-valued amplitudes as well as to data converters with quantized values of amplitude. It is very useful, for an intuitive

understanding of the frequency response of a discrete system, to plot the transfer function in the z-plane (Fig. 31.30). Figure 31.30 also shows how Eq. (31.77) can be displayed on the z-plane. A pole is located at $z = 0$ (at the location the denominator goes to zero and the transfer function goes to infinity) and a zero is located at $z = -1$ (at the location where the numerator goes to zero).

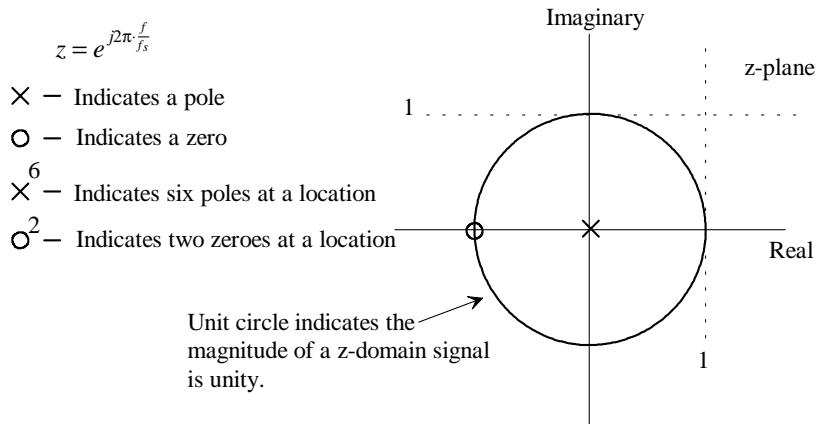


Figure 31.30 The z-plane.

The z-plane is usually used to describe the frequency response of a discrete time system, $H(f)$, by assuming the input to the system is a unit magnitude sinusoid with varying frequency, f . This input, $1 \cdot e^{j2\pi \frac{f}{f_s}} (= z)$, evaluates the output of the system or

$$H(f) = H(z) \text{ evaluated when } z = \underbrace{1}_{\text{magnitude}} \cdot \underbrace{e^{j2\pi \frac{f}{f_s}}}_{\text{phase}} \quad (31.78)$$

We should now see that the unit circle, shown in Fig. 31.30, indicates the relationship between z and f when specified by Eq. (31.78). Therefore, to determine $H(f)$ from a plot of $H(z)$ on the z-plane, we simply evaluate $H(z)$ along the unit circle. To show how this transfer function evaluation is performed, consider Eq. (31.77) and the corresponding plot of its pole and zero shown in Fig. 31.30 along with the magnitude of Eq. (31.77) or Eq. (31.59) plotted against the frequency in Fig. 31.22. At DC ($f = 0$ and $z = 1 \cdot e^0 = 1 \angle 0$) point A in Fig. 31.31, the gain of the circuit is two and is calculated using

$$|H(f)| = \frac{\text{distance to zero}}{\text{distance to pole}} \quad (31.79)$$

The distance from the zero to point A is 2 while the distance between the pole to point A is 1. Therefore, as shown in Fig. 31.22, the magnitude of $H(f)$ is 2. The phase of the transfer function is calculated along the positive x-axis using

$$\angle H(f) = \angle \text{ of zero} - \angle \text{ of pole} \quad (31.80)$$

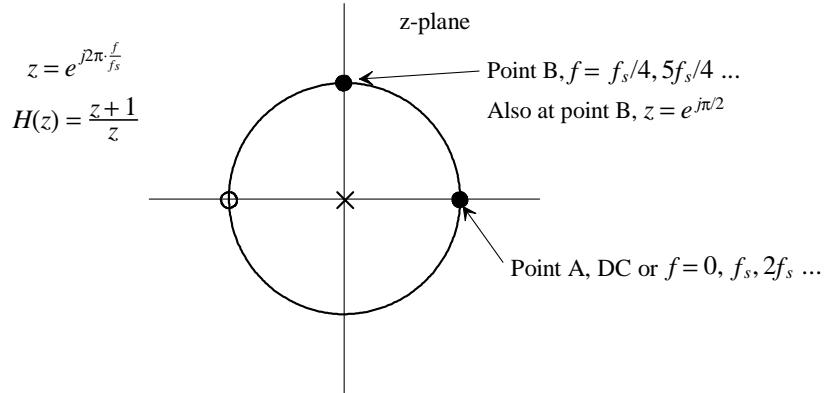


Figure 31.31 The z-plane pole and zero for Eq. (31.77).

which, as seen in Fig. 31.22, results in a phase angle of zero. Next consider evaluating the $H(z)$ at $f_s/4$ ($f = f_s/4$ and $z = 1 \cdot e^{j\frac{\pi}{2}} = 1 \angle 90^\circ$), point B in Fig. 31.31. The distance from the pole to point B is 1 while the distance from the zero is $\sqrt{2}$ resulting in a magnitude $\sqrt{2}$. The angle from the pole along the x-axis to point B is 90° , while the angle from the zero is 45° resulting in an overall phase response of -45° (verify with Fig. 31.22).

Also note that (1) any digital filter's or system's frequency response is periodic with period f_s (one complete revolution around the unit circle), (2) we normally are only concerned with evaluating $H(z)$ over the top half of the unit circle (from DC to $f_s/2$ [the Nyquist frequency, f_n]), and (3) a pole at the origin has no effect on the magnitude response of $H(z)$ but does affect the phase response (as shown in Ex. 30.6 in which multiplying $H(z)$ by z^{-1} , adding a pole at the origin to $H(z)$, simply shifts the output later in time). Finally note that the number of poles in $H(z)$ must be greater than or equal to the number of zeroes if the digital filter/system is to be realizable in hardware (the output of the system cannot occur before the system's input).

Example 31.19

Determine, using the graphical approach just discussed, the magnitude and phase of the transfer function shown in Fig. 31.32 at a frequency of $f_s/4$.

If we label the length from a pole (zero) to the evaluation point $p(z)$, then the magnitude of the transfer function is given by

$$|H(z)| = \frac{z_1}{p_1 \cdot p_2}$$

Labeling the angles for the poles and zeroes as indicated in the figure, we can write the phase response as

$$\angle H(z) = \theta_3 - \theta_2 - \theta_1 \blacksquare$$

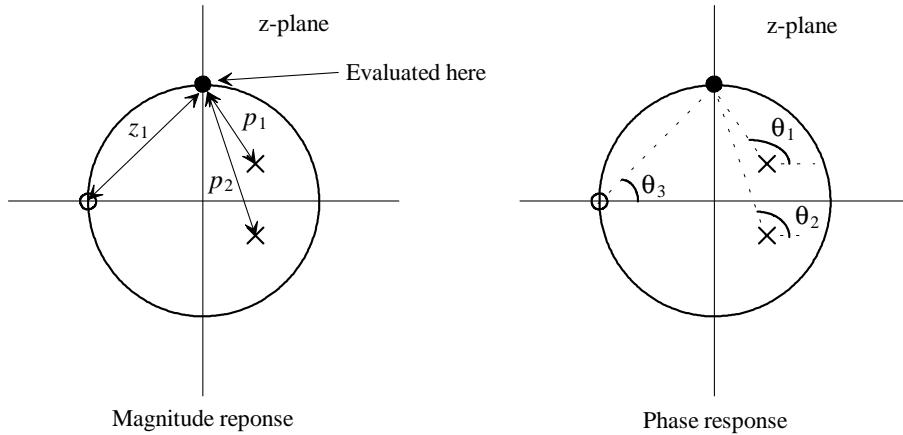


Figure 31.32 The z-plane pole and zero plot for Ex. 31.19.

Example 31.20

Determine the frequency response of a digital system with the time domain response

$$y[nT_s] = x[nT_s] + y[(n-1)T_s]$$

Sketch the hardware implementation of the system and its frequency response.

The z-domain transfer function for this system is

$$Y(z) = X(z) + Y(z) \cdot z^{-1}$$

or

$$H(z) = \frac{1}{1 - z^{-1}} = \frac{z}{z - 1} \quad (31.81)$$

The hardware implementation of the system is shown in Fig. 31.33 along with the z-domain representation. Note that the size of the words used (the number of bits coming out of the adder and the number of latches) depends on the application.

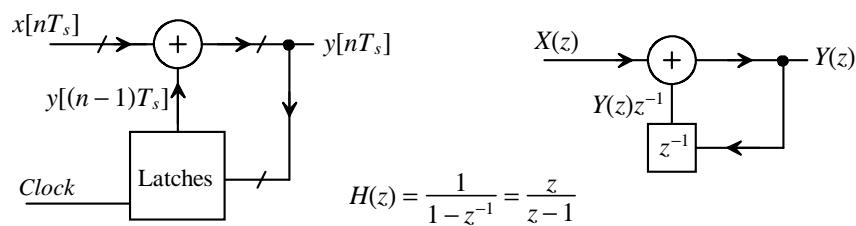


Figure 31.33 Block diagram of a digital integrator used in Ex. 31.20.

Figure 31.34 shows the z-plane representation of this system along with the magnitude and phase response of the system. This circuit is called a digital integrator. To show why, let's determine the magnitude and phase responses, using Eq. (31.81), and noting the z in the numerator is simply a phase shift

$$H(z) = \frac{z}{z-1} \xrightarrow{\text{Phase shift}} e^{\overbrace{j2\pi f_{fs}}^{\text{From zero}}} \cdot \frac{1}{e^{j2\pi f_{fs}} - 1} = e^{j2\pi f_{fs}} \cdot \frac{1}{(-1 + \cos 2\pi f_{fs}) + j \sin 2\pi f_{fs}} \quad (31.82)$$

Knowing

$$\left| \frac{1}{a+jb} \right| = \frac{1}{\sqrt{a^2+b^2}} \quad (31.83)$$

we can write (see Eq. [31.61])

$$|H(f)| = \frac{1}{\sqrt{\left(-1 + \cos 2\pi \frac{f}{f_s}\right)^2 + \left(\sin 2\pi \frac{f}{f_s}\right)^2}} = \frac{1}{\sqrt{2(1 - \cos 2\pi \frac{f}{f_s})}} \quad (31.84)$$

and, evaluating the phase directly from the z-plane plot,

$$\angle H(f) = \underbrace{2\pi \frac{f}{f_s}}_{\text{From zero}} - \underbrace{\left(\pi \frac{f}{f_s} + \frac{\pi}{2}\right)}_{\text{From pole}} = 180 \frac{f}{f_s} - 90 \text{ (degrees) for } 0 < f < f_s \quad (31.85)$$

At DC the phase contribution from the zero is 0° , while the phase contribution from the pole, at a frequency just above DC, is 90° . The result is an overall phase response of -90° . At $f_s/4$ the phase contribution from the zero is 90° , while the phase contribution from the pole is 135° , resulting in an overall phase response of -45° . ■

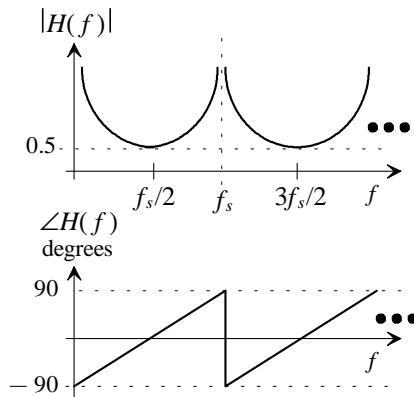
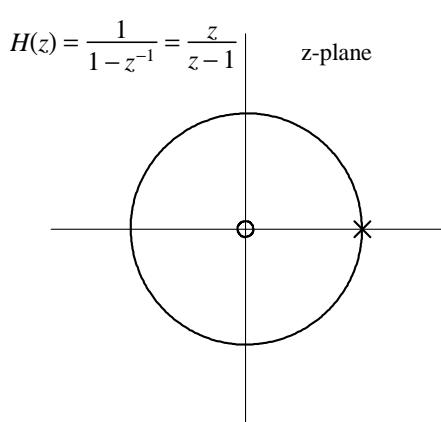


Figure 31.34 The z-plane representation along with magnitude and phase response for a digital integrator.

Consider the possible input to our digital integrator (Fig. 31.33) and the resulting output shown in Fig. 31.35a and 31.25b respectively. In this figure we are using +1 to indicate the peak positive input sinewave amplitude and -1 to indicate the peak negative input amplitude. The frequency of the sinewave is $f_s/2$ and so, according to our magnitude plot in Fig. 31.34, the gain is 0.5 (the peak-to-peak amplitude of the sinewave is reduced by one-half). Looking at Fig. 31.35, we should see that the initial state of the register (the latches) used in the integrator will, together with the input, determine the DC offset in the output waveform. For example, if the latches initially contained zero and the first sample was +1, as seen in Fig. 31.35a, then we would get the waveform shown in Fig. 31.35b. If, instead, the first sample were -1, then the entire waveform in Fig. 31.35b would shift downwards by +1. In any integrator, digital or analog, the "initial conditions" will affect the output waveform.

The next important factor we should notice in Fig. 31.35 is that we picked the peaks of the input sinusoid as our inputs to the digital integrator. Shifting our ADC output sampling points by $T_s/2$ results in a signal of all zeroes being applied to the digital integrator. The result is no change in the integrator's output. Shifting the sampling points by $T_s/3$ results in an integrator input of ± 0.5 with a corresponding integrator peak-to-peak output of ± 0.25 . In any case, at $f_s/2$, the output of the digital integrator is one-half the input signal's amplitude.

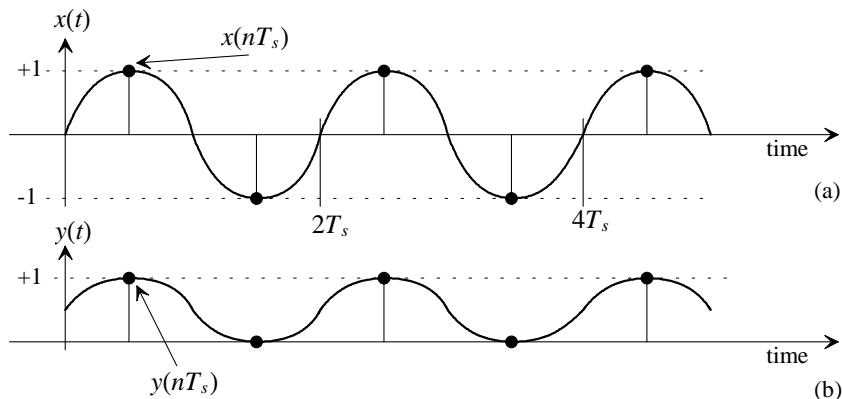


Figure 31.35 (a) Input and (b) output of the digital integrator of Fig. 31.33.

In the above discussion we used decimal numbers to represent the input and output signals of the integrator. In a practical implementation we use binary numbers. Let's use our ideal 8-bit data converters to illustrate the number system concerns. In these converters $V_{REF-} = 0$ and $V_{REF+} = 1.5$ V with $V_{LSB} = 5.859$ mV. A code of all zeroes corresponds to 0 V while a code of all ones corresponds to 1.494 V. The common mode voltage, V_{CM} , is $(V_{REF+} + V_{REF-})/2 = 0.75$ V. Again this number system is called offset binary. Figure 31.36 illustrates the representation of a full-scale sinusoid using the offset binary number format.

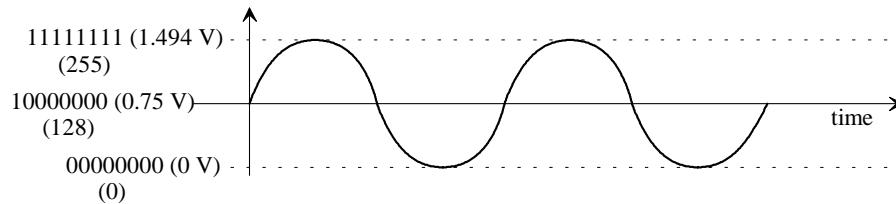


Figure 31.36 Representing a sinusoid in binary offset format.

We should compare the binary offset numbers of Fig. 31.36 to the decimal numbers of Fig. 31.35 and notice that if we apply the DC components of each signal to the integrator, we get totally different results. In Fig. 31.35 the DC component of the input has a decimal value of 0. Applying 0 to our integrator causes the output of the integrator to remain unchanged. In Fig. 31.36 the DC component of the input is the common mode voltage (halfway between the reference voltages) of 10000000. Applying this value to the integrator results in the integrator's output increasing until the output changes from all ones to all zeroes (the output rolls over). Clearly, the binary offset representation has some practical limitations when used in a digital integrator. To avoid these problems, the binary offset format is usually converted into the *two's complement* format prior to application to the digital integrator.

In two's complement the left-most bit is the sign bit. A zero represents a positive number (except for all zeroes, 00000000, or the common mode voltage) and a one represents a negative number (see Fig. 31.37). A binary offset number can be translated back and forth between a two's complement number by simply complementing the MSB of the code (running the MSB through an inverter). For this reason, and others that will be discussed later (ease of implementing subtraction and no overflow problems), two's complement is the preferred format for data words in digital filtering.

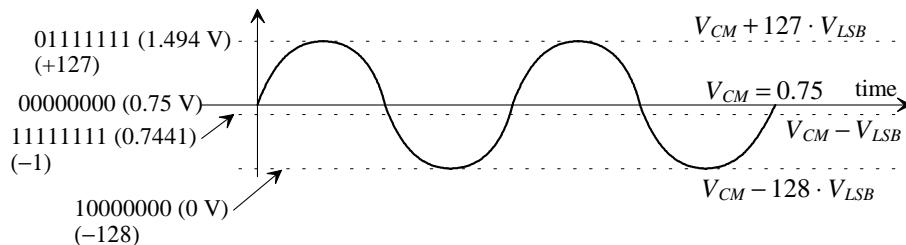


Figure 31.37 Representing a sinusoid in two's complement format.

31.2.2 Decimating Filters for ADCs

We saw, from Eq. (31.69), that when we employ averaging in a data converter we have to limit the input signal bandwidth, B , to the Nyquist frequency, f_n , divided by the number of points averaged, K . Knowing this, we may want to lower the rate at which these samples are coming out of the averaging filter to simplify the circuitry after averaging and to lower the power dissipation. Our new, effective sampling frequency is given by

$$f_{s,new} = 2B = \frac{f_s}{K} \quad (31.86)$$

This reduction in the effective sampling frequency is termed *decimation* and is illustrated with the block diagram shown in Fig. 31.38. The term decimation (or decimate) can be confusing since, among other things, the dictionary definition is, "to select by lot and kill one in every ten." The origin of the word comes from a method of punishing military troops by selecting one in every ten for execution. Our much more kind-hearted definition will mean that we are passing the input word through a lowpass digital filter and then down-sampling the result. This procedure is effectively passing the digital data through an antialiasing filter and then resampling the result at a lower rate.

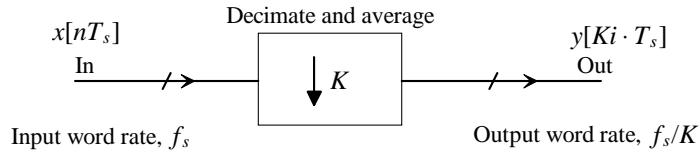


Figure 31.38 Block diagram of a decimation filter.

The Accumulate and Dump

In the simplest form we can write the input and output of the decimation filter, in the time domain, as

$$y[Ki \cdot T_s] = \sum_{n=K(i-1)}^{Ki-1} \frac{x[n \cdot T_s]}{K} \quad (31.87)$$

which is nothing more than averaging K input samples. Before going any further, we should make sure we understand this equation. Our decimation filter will take K input samples, add them together, and then divide the result by K to obtain the average of the input. If $K = 16$ and $i = 1$, then samples $x[0]$ through $x[15]$ are summed and divided by 16.

As we saw in Ex. 31.17, the actual division by K is dependent on the increase in the number of bits in the output word. For example, if our input word is 8-bits and $K = 16$ then the output word, before dividing by K , is 12-bits (adding sixteen, 8-bit words, results in a 12-bit word). If the ultimate increase in resolution is 2-bits, then the final output word size is 10-bits and we throw the lower 2-bits away. This could effectively mean that instead of dividing by 16, we divide by 4.

We can rewrite Eq. (31.87) in the z-domain (so that we have the z-domain representation for the decimation filter) as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{K} \sum_{n=0}^{K-1} z^{-n} = \frac{1}{K} (1 + z^{-1} + z^{-2} + \dots + z^{1-K}) \quad (31.88)$$

or

$$H(z) = \frac{1}{K} \cdot \frac{1 - z^{-1}}{1 - z^{-1}} \cdot (1 + z^{-1} + z^{-2} + \dots + z^{1-K}) \quad (31.89)$$

or finally, the z-domain transfer function for the decimator (averager), is

$$H(z) = \frac{1}{K} \frac{1 - z^{-K}}{1 - z^{-1}} \quad (31.90)$$

If $K = 2$, Eq. (31.90) becomes

$$H(z) = \frac{1}{2} \frac{(z+1)(z-1)}{z(z-1)} = \frac{1}{2}(1 + z^{-1}) \quad (31.91)$$

noting that we have already discussed this case, Eq. (31.59), earlier. Note also that the division by K may be ignored in this case since, as discussed earlier, the word size increases by one bit when adding the two words. (However, our realized increase in resolution is only 0.5 bits meaning the SNR increases by 3 dB.)

One circuit used to implement Eq. (31.90) is shown in Fig. 31.39 and is called an *accumulate-and-dump* circuit. To understand the operation of this circuit let's assume the bottom set of latches are reset. The sampling clock is used to clock this set of latches K times until the sum of K inputs is accumulated. At this time the accumulated sum is dumped into the output latches. Also, at this time, the bottom set of latches is reset to zero to start the accumulation process for the next set of K input samples. Note the clock rate on the input of the circuit is f_s and the clock rate coming out of the circuit is f_s/K .

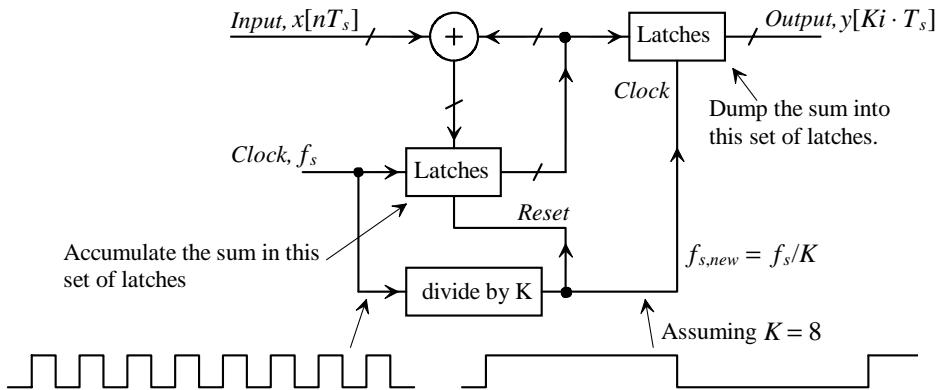


Figure 31.39 An accumulate-and-dump circuit used for decimation and averaging.

We can determine the frequency response of the accumulate-and-dump circuit by setting $z = e^{j2\pi f/f_s}$ in Eq. (31.90) or

$$H(f) = \frac{1}{K} \cdot \frac{1 - e^{-jK2\pi f/f_s}}{1 - e^{-j2\pi f/f_s}} = \frac{1}{K} \cdot \frac{1 - e^{-j2\pi f_{fs,new}/f}}{1 - e^{-j2\pi f/f_s}} \quad (31.92)$$

or, knowing $|1 - e^{-jx}| = |(1 - \cos x) + j \sin x| = \sqrt{(1 - \cos x)^2 + (\sin x)^2} = \sqrt{2(1 - \cos x)}$,

$$|H(f)| = \frac{1}{K} \cdot \frac{\sqrt{2(1 - \cos K2\pi f/f_s)}}{\sqrt{2(1 - \cos 2\pi f/f_s)}} = \frac{1}{K} \cdot \frac{\sin(K\pi f/f_s)}{\sin(\pi f/f_s)} = \frac{\text{sinc}(K\pi f/f_s)}{\text{sinc}(\pi f/f_s)} \quad (31.93)$$

We can sketch the frequency response of the accumulate-and-dump averager/decimator for different values of K as seen in Fig. 31.40. Note that the accumulate-and-dump circuit averages K samples while also reducing the output word rate to f_s/K (decimation). For

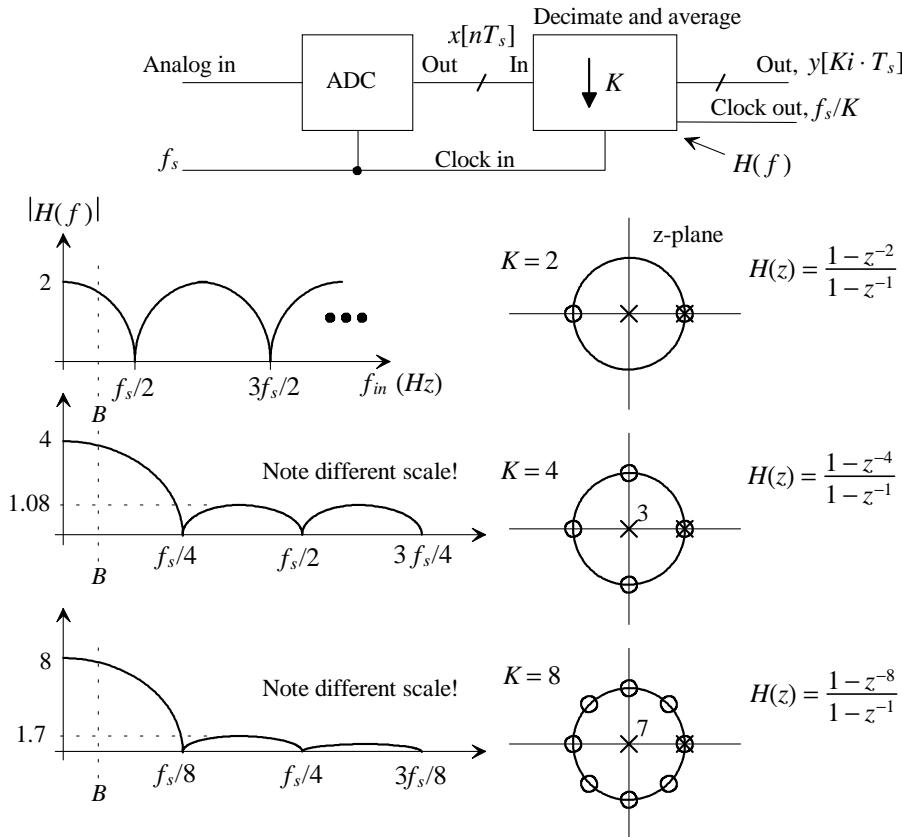


Figure 31.40 Frequency response of the accumulate-and-dump for various values of K .

obvious reasons, Eq. (31.93), the frequency response of the accumulate-and-dump circuit, or an averaging filter with the z-domain response given by Eq. (31.90), is sometimes called a *sinc filter*.

Example 31.21

Determine the pole and zero locations (verify the z-plane plot in Fig. 31.40) for an averaging filter that averages eight samples.

We can write the z-domain representation of the averager using Eq. (31.90) without the scaling factor K as

$$H(z) = \frac{1-z^{-8}}{1-z^{-1}} = \frac{z^8-1}{z^7(z-1)}$$

or

$$H(z) = \frac{(z-1)(z+1)(z-j)(z+j)\left(z+\frac{1}{2}-j\cdot\frac{1}{2}\right)\left(z+\frac{1}{2}+j\cdot\frac{1}{2}\right)\left(z-\frac{1}{2}-j\cdot\frac{1}{2}\right)\left(z-\frac{1}{2}+j\cdot\frac{1}{2}\right)}{z^7(z-1)}$$

■

The general shape of the frequency response of an averaging filter is shown in Fig. 31.41. It's desirable to determine the amount of attenuation provided by this filter by specifying the ratio of the peak value of the main lobe to the peak value of the first sidelobe. If this ratio is large enough, we can limit our concerns to the filter response below f_s/K . Also note that the "gain" of the filter K is somewhat irrelevant in a frequency response discussion because increasing K simply means our digital word size is increasing. The output of the filter is scaled (divided), as discussed earlier, by some number less than K to get the final increase in resolution (the final number of bits increase over the input word size).

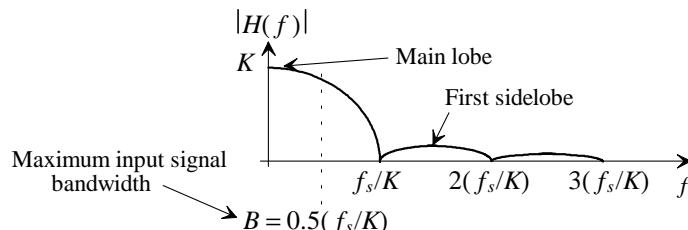


Figure 31.41 General frequency response of an averaging filter, Eq. (31.90).

The attenuation can be determined using Eq. (31.93) evaluated at $1.5(f_s/K)$ as

$$\left| \frac{\text{Main lobe}}{\text{First sidelobe}} \right| = K \cdot \sin\left(\frac{1.5\pi}{K}\right) \text{ for } K \geq 3 \quad (31.94)$$

This equation is plotted in Fig. 31.42 against averaging factor K . Note how the maximum amount of attenuation, as the number of averages increases, approaches 13.5 dB. This is a *significant limitation* and results in the need to cascade averaging filter stages to attain a large amount of attenuation at frequencies above f_s/K (more on this topic in a moment).

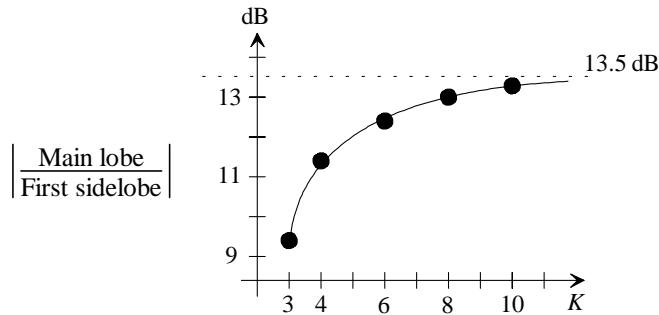


Figure 31.42 Averaging filter attenuation versus number of points averaged K.

It's also of interest to determine how much droop the filter will introduce into the signal frequencies of interest. Figure 31.43 shows the droop (attenuation) at the maximum input bandwidth, B . We can calculate the amount of droop, again using Eq. (31.93) when $f = f_s/(2K) = B$, as

$$\text{Droop} = \frac{1}{K \cdot \sin\left(\frac{\pi}{2K}\right)} \quad (31.95)$$

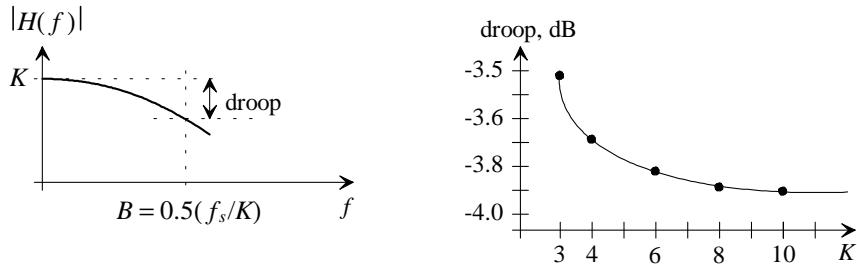


Figure 31.43 Droop at edge of signal bandwidth when using an averaging filter.

It should be obvious that if we desire large amounts of attenuation through our digital averaging (sinc) filter, we have severe limitations (only 13.5 dB attenuation maximum) when using the basic accumulate-and-dump circuit of Fig. 31.39. To increase the attenuation we might try to cascade accumulate-and-dump circuits as seen in Fig. 31.44. This cascade has several practical problems. While the attenuation can be increased to $L \cdot 13.5$ dB, where L is the number of stages, the final output sampling frequency drops to f_s/K^L and the maximum input frequency, B , drops to $0.5f_s/K^L$. To illustrate the limitations imposed by a cascade of accumulate-and-dumps, let's assume we need 60 dB of attenuation through the filter and our sampling frequency, $f_s = 100$ MHz. If $K = 8$, then we need to cascade five accumulate-and-dump stages. The sampling frequency coming out of the final stage is 3 kHz! The droop at 1.5 kHz (B) remains 3.9 dB. Clearly, cascading accumulate-and-dump circuits is not practical for most situations.

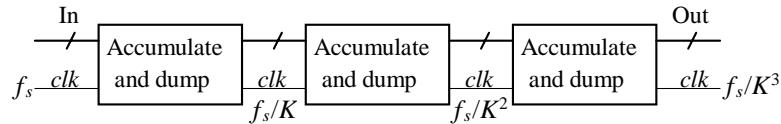


Figure 31.44 Cascading accumulate-and-dump circuits to increase filter attenuation.

Averaging without Decimation

The accumulate-and-dump performed averaging and decimation in one stage. In other words, for example with $K = 4$, it summed four input samples, as shown in the sequence below, and passed the result to the output:

$$\overbrace{x(1) + x(2) + x(3) + x(4)}^{\text{First output of the accumulate and dump}} + \overbrace{x(5) + x(6) + x(7) + x(8)}^{\text{Second output}} + x(9) + \dots \quad (31.96)$$

We can get a much more efficient and practical filter if we average the input samples without decimation (sample frequency reduction). The final output clocking frequency (after decimation) can be set to $2B$ (Eq. [31.69]) and can occur in a later stage in the filter's construction. The reduction in sampling frequency reduces power and circuit complexity (for example, serial multipliers can be used in a digital filter).

Consider the running sum shown below

$$\begin{aligned} &\overbrace{x(1) + x(2) + x(3) + x(4)}^{\text{First output of averager}} + x(5) + x(6) + x(7) + x(8) + x(9) + \dots \\ &x(1) + \overbrace{x(2) + x(3) + x(4) + x(5)}^{\text{Second output of averager}} + x(6) + x(7) + x(8) + x(9) + \dots \\ &x(1) + x(2) + \overbrace{x(3) + x(4) + x(5) + x(6)}^{\text{Third output of averager}} + x(7) + x(8) + x(9) + \dots \\ &x(1) + x(2) + x(3) + \overbrace{x(4) + x(5) + x(6) + x(7)}^{\text{Fourth output of averager}} + x(8) + x(9) + \dots \end{aligned} \quad (31.97)$$

It should be obvious that the outputs of the averager occur at the same rate as the averaging filter's input (no change in the sampling frequency). The z-domain representation of the averager is the same as the accumulate-and-dump's transfer function

$$y(nT_s) = \frac{x[nT_s] + x[(n-1)T_s] + x[(n-2)T_s] + \dots}{K} \rightarrow Y(z) = \frac{X(z)(1+z^{-1}+z^{-2}+\dots+z^{1-K})}{K} \quad (31.98)$$

or, reviewing Eqs. (31.88) and (31.89), results once again in

$$H(z) = \frac{1}{K} \frac{1-z^{-K}}{1-z^{-1}} \quad (31.99)$$

where the division by the number of points averaged, K (K is four in Eq. [31.97]) is performed by simply adjusting the final word size to the desired length (as discussed

earlier and in Ex. 31.17). The transfer function of a cascade of L of these averaging filters can be written as

$$H(z) = \left[\frac{1}{K} \frac{1-z^{-K}}{1-z^{-1}} \right]^L \quad (31.100)$$

or

$$|H(f)| = \left[\frac{\text{sinc } K\pi \frac{f}{f_s}}{\text{sinc } \pi \frac{f}{f_s}} \right]^L \quad (31.101)$$

Before we discuss the implementation of the averaging filter (a.k.a. sinc filter), let's borrow the results from Figs. 31.42 and 31.43 and notice the attenuation for a cascade of L averaging filters is

$$\left| \frac{\text{Main lobe}}{\text{First sidelobe}} \right| = \left| K \cdot \sin\left(\frac{1.5\pi}{K}\right) \right|^L \approx L \cdot 13 \text{ dB for } K \geq 8 \quad (31.102)$$

while the droop, at the maximum input frequency, B , is

$$\text{Droop} = \left[K \cdot \sin\left(\frac{\pi}{2K}\right) \right]^L \approx L \cdot (-3.9) \text{ dB for } K \geq 8 \quad (31.103)$$

Also note that Eq. (31.69) is still valid, which means that we don't have a significant restriction on the maximum allowable input frequency.

To compare the cascade of averaging filters to the limitations imposed by a cascade of accumulate-and-dumps as discussed earlier, let's once again assume we need 60 dB of attenuation through the averaging filter and our sampling frequency is $f_s = 100$ MHz. If $K = 8$, then we need to cascade five averaging filter stages as seen in Fig. 31.45. The clock frequency coming out of the final stage is 100 MHz and needs to be reduced to 12.5 MHz in the last stage by simply dividing the clock down before clocking a final set of latches or by using an accumulate-and-dump for the final stage. The 12.5 MHz output rate and 6.25 MHz input frequency bandwidth, B , should be compared to the 3 kHz output clock frequency and 1.5 kHz input frequency calculated earlier for the cascade of accumulate-and-dumps. The droop B remains 19.5 dB and can be a serious concern in many situations. Obviously, limiting the input bandwidth further reduces the droop at B . Figure 31.46 shows the frequency response of a cascade of averaging filters (the frequency response is given by Eq. [31.101]).

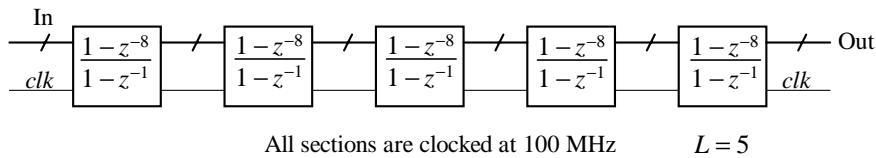


Figure 31.45 Cascading averaging circuits to increase filter attenuation.

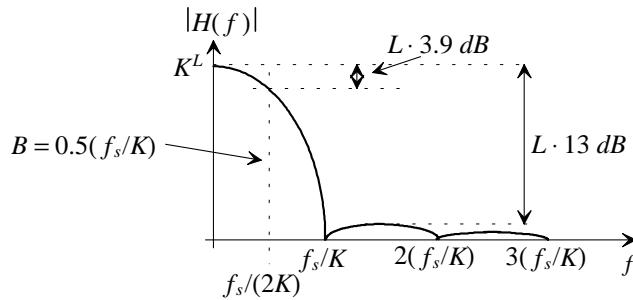


Figure 31.46 General frequency response of an averaging filter, Eq. (31.101).

Relaxed Requirements Placed on the Antialiasing Filter Revisited

Consider the input (to the ADC) spectrum shown in Fig. 31.47a resulting from passing an input signal through an anti-aliasing filter (AAF). The AAF has to remove all spectral content above $f_s - f_s/K$, as we'll show in a moment, to avoid aliasing when using a digital averaging filter (with spectral response shown in Fig. 31.46). On the output of the ADC, after sampling and aliasing, the characteristics of the signal are shown in Figs. 31.47b and 31.47c. In Fig. 31.47b we show the individual reproductions of the original base spectrum. In part (c) we combine the spectrums to show the continuous ADC output spectrum and include a sketch of the digital filter frequency response. From Fig. 31.47c we should see

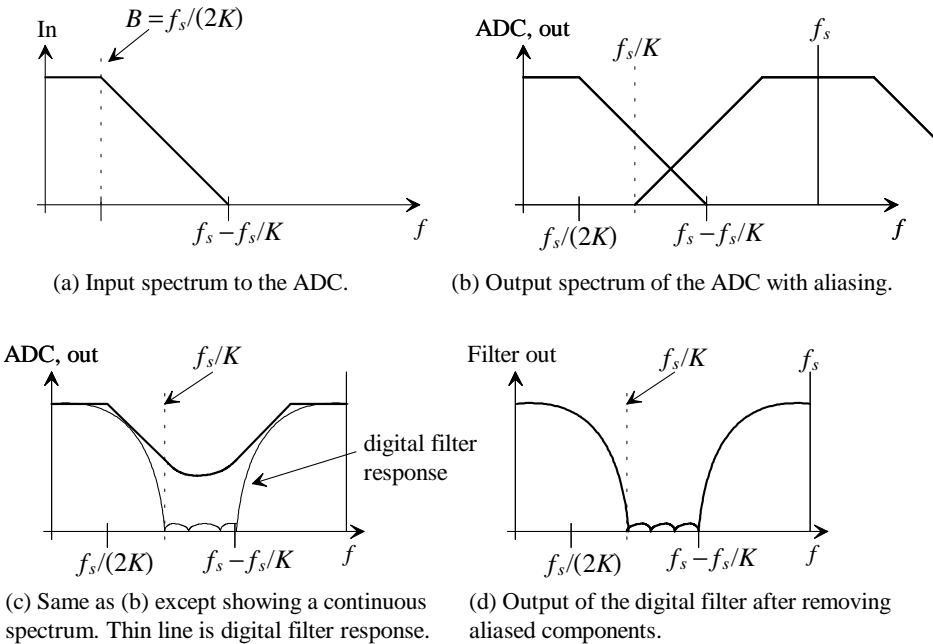


Figure 31.47 How a digital (averaging) filter helps remove aliased spectral components.

why the AAF has to limit the ADC input spectral content to $f - f_s/K$. At f_s/K the output of the digital filter goes to zero. This assumes the first sidelobe in the digital filter is sufficiently small so that the amount of aliasing past f_s/K is negligible. In many systems that employ a digital signal processor, an additional abrupt-cutoff lowpass digital filter is used after the averaging and decimation process to provide additional alias signal removal.

Note, in Fig. 31.47d the output of the digital filter, as mentioned earlier, has a periodic frequency response (and so the spectrum out of the digital filter will still have aliased components but hopefully not in the base spectrum). As a further example of the periodic nature of all digital filters, consider the $K = 8$ averaging filter frequency response shown in Fig. 31.48 (see Fig. 31.40 for a comparison). Note that the number of points in the frequency response between DC and f_s that go to zero (the number of zeroes in the transfer function) is seven (since the zero at DC is canceled by the pole at DC [$z = 1$]).

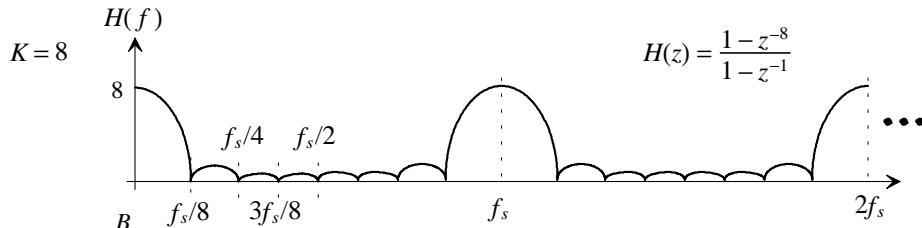


Figure 31.48 The periodic nature of a $K = 8$ averaging filter.

Implementing Averaging Filters

If we are not careful, the averaging filter can take up a large amount of chip area. In this section we discuss how to implement an averaging filter with a response given by Eqs. (31.100) and (31.101). To begin, let's break the averaging filter up into two parts, without including the scaling factor K ,

$$H(z) = \left[\frac{1 - z^{-K}}{1 - z^{-1}} \right]^L = \overbrace{(1 - z^{-K})^L}^{\text{L differentiators}} \cdot \overbrace{\left(\frac{1}{1 - z^{-1}} \right)^L}^{\text{L integrators}} \quad (31.104)$$

We have already looked at the transfer function of an integrator in Ex. 31.20. An alternate digital integrator design is shown in Fig. 31.49. The only difference between the circuit of Fig. 31.49 and Fig. 31.33 is the change in the phase response of the circuit.

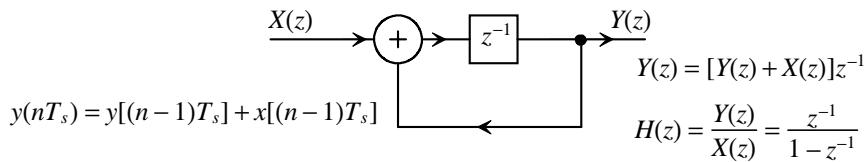
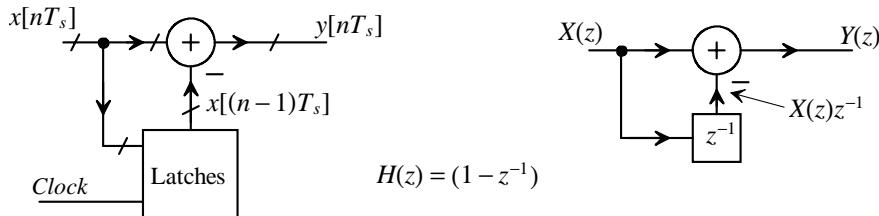


Figure 31.49 Alternate digital integrator.

**Figure 31.50** Block diagram of a digital differentiator.

The z-domain representation and schematic of a differentiator are shown in Fig. 31.50. Note the use of subtraction in the differentiator. The time domain description of a digital differentiator is given by

$$y[nT_s] = x[nT_s] - x[(n-1)T_s] \quad (31.105)$$

and the z-domain transfer function is given by

$$H(z) = \frac{Y(z)}{X(z)} = (1 - z^{-1}) \quad (31.106)$$

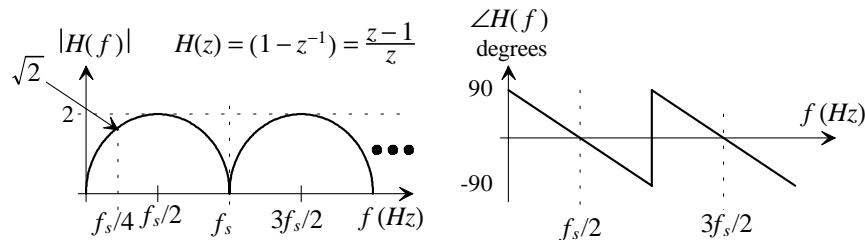
The frequency response of the differentiator is (see Fig. 31.51)

$$|H(f)| = \sqrt{2 \left(1 - \cos 2\pi \frac{f}{f_s} \right)} \quad (31.107)$$

and the phase response is

$$\angle H(f) = \frac{\pi}{2} - \pi \frac{f}{f_s} \text{ for } 0 < f < f_s \quad (31.108)$$

Again, like all of the digital filters we have discussed in this chapter, the phase response is linear (constant delay, meaning no phase distortion). Also note, assuming the two's complement number representation is used for the input/output (I/O) in the differentiator, that subtraction of two numbers, $A - B$, can be accomplished by using an adder with the B input complemented (run through inverters) prior to application to the adder and the adder carry-in tied to a logic one.

**Figure 31.51** Magnitude and phase response of digital differentiator.

Next consider the digital comb filter (or differentiator over a range of frequencies) circuit shown in Fig. 31.52. We should recognize this circuit's transfer function from Eq. (31.104) as

$$H(z) = 1 - z^{-K} = \frac{z^K - 1}{z^k}$$

with a magnitude response given by

$$|H(f)| = \sqrt{2\left(1 - \cos 2\pi K \frac{f}{f_s}\right)}$$

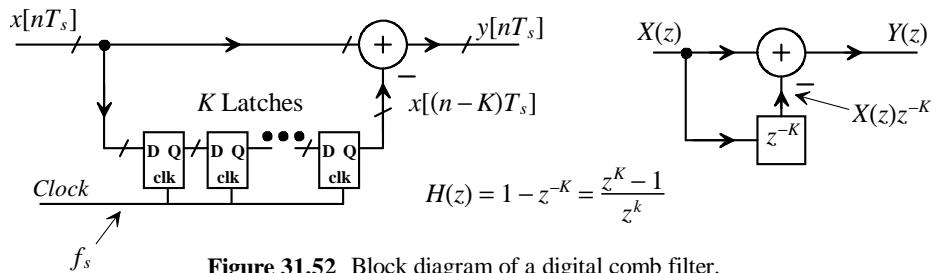


Figure 31.52 Block diagram of a digital comb filter.

Figure 31.53 shows the z-plane and frequency responses for comb filters with various values of comb filter delays K . Before we proceed with the implementation of the averaging filter defined by Eq. (31.104), let's discuss, intuitively, how we take the basic digital comb filter and make a lowpass, averaging filter. Remember that we evaluate $1 - z^{-K}$ around the unit circle, in the z-plane, to determine $H(f)$. If we look at Fig. 31.40, we see that the only difference between a comb filter and a lowpass averaging filter is the fact that we have added a pole to the transfer function at DC (i.e. $z = 1$) to cancel the zero at DC. This is important as we will be able to make highpass and bandpass averaging circuits by taking a comb filter and canceling the zeroes placed at other points on the unit circle. We'll discuss this in more detail in the next two sections. Note that by using Eqs. (31.79) and (31.80), we should be able to see why canceling a zero with a pole at DC results in a lowpass filter.

The comb filter of Fig. 31.52, or the differentiator of Fig. 31.50, is an example of a *finite impulse response* (FIR) digital filter. Applying a unit amplitude impulse to the input of the comb filter, and zeroes at all other times, causes the output of the comb filter to go to a one at the moment the impulse is applied and KT_s seconds later, and a zero at other times. In other words, the output response of the filter has a finite duration.

The integrator (sometimes called an accumulator) shown in Fig. 31.34 is an example of an *infinite impulse response* (IIR) digital filter. Applying a unit amplitude impulse to the input of the digital integrator, with zeroes at the remaining times, causes the output of the integrator to increase to one and remain at one indefinitely. In other words, the output response of the integrator is of infinite duration.

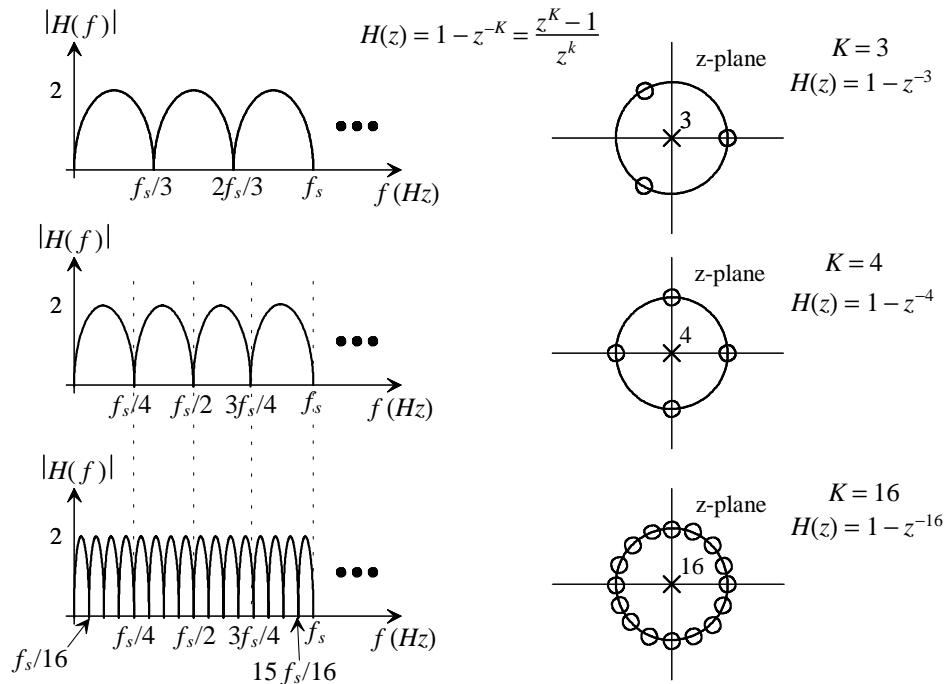


Figure 31.53 Frequency response and z-plane plots for various values of K in a comb filter.

Equation (31.104) can be implemented by cascading L integrators and L comb (differentiators) filters as shown in Fig. 31.54. As we discussed earlier, the integrators have "infinite" gain at DC, which can result in register overflow. However, if we use two's complement number representation, we'll see that we get the correct answer out of the filter as long as the word length used to implement the filter is long enough (using two's complement will avoid overflow problems).

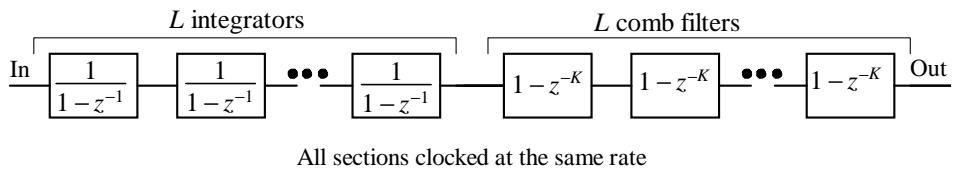


Figure 31.54 Implementation of an averaging filter using integrators and comb filters.

Example 31.22

Consider an averaging filter (Eq. 31.104) using $L = 3$ and $K = 8$ with an input word length of 8 bits. Determine the final number of bits coming out of the filter and show that a constant input of 01110000 (+112 two's complement) results in the correct output code. Also, discuss overflow concerns.

From Eq. (31.53) we can calculate an increase in resolution of 1.5 bits per filter stage so the final output word size should be 13 bits (the original 8-bits input plus an additional 4.5 bits from averaging $8^3 [= K^L]$ samples).

The "gain" of the filter at DC is, from Fig. 31.46, $K^L = 512$. This means that our 8-bit input of 112 (0111 0000) will be multiplied by 512 and result in an output, prior to scaling, of 57344, or a binary code of 0 1110 0000 0000 0000 (17 bits in the general case). From Eq. (31.100) we would then divide this code by 512 (drop the lower nine bits.) However, this would result in an output word size equal to the input word size (both 8-bits and no increase in resolution). So, for the general input signal that is time-varying and to get the final 13 bits, we divide the 17-bit filter output by 16 (drop the lower four bits so our final output is 13 bits or $0\ 1110\ 0000\ 0000 [3,584 = 32 \cdot 112 \text{ since 13-bits}]$). A block diagram of the filter implementation is shown in Fig. 31.55. A MUX is needed in between each integrator stage to adjust the two's complement word size up by $\log_2 K$ bits.

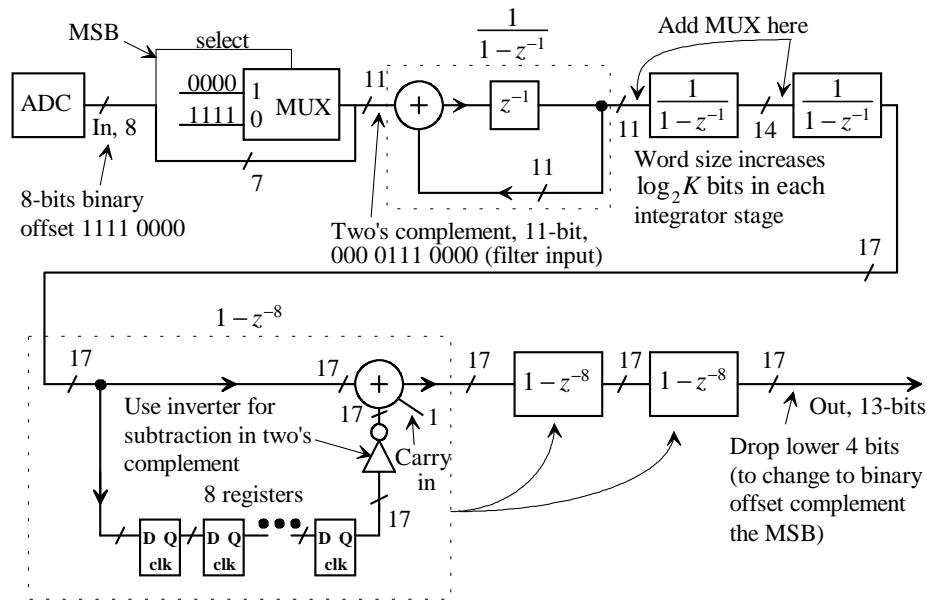


Figure 31.55 Block diagram of the filter discussed in Ex. 31.22.

Let's discuss overflow concerns. To keep the discussion simple, let's just consider a single integrator and comb filter stage, that is, $L = 1$. A constant two's complement, 8-bit, input of 0111 0000 (+112) into the integrator will result in an output, assuming we start with all zeroes, of

Output (Sum0)	000 0000 0000 0 (or V_{CM} [two's complement])
Input1	000 0111 0000 112
Sum1	000 0111 0000 112

<i>Input2</i>	000 0111 0000 112
Sum2	000 1110 0000 224
<i>or since the input is a constant</i>	000 0111 0000 (112)
Sum3	001 0101 0000 336
Sum4	001 1100 0000 448
Sum5	010 0011 0000 560
Sum6	010 1010 0000 672
Sum7	011 0001 0000 784
Sum8	011 1000 0000 896
Sum9	011 1111 0000 1008
Sum10	100 0110 0000 1120
Sum11	100 1101 0000 1232
Sum12	101 0100 0000 1344
Sum13	101 1011 0000 1456
Sum14	110 0010 0000 1568
Sum15	110 1001 0000 1680
Sum16	111 0000 0000 1792
Sum17	111 0111 0000 1904
Sum18	111 1110 0000 2016
Sum19	000 0101 0000 80
Sum20	000 1010 0000 192
	Overflow!

These sums are applied to our comb filter. Since $K = 8$, we won't have a meaningful comb filter output until our ninth integrator output. At this time the output of the comb filter will be the difference between Sum1 and Sum9, that is, $\text{Sum9} - \text{Sum1}$. The ninth integrator output is 1008 while the first is 112. The difference being 896 (011 1000 0000 or $8 \cdot 112$ with 11 bits). In fact, we can take any difference between sums spaced eight clock cycles apart, even after overflow, and get this result (looking only at the lower 11 bits.) For example, $\text{Sum18} - \text{Sum10}$ is

$$\begin{array}{rcl} \text{Sum18} & & 111 1110 0000 (2016) \\ \text{minus } \text{Sum10} & & 100 0110 0000 (1120) \\ & & (896) \end{array}$$

In two's complement, for subtraction, we complement, and add one (set the adder carry-in bit high) to the number we are subtracting, which gives, for this example,

$$\begin{array}{rcl} \text{Sum18} & & 111 1110 0000 \\ \text{Sum10(comp)} & & 011 1001 1111 \\ \text{plus} & & 1 \text{ (adder carry)} \\ & & 011 1000 0000 (896) \end{array}$$

noting that we threw out the 12th bit in the sum. As a final example,

$$\begin{array}{rcl} \text{Sum19 } 000 0101 0000 (80) & \text{or} & \text{Sum19 } 000 0101 0000 \\ \text{minus } \text{Sum11 } 100 1101 0000 (1232) & \text{or} & \text{Sum11(comp) } 011 0010 1111 \\ \text{difference } (-1152 \text{ or } 896) & & \text{plus } 1 \\ & & 011 1000 0000 (896) \end{array}$$

While this last discussion focused on $L = 1$, we could use any number of stages as long as the register size in our integrators can accommodate a binary number of at least $K \cdot 2^N$, where N is the number of bits in the input word. ■

We might notice from this example that the amount of hardware needed to implement the averaging filter is significant. The main contributors to the final filter layout size are the registers used in the comb filters (a total of twenty-four 17-bit registers are used). It turns out, as discussed earlier, that we can use the reduction in clock frequency (decimation) to reduce the number of registers used in the comb filter, Fig. 31.56. By dividing the clock frequency down by K , we can reduce the number of registers used in each comb filter to one. In either Fig. 31.54 or Fig. 31.56 the delay used in the comb filter is KT_s . Figure 31.56 is the preferable way to implement decimation/averaging filters (however, see aliasing description below for practical implementation concerns).

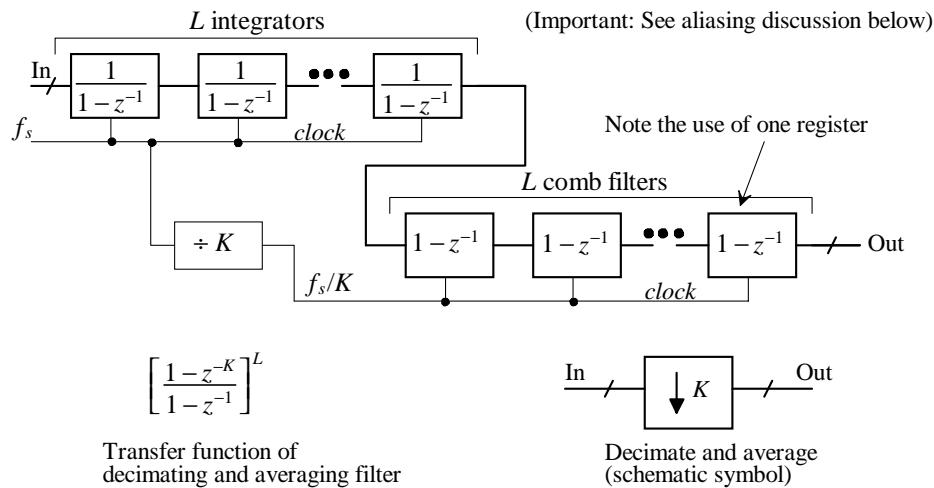
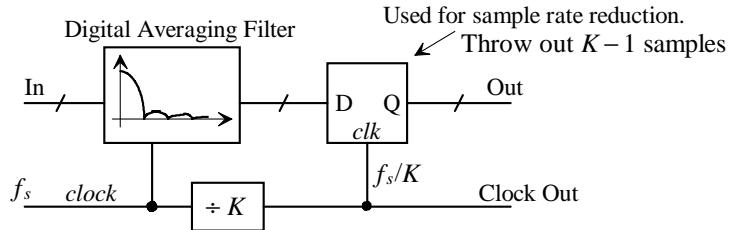


Figure 31.56 Using a reduction in clock frequency to lower complexity in averaging filters.

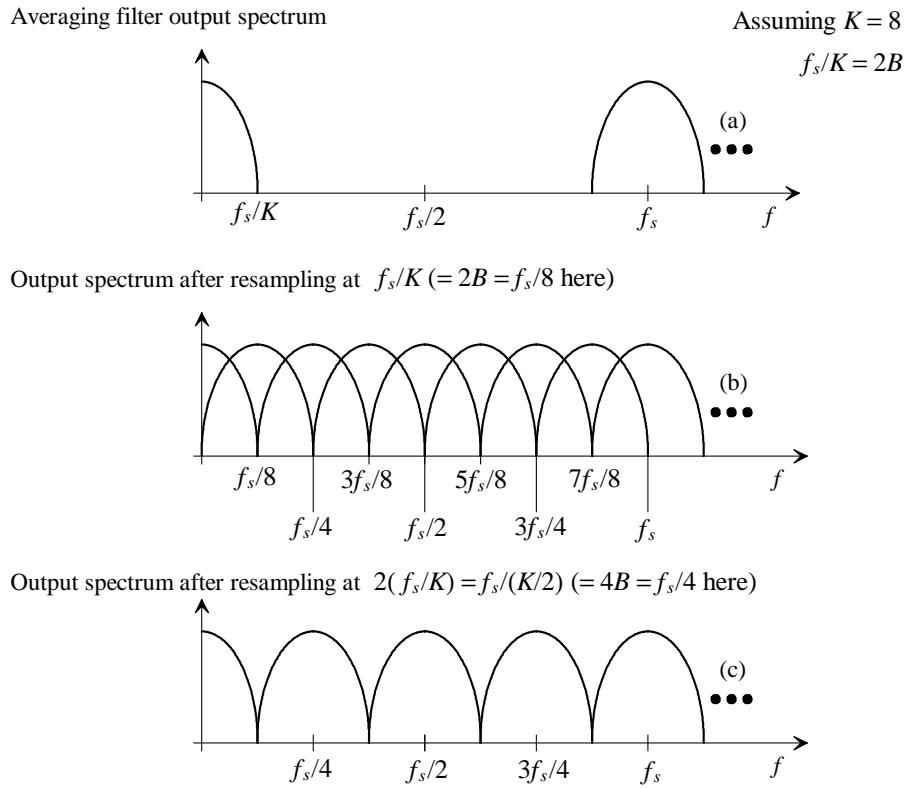
Aliasing Concerns When Using Decimation

While we are used to discussing aliasing concerns when sampling an analog waveform, we can also discuss aliasing when (re-) sampling a digital waveform. Reducing the sampling or clocking frequency (decimation) can be thought of as resampling a digital waveform at a lower rate, Fig. 31.57. In Fig. 31.57 the averaging filter can be thought of as the antialiasing filter with nonzero spectral content up to f_s/K (assuming the side lobes are sufficiently below the main lobe), while the set of latches on the output can be thought of as the sampler.

Figure 31.58a shows the output spectrum of the averaging filter (see, also, Fig. 31.47), assuming the side lobes are sufficiently small so that they are not a concern. If we resample this signal at f_s/K , we get the spectrum shown in Fig. 31.58b. The resulting spectrum shows a *problem* (aliasing in the base spectrum) that would be encountered using the filter of Fig. 31.56 unless our input signal bandwidth, B , is limited, using an analog antialiasing filter or a separate digital filter, to $f_s/2K$. As discussed earlier (Fig.

**Figure 31.57** Resampling a digital waveform.

31.47), we can use the averaging filter for additional aliased signal removal. While we may only be interested in signal content up to B (Eq. [31.69]), we can still have unwanted signal content between B and f_s/K that will alias into the base spectrum. It is desirable to eliminate this problem altogether. We can do this by resampling at $2(f_s/K)$. The averaging/decimation filter shown in Fig. 31.56 is changed so that the divider divides by

**Figure 31.58** Showing signal spectrum (a) prior to decimation and (b) and (c) after.

$K/2$. By adding a register to the comb filter the comb filters are $1 - z^{-2}$. The resulting spectrum is shown in Fig. 31.58c. Note that aliasing is a very *important* concern when designing the averaging filter.

If the first sidelobe amplitude isn't sufficiently small, as was assumed in Fig. 31.58, a larger resampling frequency or decimation frequency can be used to minimize aliasing. A common intermediate clocking frequency is $4 \cdot (f_s/K)$. A sample averaging filter output waveform is shown in Fig. 31.59a, where the sidelobe amplitude is no longer insignificant. Figure 31.59b shows the spectrum, assuming $K = 8$ and the resampling frequency is $f_s/2 (= 4 \cdot [f_s/8])$ (meaning the divider in Fig. 31.56 is $f_s/[K/4]$) and each comb filter stage uses four registers [$1 - z^{-4}$]). Note how the third side lobe is aliased into the base spectrum in this example, while the first sidelobe was aliased into the base spectrum in Fig. 31.58c (although it was not shown in the figure). Figure 31.60 shows that the ratio of the main lobe to the third sidelobe is approximately 20 dB (assuming $K \geq 8$). The possible large amount of baseband aliasing together with the droop at B may result in the desired *input bandwidth, B, being limited to frequencies below $f_s/(2K)$* with an external analog AAF or an additional digital filter (more on this in a moment). Finally, notice that at DC (or very low frequencies) in Figs. 31.58b, 31.58c, or 31.59b there is essentially no aliased signal. This is the result of the zeroes in the averaging filter transfer function, Eq. (31.99), falling at multiples of the decimation frequency.

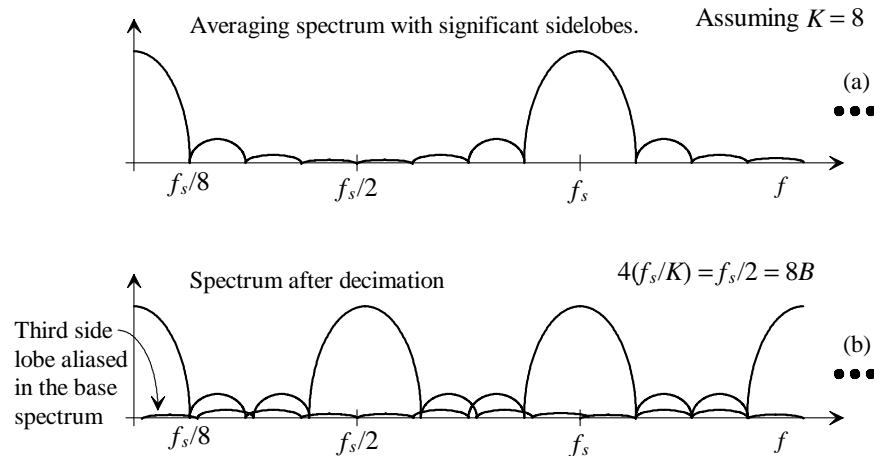


Figure 31.59 Showing signal spectrum with significant side lobes (a) prior to decimation and (b) after decimation.

A Note Concerning Stability

Consider the weighted integrating filter block diagram shown in Fig. 31.61. The output of the circuit is fed back to the input after it is multiplied by a . Multiplication by a may be performed with a dedicated multiplier or it may simply be a shift operation (multiplying by

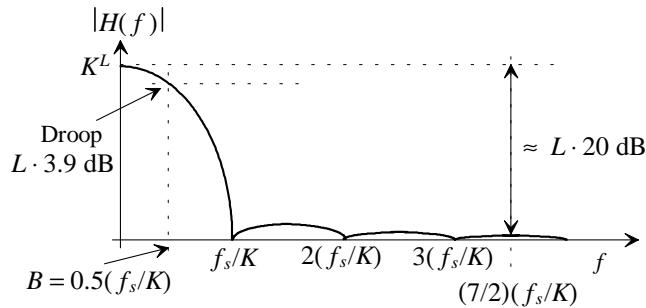


Figure 31.60 Decimating at four times the Nyquist rate. Showing (aliased) third sidelobe.

0.5 is simply a shift-right operation). The output of the circuit in the time domain may be written as

$$y[nT_s] = x[(n-1)T_s] + a \cdot y[(n-1)T_s] \quad (31.109)$$

or

$$y[nT_s] = x[(n-1)T_s] + a \cdot x[(n-2)T_s] + a^2 \cdot x[(n-3)T_s] + a^3 \cdot x[(n-4)T_s] + \dots, \quad (31.110)$$

which will obviously blow up if $a > 1$.

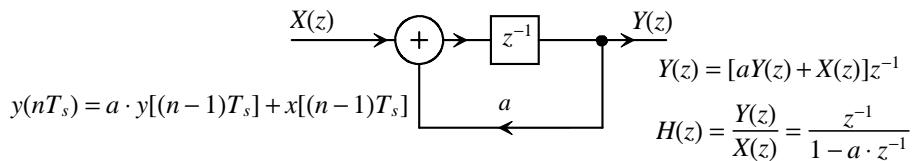


Figure 31.61 A weighted integrating filter.

The z-domain representation of Eq. (31.109) is

$$H(z) = \frac{1}{z-a} \quad (31.111)$$

Figure 31.62 shows the z-plane and magnitude plots specified by this equation. If $a > 1$ $H(z)$ becomes unstable, so *for a stable system* we must require our poles to reside within the unit circle. (There are no restrictions on the location of zeroes.) This sounds simple enough; however, notice that we have, in most of the previously discussed digital filters, placed poles right on the unit circle. If there is rounding in our digital numbers, we could be faced with an unstable digital filter. This would be a very common occurrence in a digital filter implemented using software, if care was not taken to avoid rounding errors. Since we use integer numbers in our hardware implementations, instability shouldn't be a problem unless we start to try to round numbers to decrease hardware complexity (performing divisions or multiplications) without being careful.

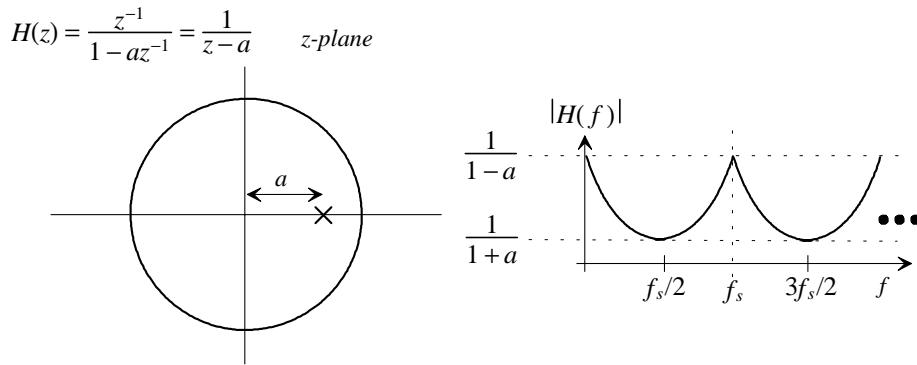


Figure 31.62 The *z*-plane representation and magnitude response for a weighted integrating filter.

Decimating Down to $2B$

In many situations (for example, we want to transmit the modulator output) it is desirable to reduce the clocking frequency down to twice the Nyquist frequency. Since the Nyquist frequency for our input bandwidth B is $f_s/(2K)$, our final output clocking frequency would be $f_s/K = 2B$. As we have just discussed this second stage decimation can result in significant aliasing. To eliminate this aliasing, a digital filter is often used to limit the bandwidth, after the first stage decimation, to $f_s/(2K)$ ($= B$). Before we proceed any further, let's summarize the discussion so far for the different situations.

1. The input to the ADC is bandlimited to B [$= f_s/(2K)$]. In this situation we can use the topology of Fig. 31.56 directly, decimating the sampling clock from f_s to f_s/K in one stage. The comb filters use one register. The smallest size averaging/decimation filter results in this situation and the output clock rate is $2B$.
2. The input, to the ADC is bandlimited to $2B$ ($= f_s/K$). In this situation the averaging filter provides some aliased signal removal, see Fig. 31.47. Assuming the aliased signal content in the first side lobe is insignificant, we can use the topology of Fig. 31.56 with a divider of $(K/2)$ and two registers in each comb filter; that is, each comb filter has a transfer function of $1 - z^{-2}$. The sampling clock gets reduced from f_s to $f_s/(K/2)$. The output clocking rate is now $4B$.
3. The input to the ADC is bandlimited to the Nyquist frequency $f_s/2$ (the general situation where the analog AAF has the most relaxed requirements). In this situation the averaging filter will again provide some aliased signal removal. We can use the topology of Fig. 31.56 with a divider of $(K/4)$ and four registers in each comb filter, that is, each comb filter has a transfer function of $1 - z^{-4}$. The sampling clock gets reduced from f_s to $f_s/(K/4)$ and the output clock rate is $8B$.

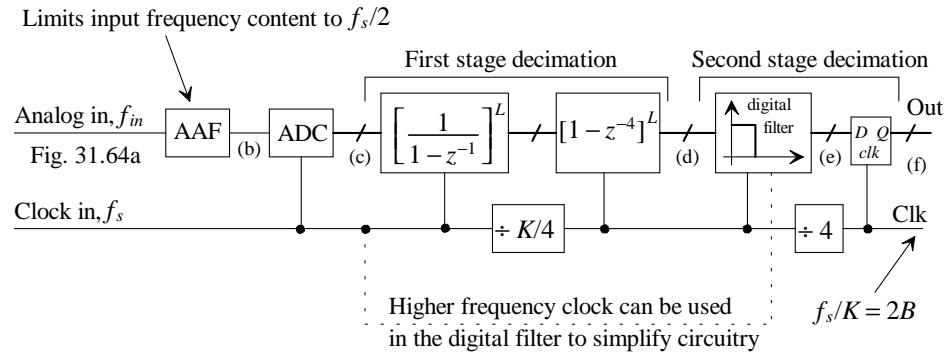


Figure 31.63 General averaging and decimation topology for an oversampled ADC.

4. Figure 31.63 shows the entire system for case 3 above, with the addition of a digital filter on the output of the first-stage decimation for reducing the sampling, or output clocking frequency, to $2B$. The digital filter used in the second decimation stage is generally a *half-band digital filter* (covered in most books on digital filtering). Half-band filters are used because of the simplicity of their implementation (half of the filter's coefficients are zero) and the fact that the filter's transition frequency is symmetric around its clocking frequency divided by four (which we can use, with a divider or using two, cascaded, half-band filters to set the filter's cutoff frequency to B when the clocking frequency is $4[f_s/K] = 8B$).

Figure 31.64 shows the spectrum of the signals in Fig. 31.63 for the general situation. Figure 31.64a is the AAF input, which we have drawn with an arbitrary shape. In 31.64b we see that the AAF limits the input signal spectrum to $f_s/2$ (and we should see, once again, the relaxed requirements placed on the AAF when using oversampling). Figure 31.64c shows the ADC's output spectrum resulting from sampling the input waveform (actually the output of the AAF). After first stage decimation, Fig. 31.64d, the signal is passed through a sinc (averaging) filter and then resampled at $4(f_s/K)$ ($=f_s/2$ when $K = 8$). Figure 31.64e shows the output of the half-band filter prior to down-sampling (second stage decimation). The figure assumes the half-band filter is clocked with an effective clock frequency of $4B$ (the actual clock frequency is $8B$ as discussed above) or $4[f_s/(K/4)]$. So the filter's cutoff frequency is B . The half-band filter's implementation may also use the high frequency clock signal to simplify the filter's implementation. Finally, Fig. 31.64f shows the spectrum resulting after final decimation. The clock frequency out of the final stage is $2B$, while the desired signal bandwidth is B .

It's important to remember that unless there is some reason to lower the clock frequency (for example, we want to store the ADC/averaging filter's digital output in memory), we can avoid the aliasing problems associated with decimation (see Fig. 31.58) and the added complexity. Also note that the desired spectrum of Fig. 31.64e must ultimately be reconstructed using a DAC and a reconstruction filter (RCF). Because of the unwanted spectral content, directly adjacent to the desired content, the reconstruction becomes more challenging when decimating down to $2B$.

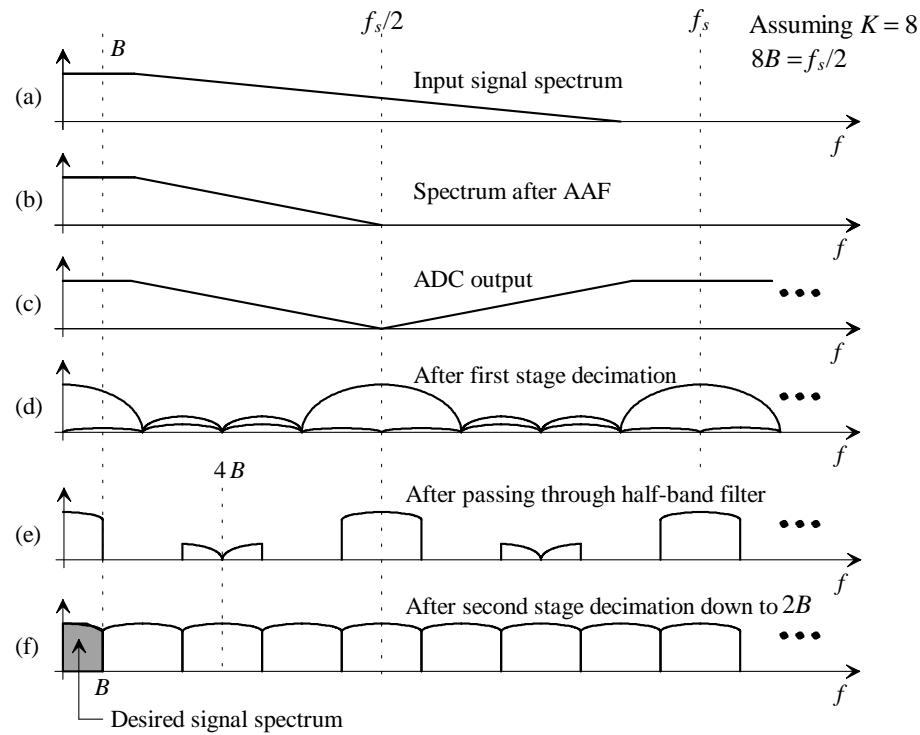


Figure 31.64 Spectrums of the resulting signals for the decimation scheme shown in Fig. 31.63.

31.2.3 Interpolating Filters for DACs

In the last section we discussed how we were able to average the outputs of a lower resolution ADC with an averaging filter to increase the effective ADC resolution, Fig. 31.40. In this section we discuss how to interpolate between adjacent digital DAC input words to attain a large effective output resolution while reducing the required resolution of the DAC, Fig. 31.65. (To *interpolate* is to estimate a value between two known values.) We use the same symbols and terminology of the last section in this section.

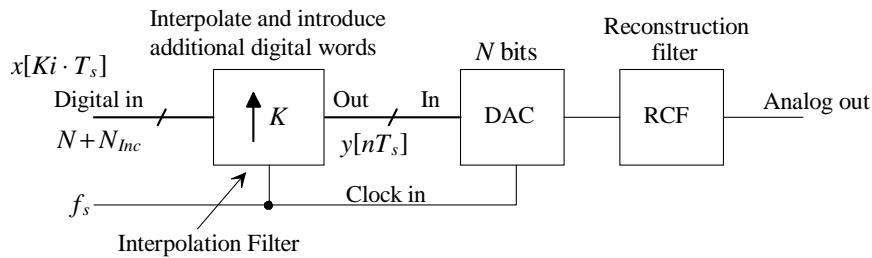


Figure 31.65 Block diagram of a DAC that uses interpolation to increase effective DAC resolution.

Also, as in the last section, Bennett's criteria must be valid. In particular, the digital word must be busy and the DAC must be linear to the final desired resolution (that is, the N -bit DAC must be linear to $N + N_{inc}$).

The Dump and Interpolate

Figure 31.66 shows the basic idea of introducing digital words in between the words coming into the interpolation circuit of Fig. 31.65. The inputs to the interpolating filter are indicated by the thicker lines in the figure. The interpolator introduces additional samples in between these inputs. If the frequency of the input samples is $2B$ then the frequency of the samples coming out of the interpolator is

$$f_s = K \cdot 2B \quad (31.112)$$

noting that, since we are using the same notation as used in the last section, the rate of words being clocked into the DAC is the same (f_s) as the rate at which the ADC was clocked in the last section.

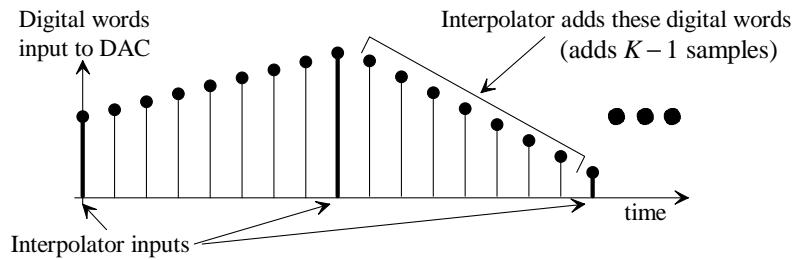


Figure 31.66 How the interpolation circuit increases the sample rate while introducing samples in between the existing samples.

If the inputs to the interpolator are $x[Ki \cdot T_s]$ and the outputs of the interpolator are $y[nT_s]$, we can write

$$y[n \cdot T_s] = x[K(i-1) \cdot T_s] + \sum_{n=K(i-1)}^{Ki-1} [n - K(i-1)] \cdot \frac{x[Ki \cdot T_s] - x[K(i-1) \cdot T_s]}{K} \quad (31.113)$$

to describe the operation of the interpolator. Rewriting this equation to show only the change between adjacent outputs results in

$$y[n \cdot T_s] - y[(n-1) \cdot T_s] = \frac{x[Ki \cdot T_s] - x[K(i-1) \cdot T_s]}{K} \quad (31.114)$$

Taking the z-transform of this equation results in

$$Y(z)(1 - z^{-1}) = X(z) \cdot \frac{1 - z^{-K}}{K} \quad (31.115)$$

or

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{K} \frac{1 - z^{-K}}{1 - z^{-1}} \quad (31.116)$$

which is the familiar transfer function for our averager presented in the last section. The implementation of our interpolator, termed a *Dump and Interpolate*, is shown in Fig. 31.67. The input words are dumped into latches which serve two purposes: (1) to store two consecutive, slow input words for generation of the incremental change in the fast output samples, and (2) to pass the interpolator input words directly to the output, through the multiplexer (MUX), every K clock cycles. The $\div K$ is implemented simply by removing the lower bits of the adders output word. As we saw with the accumulate-and-dump circuit, this implementation has practical problems that result in the need to use other implementations (which we'll discuss next).

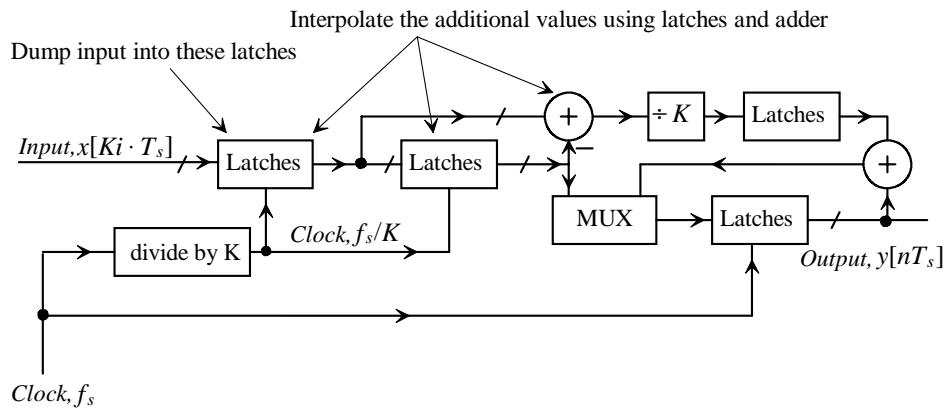


Figure 31.67 A dump-and-interpolate circuit used for interpolation and reverse averaging.

Practical Implementations of Interpolators

We can demonstrate the implementation of interpolating filters for reducing the requirements placed on the resolution of the DAC by considering the four basic cases discussed in the last section concerning the spectral content of the input signals (which are now digital signals, that is, in the last section our input signal to the ADC was analog, while in this section the input to the interpolating filter is digital).

1. The input signal bandwidth is limited to $B [=f_s/(2K)]$ and is clocked at $2B$. In this case we can perform the interpolation in one stage as seen in Fig. 31.68. Note how we have, when compared to Fig. 31.56, switched the order of the integrators and the comb filters. We are still using the slower clock, f_s/K , to generate the $K \cdot T_s$ delay in the comb filters. The rate at which the words are coming out of the interpolator is f_s ($=16B$ if $K=8$). The word size coming out of the comb filters is the same as the input word size, that is, $N+N_{inc}$. The word size increases as we move through the integrators, as we saw in Ex. 31.22. The word size coming out of the final integrating stage, prior to dropping the lower bits, is $N+N_{inc}+L \cdot \log_2 K$. The digital bits we connect to the DAC are the MSBs of this word, where the lower $N_{inc}+L \cdot \log_2 K$ bits are dropped.

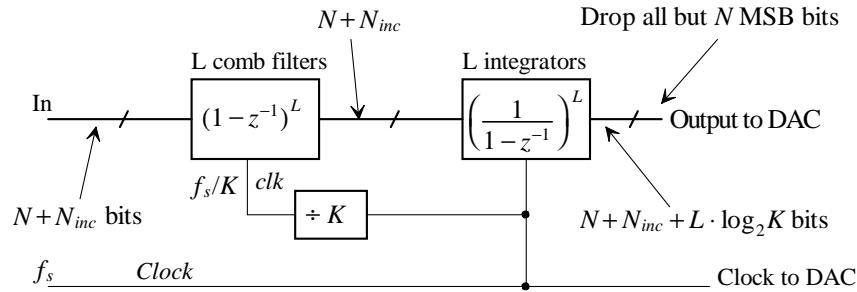


Figure 31.68 Implementation of a single stage interpolation filter. See Fig. 31.55 for handling the word size increase in the integrators.

2. The input to the interpolator/DAC has the spectrum shown in Fig. 31.58c and is clocked at $4B$. The interpolating filter has the same topology shown in Fig. 31.68 except that two registers are used in the comb filter and the divider is changed to $K/2$. Again, the DAC is clocked at f_s . The RCF together with the sinc filter used in the interpolator limits the output spectral content.

3. The input to the interpolator/DAC has the spectrum shown in Fig. 31.59b and is clocked at $8B$. The interpolating filter has the same topology as the one shown in Fig. 31.68 except that four registers are used in the comb filter (see Fig. 31.63) and the divider is changed to $K/4$. Again, the DAC is clocked at f_s . The RCF together with the sinc filter used in the interpolator limit the output spectral content.

4. Our input spectrum is shown in Fig. 31.64f and clocked into the interpolation filter at $2B$ (the general situation that results in the most relaxed requirements on the RCF). The basic, general interpolation structure is shown in Fig. 31.69. Figure 31.70 shows the spectrums at various points in this circuit. The input spectrum to the interpolation circuit is shown in Fig. 31.70a. This input is connected to a set of latches clocked at $2B$. The output of these latches is connected to the digital filter, which clocks the values in at $8B$ and has a response, as seen in Fig. 31.70b. We need to understand what's happening at this point. If we look at the output register used in Fig. 31.63, we see

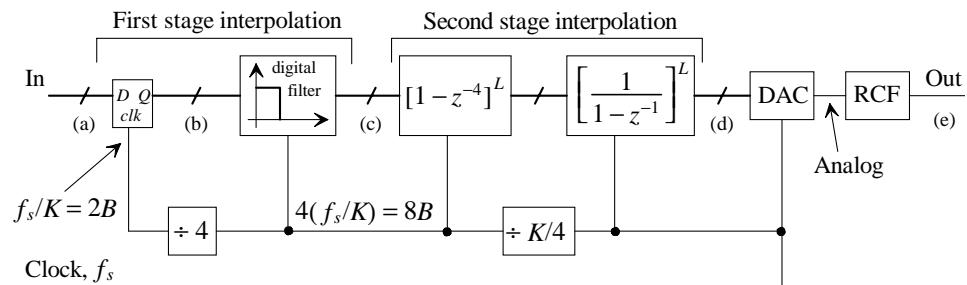


Figure 31.69 General interpolation and reverse averaging topology for an oversampled DAC.

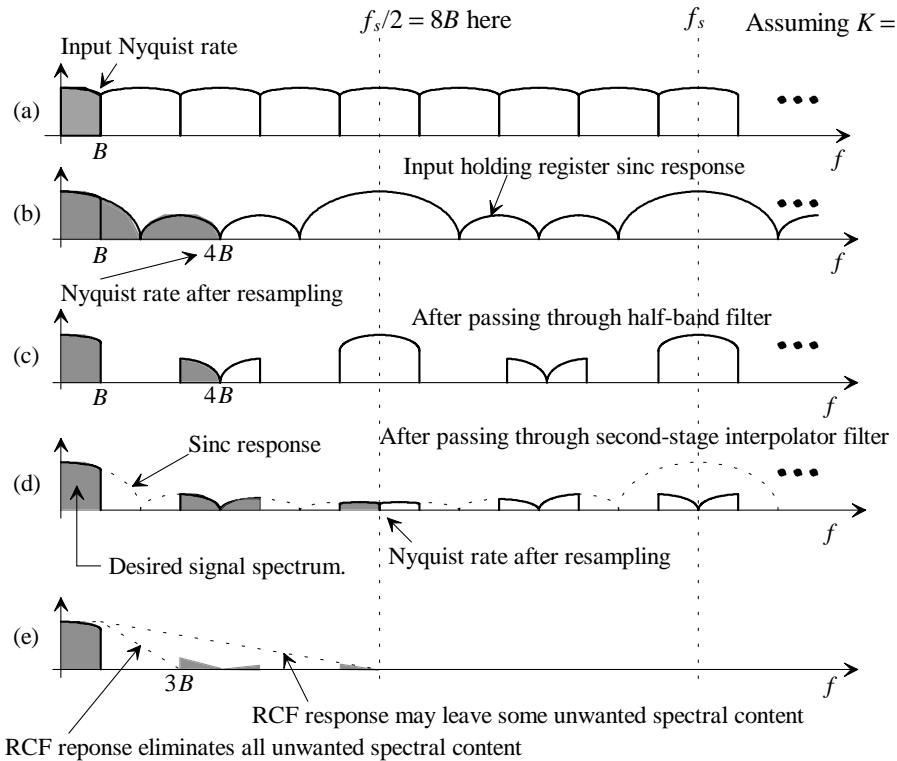


Figure 31.70 Spectrums of the resulting signals for the interpolation scheme shown in Fig. 31.69.

that we were only saving one out of every four samples coming out of the digital filter. In Fig. 31.69 we are "estimating" these samples by simply clocking each input value four times into the digital filter. Figure 31.71 shows the situation in more detail. It is desirable to determine how this input holding register affects the spectrum of the input signal. We can relate the input of the register (a set of latches) to the register's output using

$$y[nT_s] = \sum_{n=4(i-1)}^{4i-1} \frac{x[4(i-1) \cdot T_s]}{4} \quad (31.117)$$

If we look at the change between adjacent outputs we get

$$y[n \cdot T_s] - y[(n-1) \cdot T_s] = x[4i \cdot T_s] - x[4(i-1) \cdot T_s] \text{ where } n = 4(i-1) \quad (31.118)$$

which results in a transfer function of

$$H(z) = \frac{1-z^{-4}}{1-z^{-1}} \text{ (a sinc filter)} \quad (31.119)$$

Since we are only looking at one out of every four pairs of possible samples, set by the requirement that $n = 4(i-1)$ (where n and i are both integers), the value of the transfer function will actually be 1/4 of Eq. (31.119).

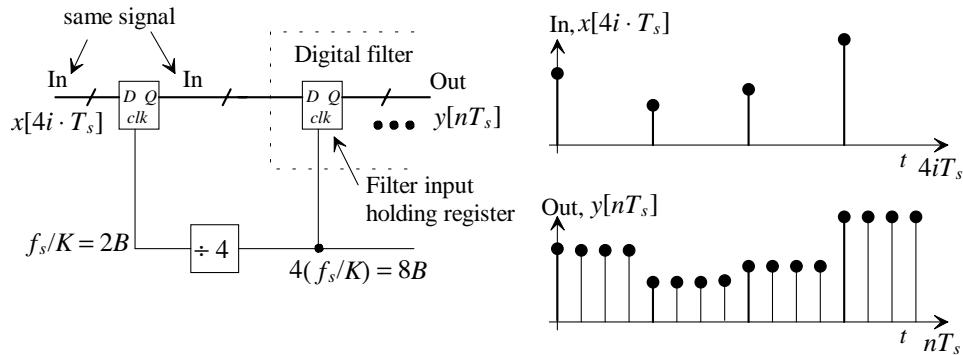


Figure 31.71 Showing effects of digital filter input holding register on the input data.

In the general interpolation scheme discussed in most digital signal processing books, zeroes, that is digital words with a value of zero, are used for the extra values when interpolating (increasing the sampling frequency). This is commonly known as *zeroes padding* the input waveform. Adding $K - 1$ zero values into a waveform, between adjacent words, results in an effective reduction in the input waveform's amplitude by K . This is easy to understand if we think of a DC input of 1 and then add three adjacent zeroes. The resulting waveform will now have an average value of 0.25. The reduction in amplitude can be compensated for by multiplying the input (or output) by four, in this example, which is simply a shift left two times. Here, in Fig. 31.69, we are avoiding the amplitude reduction by simply clocking the same input value four times (here $K = 4$). The drawback, as discussed above, is the added sinc response in the signal path.

Figure 31.70c shows the output of the digital filter which, again, is generally (but not necessarily, see Ch. 35) implemented using a half-band digital filter. At this point the digital word size is essentially the same size as the input word size. After passing through the sinc filter, the word size and word rate (frequency) increases, as indicated in Fig. 31.68 and Ex. 31.22, with a resulting attenuation of the images in the spectrum, Fig. 31.70d. Note how the first side lobe amplitude affects the amount of residual spectral content from the images at $4B$. Finally, the reconstruction filter attenuates the remaining unwanted spectral content.

The RCF deserves additional comment. Figure 31.70e shows two RCF responses. In one case the RCF limits the spectral content to $3B$. This results in elimination of any unwanted spectral content at the cost of a complex analog filter implementation. In the second case the RCF limits the spectral content to $f_s/2$ (the same response as the AAF). While resulting in a simpler filter, the spectrum still contains unwanted spectral content, in addition to the desired content between DC and B , as shown in Fig. 31.70e. The actual RCF used depends on the application. The point here is that the design of the RCF can be more challenging than the design of the AAF in a mixed-signal system.

31.2.4 Bandpass and Highpass Sinc Filters

There are many situations, especially in communication systems, where we may want to perform data conversion on a range of frequencies that doesn't extend from DC to B , as has been assumed in this book up until this point. Bandpass ADCs and DACs, for example, are becoming popular in radio communication systems. In this section, we introduce bandpass and highpass averaging sinc-shaped filters.

Cancelling Zeros to Create Highpass and Bandpass Filters

As we saw in Fig. 31.40, we can generate a lowpass filter by canceling the zero at DC in a comb filter, see also Eq. (31.79) for an intuitive explanation of the lowpass frequency response. We can generate a highpass filter by canceling a comb filter zero at $f_s/2$, as seen in the example shown in Fig. 31.72 with $K = 8$. The same equations, Eqs. (31.94) and (31.95), can be used to describe the behavior of this filter where, in the highpass response, the main lobe has shifted to $f_s/2$. Also, when looking at Fig. 31.72, remember that the frequency response of a digital filter is periodic with period f_s .

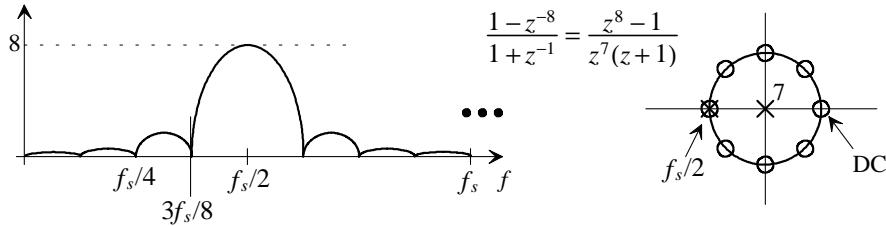


Figure 31.72 A highpass filter implementation using a comb filter.

We can generate a bandpass filter by canceling the zeroes at $f_s/4$ and $3f_s/4$, or some other frequencies, using a *digital resonator*. The general topology of the bandpass digital filter is shown in Fig. 31.73. Keeping in mind that the digital resonator is used to cancel the zeroes of the comb filter, we can write

$$H_D(z) = \frac{1}{1 - 2 \cos\left[2\pi\frac{f}{f_s}\right] \cdot z^{-1} + z^{-2}} = \frac{z^2}{z^2 - 2 \cos\left[2\pi\frac{f}{f_s}\right] \cdot z^1 + 1} = \frac{z^2}{(z - e^{+j2\pi\frac{f}{f_s}})(z - e^{-j2\pi\frac{f}{f_s}})} \quad (31.120)$$

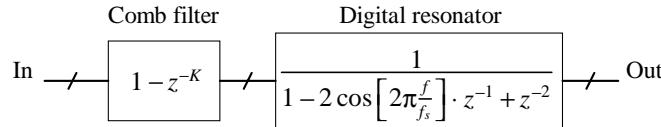


Figure 31.73 Implementing a sinc bandpass filter.

The time domain representation of this equation is

$$y[nT_s] = 2 \cos\left[2\pi\frac{f}{f_s}\right] \cdot y[(n-1)T_s] - y[(n-2)T_s] + x[nT_s] \quad (31.121)$$

It's desirable to determine at which frequencies the cosine term is an integer, a zero, or a value that results in a trivial multiplication, that is, a shift so that we can implement the bandpass filter with trivial multiplications. In other words, we want a filter that uses only delays and additions so that its implementation is simple. The first frequency we will investigate is $f_s/4$. At this frequency the cosine term is zero and the digital-resonator/comb filter transfer function (the bandpass transfer function in Fig. 31.73) reduces to

$$H(z) = \frac{1-z^{-K}}{1+z^{-2}} \quad (31.122)$$

The magnitude and z-plane response of this filter, for $K=8$, is shown in Fig. 31.74.

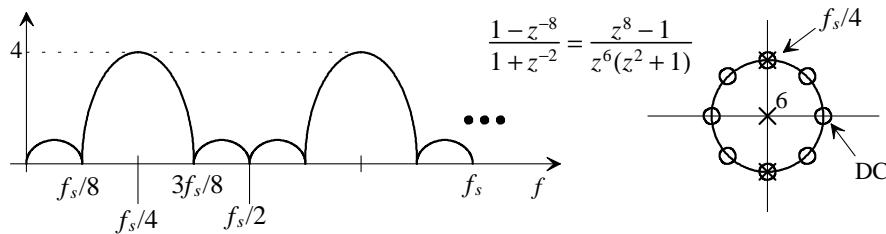


Figure 31.74 A bandpass filter implementation using a comb filter and digital resonator.

We can determine the magnitude response of Eq. (31.122) following the same procedure used to determine Eq. (31.93). The result, for the $f_s/4$ resonator, is

$$|H(f)| = \frac{\sqrt{2(1-\cos K2\pi\frac{f}{f_s})}}{\sqrt{2(1+\cos 4\pi\frac{f}{f_s})}} = \frac{\sin(K\pi\frac{f}{f_s})}{\cos(2\pi\frac{f}{f_s})} \quad K=4, 8, 12, 16, \dots \quad (31.123)$$

At the center of the passband, that is $f_s/4$, $|H(f)| = K/2$. The ratio of the main lobe to the first side lobe, on either side, is plotted in Fig. 31.75 along with the lowpass sinc filter response and is calculated using

$$\left| \frac{\text{Main lobe}}{\text{First side lobe}} \right| = \frac{K}{2} \cdot \sin \frac{3\pi}{K} \quad (31.124)$$

The cosine term in Eq. (31.121) can be set to ± 1 when $f=f_s/6$ or $f_s/3$ resulting in a bandpass filter that is easy to implement. It should be clear that with the appropriate choice of sampling frequency, number of zeroes K used in the comb filter, and value of the cosine term, many different combinations of simple bandpass filters can be implemented using these techniques.

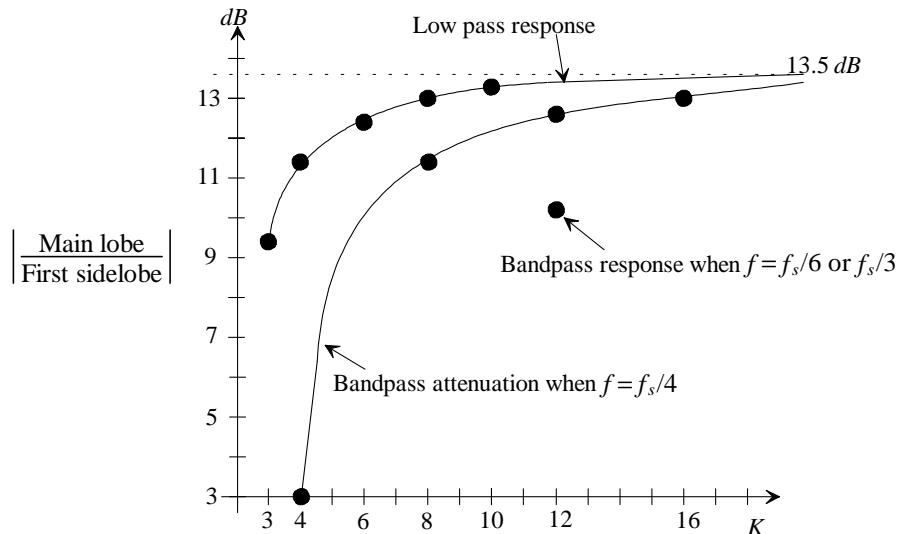


Figure 31.75 Lowpass and bandpass filter attenuation versus number of comb filter zeroes K.

The ratio of the main lobe to the first sidelobe for $f = f_s/6$ or $f_s/3$ is given, assuming $K = 12, 24, \dots$, by

$$\left| \frac{\text{Main lobe}}{\text{First side lobe}} \right| = \frac{K \sin\left(\frac{3\pi}{2K}\right) \sin\left(\frac{\pi}{3} - \frac{3\pi}{2K}\right)}{\sin\frac{\pi}{3}} = 1.15K \sin\left(\frac{3\pi}{2K}\right) \sin\left(\frac{\pi}{3} - \frac{3\pi}{2K}\right) \quad (31.125)$$

which is approximately 13.5 dB for $K = 24, 36, 48 \dots$ and 10.15 dB for $K = 12$, see Fig. 31.75.

To increase the amount of attenuation between the main lobe and the first side lobe in a bandpass filter implementation, we can cascade filter sections (as we did in the lowpass filter implementations discussed earlier). For example, cascading five $f_s/4$ bandpass filters with $K = 8$ will result in an attenuation of 57 dB. Also, note that by changing the sampling, or filter clock frequency f_s , we can easily change the bandpass filter's center frequency. A change in the clock frequency, and its selection, can easily be implemented using a counter and some control logic.

Example 31.23

Sketch the block level circuit diagram for an $f_s/4$ digital resonator.

From Eq. (31.121) the time domain representation of the $f_s/4$ resonator can be written as

$$y[nT_s] = x[nT_s] - y[(n-2)T_s]$$

The implementation is shown in Fig. 31.76. ■

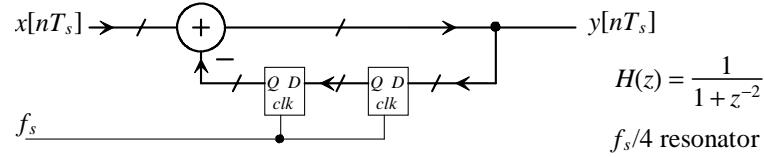


Figure 31.76 Implementation of a digital resonator.

Frequency Sampling Filters

Consider the topology of a comb filter and resonators shown in Fig. 31.77. We are feeding the output of the comb filter through the resonators (with different center frequencies) and then using the combined sum of the resulting bandpass filter responses (the sinc shapes) to build a bandpass filter. This is exactly the same as reconstructing a waveform in the time domain using an ideal RCF, as discussed in the last chapter (see Fig. 30.17), except now we are using the summation of the frequency domain sinc responses to generate a bandpass filter with a variable width. Note how every other digital resonator is subtracted rather than added to the final result. This is to account for the phase reversal between adjacent resonator outputs.

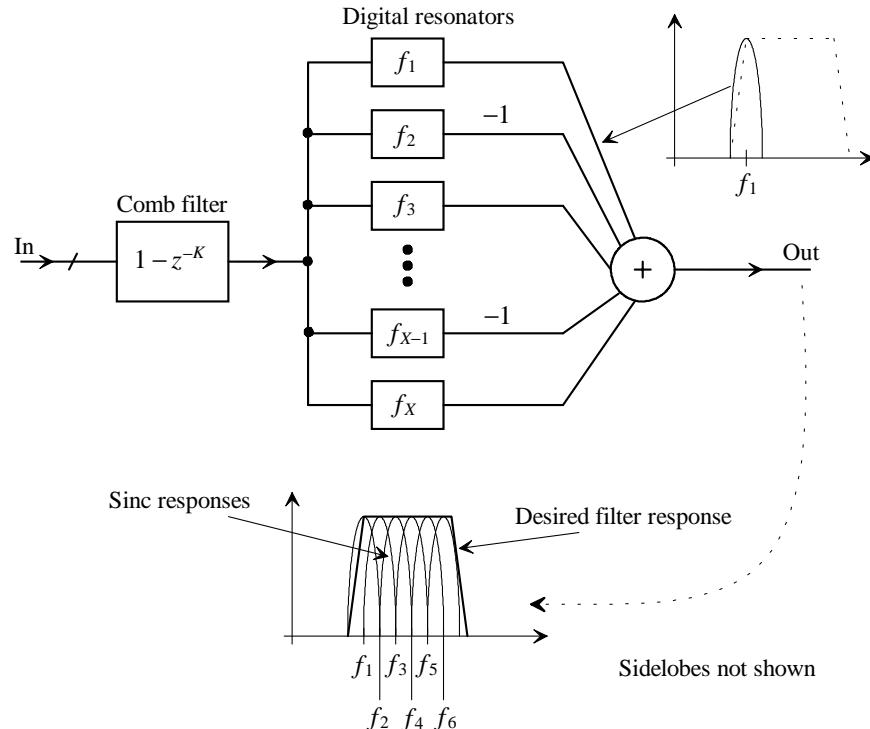


Figure 31.77 A frequency sampling filter.

31.3 Using Feedback to Improve SNR

We have seen that by averaging the outputs of an ADC, or interpolating between inputs of a DAC, the effective data converter resolution can be increased. As specified by Eq. (31.52), every doubling in (octave increase in) K (where K is the number of points averaged or the oversampling ratio) results in a 0.5-bit increase in effective resolution. An effective ADC resolution increase of 6-bits requires averaging 4,096 samples. If a 1 MHz signal bandwidth is of interest, our sampling clock frequency, f_s , will have to be 8.192 GHz!

In this section we briefly introduce the idea that feedback can be used with data converters (ADCs and DACs) to improve overall data conversion system performance (lower the amount of averaging or oversampling needed to attain a given resolution over a certain bandwidth). A topology of this nature is called a *modulator* or *coder* (for analog-to-digital conversion) or a *demodulator* or *decoder* (for digital-to-analog conversion). The complete analog-to-digital interface (a circuit block that functions as an ADC) would be made up of a modulator and a decimating filter, while the digital-to-analog interface (a circuit block that functions as a DAC) would consist of an interpolating filter and a demodulator. This can be confusing since, for example, a modulator will contain a low-resolution ADC in a feedback configuration which, together with the decimating filter, behaves like a high-resolution ADC.

31.3.1 The Discrete Analog Integrator

An analog building block that we will find useful in implementing our data converters, using feedback, is the discrete analog integrator, DAI, shown in Fig. 31.78. Here we're *assuming* the reader is familiar with the material presented back in Ch. 27 covering switched-capacitor circuits (for example, the reader is familiar with the operation of parasitic insensitive integrators). The two clock signals, ϕ_1 and ϕ_2 , form nonoverlapping clock signals (see Fig. 14.5 in Ch. 14). Also, we are assuming the common mode voltage, V_{CM} , falls halfway between the mixed-signal system's high- and low-reference voltages.

Table 31.2 shows the various relationships between the possible inputs and outputs for the DAI of Fig. 31.78. Let's derive the input/output relationships for the most general situations where both v_1 and v_2 are the inputs.

Input	Output connected to ϕ_1	Output connected to ϕ_2
$v_1 = \text{input}$ and $v_2 = V_{CM}$	$\frac{z^{-1}}{1-z^{-1}} \cdot \frac{C_I}{C_F}$	$\frac{z^{-1/2}}{1-z^{-1}} \cdot \frac{C_I}{C_F}$
$v_2 = \text{input}$ and $v_1 = V_{CM}$	$\frac{-z^{-1/2}}{1-z^{-1}} \cdot \frac{C_I}{C_F}$	$\frac{-1}{1-z^{-1}} \cdot \frac{C_I}{C_F}$
v_1 and v_2 are both inputs	$\frac{V_1(z) \cdot z^{-1} - V_2(z) \cdot z^{-1/2}}{1-z^{-1}} \cdot \frac{C_I}{C_F}$	$\frac{V_1(z) \cdot z^{-1/2} - V_2(z)}{1-z^{-1}} \cdot \frac{C_I}{C_F}$

Table 31.2 Discrete analog integrator input/output relationships
(see also Eqs. [31.136] and [31.137]).

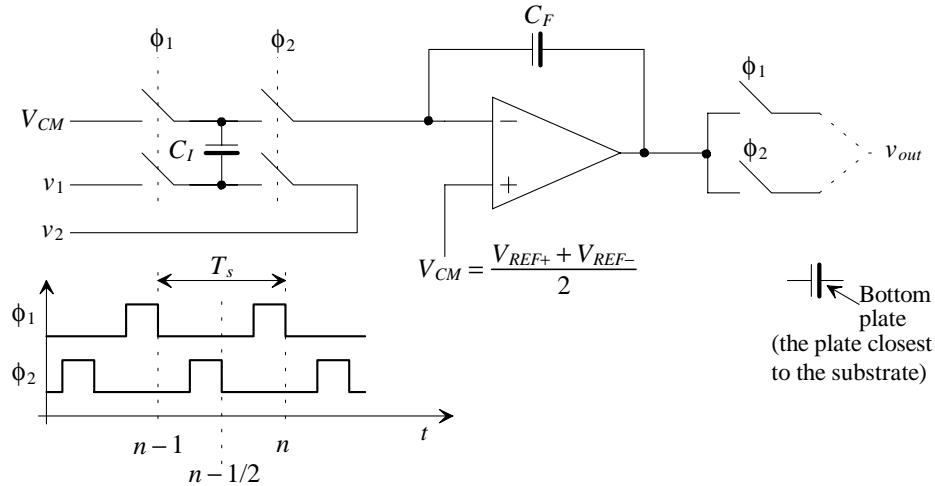


Figure 31.78 Schematic diagram of a discrete analog integrator (DAI).

To begin, let's assume the output of the DAI is connected to the op-amp through the ϕ_1 switch. When the ϕ_1 switches are closed (ϕ_1 is high) at $n - 1$ (the instance when the switches shut off), the charge stored on C_I is

$$Q_1 = C_I(V_{CM} - v_1[(n-1)T_s]) \quad (31.126)$$

and the output of the integrator is $v_{out}[(n-1)T_s]$. When the ϕ_2 switches turn on the charge stored on C_I becomes

$$Q_2 = C_I(V_{CM} - v_2[(n-1/2)T_s]) \quad (31.127)$$

remembering that the op-amps holds its noninverting input terminal at V_{CM} . The difference in these charges, $Q_2 - Q_1$, is transferred to the op-amp's feedback capacitor resulting in an output voltage change. This change can be written as

$$(v_{out}[nT_s] - v_{out}[(n-1)T_s])C_F = C_I(v_1[(n-1)T_s] - v_2[(n-1/2)T_s]) \quad (31.128)$$

or writing this equation in the z-domain results in

$$V_{out}(z)(1 - z^{-1}) = \frac{C_I}{C_F}(V_1(z) \cdot z^{-1} - V_2(z) \cdot z^{-1/2}) \quad (31.129)$$

The transfer function of the DAI with the output connected to the ϕ_1 switches is then

$$V_{out}(z) = \frac{C_I}{C_F} \cdot \frac{V_1(z) \cdot z^{-1} - V_2(z) \cdot z^{-1/2}}{1 - z^{-1}} \quad (31.130)$$

Similarly, if we connect the output through the ϕ_2 switches (the edges we label n in Fig. 31.78 shift in time by $T_s/2$) we can write

$$Q_1 = C_I(V_{CM} - v_1[(n-1/2)T_s]) \quad (31.131)$$

$$Q_2 = C_I(V_{CM} - v_2[nT_s]) \quad (31.132)$$

and

$$(v_{out}[nT_s] - v_{out}[(n-1)T_s])C_F = C_I(v_1[(n-1/2)T_s] - v_2[nT_s]) \quad (31.133)$$

The transfer function of the DAI with the output connected to the ϕ_2 switches is then

$$V_{out}(z) = \frac{C_I}{C_F} \cdot \frac{V_1(z) \cdot z^{-1/2} - V_2(z)}{1 - z^{-1}} \quad (31.134)$$

Note that if $V_2(z) = V_{CM}$, this equation can be written as

$$H(z) = \frac{V_{out}(z)}{V_1(z)} = \frac{C_I}{C_F} \cdot \frac{z^{-1/2}}{1 - z^{-1}} \quad (31.135)$$

which has a frequency response, $H(f)$, shown in Fig. 31.34. Note that the factor C_I/C_F simply scales the amplitude response. If this factor is unity then the magnitude response, as shown in Fig. 31.34, is 0.5 at $f_s/2$. The $z^{-1/2}$ term in the numerator simply modifies the phase response of the DAI (delaying the output by $T_s/2$ or -180 degrees) and has no effect on the magnitude response. (We'll discuss this more in a moment.) Note that at this point we could discuss the frequency responses of the transfer functions given in Table 31.2. However, we would see that the discussions and results earlier in the chapter for the digital integrator would apply to the DAI with no, or little, modifications.

Example 31.24

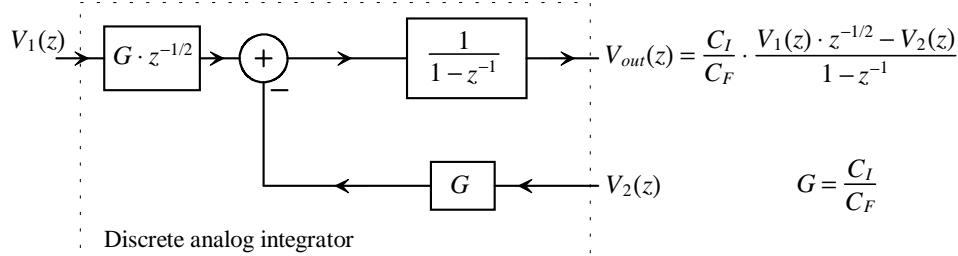
Determine the transfer function of the DAI of Fig. 31.78 *without* the switches on the output of the op-amp.

Reviewing Fig. 31.78 we see that charge is transferred to the feedback capacitor only when the ϕ_2 switches are closed. Therefore, the output only changes states during the time interval when the ϕ_2 switches are closed. The transfer function of the DAI, when no switches are used on the output of the op-amp, is given by Eq. (31.134). Using the ϕ_1 switches simply adds a half clock cycle delay, $z^{-1/2}$, to the integrator's transfer function (instead of the output changing with the rising edge of ϕ_2 , the output changes one-half cycle later on the rising edge of ϕ_1). ■

A Note Concerning Block Diagrams

As we draw block diagrams describing our modulator topologies in this chapter and the next we often show a circuit like the one shown in Fig. 31.79. The summation, gain, and integrating blocks are implemented with a single switched-capacitor DAI having the transfer function given by Eq. (31.134). The gain, G , of the DAI is set by the ratio of capacitors as indicated in the figure. It's important to realize that this circuit is entirely analog and is interfaced to, in general, both ADCs ($V_{out}[z]$ is connected to the input of an ADC) and DACs ($V_z[z]$ is connected to the output of a DAC).

It should be clear from both Fig. 31.79 and Table 31.2 that many different combinations of discrete analog building blocks are possible. Figure 31.80 shows two

**Figure 31.79** Block diagram of a DAI.

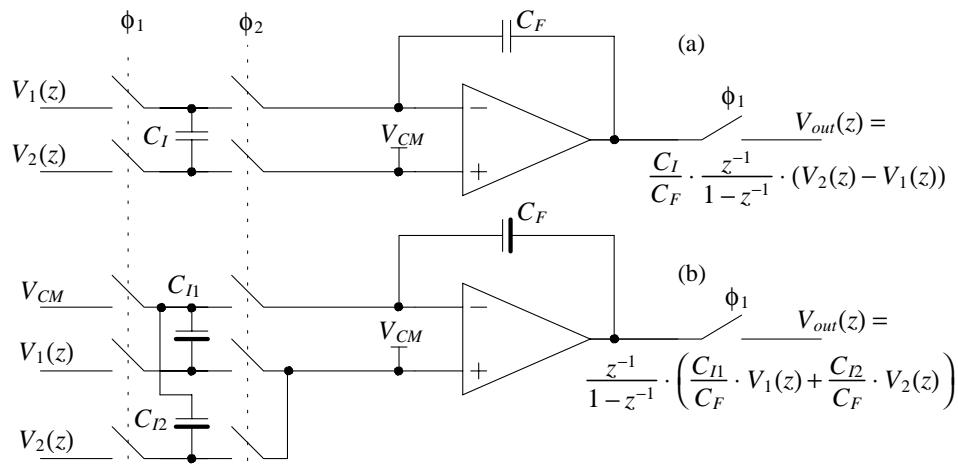
other possibilities. In part (a) the capacitors used have the same parasitic capacitance on each plate (see the lateral capacitor in Fig. 33.11 for example), so there is no benefit to using a bottom-plate insensitive topology (Fig. 31.78). The transfer function of this DAI is

$$V_{out}(z) = \frac{C_I}{C_F} \cdot \frac{z^{-1}}{1-z^{-1}} \cdot (V_2(z) - V_1(z)) \quad (31.136)$$

noting each input signal sees the same delay, i.e., z^{-1} when the outputs are connected through ϕ_1 controlled switches and $z^{-1/2}$ delay when no switches or ϕ_2 controlled switches are used. If the integrator inputs must see the same delay and the capacitors available have asymmetric parasitic capacitance, the topology of Fig. 31.80b can be used. Its transfer function is

$$V_{out}(z) = \frac{z^{-1}}{1-z^{-1}} \cdot \left(\frac{C_{I1}}{C_F} \cdot V_1(z) + \frac{C_{I2}}{C_F} \cdot V_2(z) \right) \quad (31.137)$$

noting the input signals can be scaled independently (a useful feature in filter design).

**Figure 31.80** Other forms of DAIs.

31.3.2 Modulators

The basic topology of a feedback modulator or coder is shown in Fig. 31.81. Depending on the circuit blocks used for $A(z)$ and $B(z)$ feedback modulators can be separated into two categories: *predictive modulators* and *noise-shaping modulators* [10].

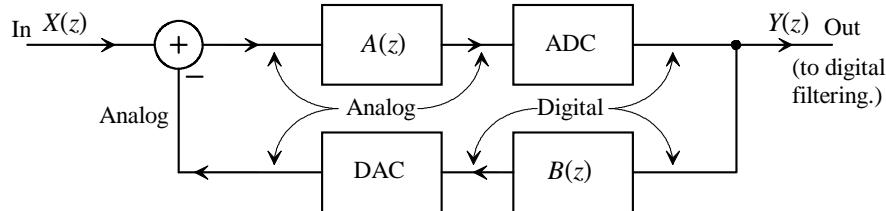


Figure 31.81 Block diagram of a feedback modulator.

Predictive modulators (a.k.a. predictive coders), an example being delta-modulation, attempt to feed back an analog signal with the same value as the input signal. This drives the output of the summer to zero, reducing the required input range of the ADC and, possibly, the quantization error introduced by the ADC. *Predictive modulators effectively output the change in the input signal over time*. Noise-shaping modulators, an example being sigma-delta-modulation (also known as delta-sigma-modulation), on the other hand, feed back (and output) the average value of the input signal. This signal can be filtered (averaged) to reduce the accuracy required of the analog circuit components. *Noise-shaping modulators effectively output the average of the input signal over time*. In a noise-shaping data converter the averaging and decimating filter, as discussed earlier, is connected to the output of the modulator. Because of the averaging used in noise-shaping modulators, the analog components, in the forward path of Fig. 31.81, require less accuracy. However, the DAC's output, in the feedback path (which is subtracted from the input), doesn't experience the averaging so, once again, the DAC must be linear to the final desired resolution of the data converter. DAC linearity concerns have led to the use of a single-bit DAC (an inverter, see Ch. 29), in many noise-shaping data converter applications. The one-bit DAC is inherently linear. (Two output points determine a line!) Because of the relaxed requirements placed on the analog circuit components, we will concentrate the next chapter, in detail, on noise-shaping topologies for both ADCs and DACs. Notice that both predictive and noise-shaping modulators utilize oversampling.

To understand these statements in more detail, let's use the additive quantization noise model for the ADC developed in Ch. 30 and shown in Fig. 30.56. Figure 31.82 shows Fig. 31.81 redrawn using this model where the quantization noise is represented in the z -domain by $E(z)$. We can relate the inputs (the wanted input signal and the unwanted quantization noise) to the output of the feedback modulator by

$$Y(z) = \underbrace{\frac{A(z)}{1 + A(z) \cdot B(z)}}_{\text{Signal transfer function, } STF(z)} \cdot X(z) + \underbrace{\frac{1}{1 + A(z)B(z)}}_{\text{Noise transfer function, } NTF(z)} \cdot E(z) \quad (31.138)$$

In a predictive modulator the feedback filter, $B(z)$, has a large gain so that, ideally, the fed back signal equals the input signal. If $A(z) = 1$ (a wire), then both the STF (signal transfer function) and the NTF (noise transfer function) have a value of, approximately, $1/B(z)$. Recovering the input signal requires passing the output of the predictive modulator through an analog filter with a transfer function of precisely $B(z)$ (noting that $B[z]$ is a digital filter in the modulator of Fig. 31.82). The required precision of the analog filter (the matching between the filter in the modulator and the filter in the demodulator) limits the attainable resolution when using predictive modulators. Notice that both the input signal and the quantization noise experience the same spectral shaping (spectral discrimination is absent in a predictive modulator). Also note that the name "predictive" comes from the modulator attempting to predict the input signal in order to drive the output of the summer to zero. If the prediction is perfect, the signal that is fed back exactly matches the input signal.

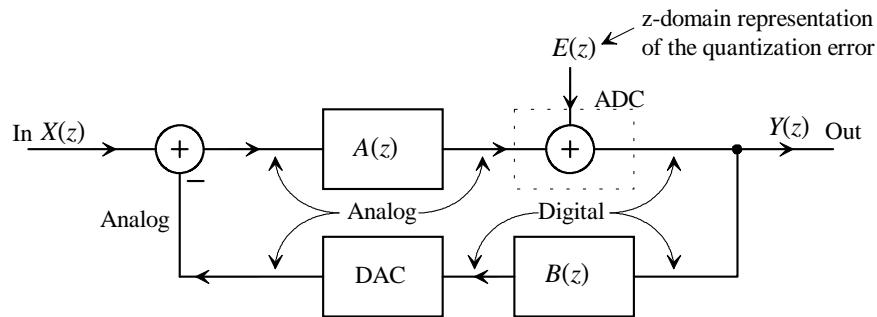


Figure 31.82 Block diagram of a feedback modulator.

In a noise-shaping modulator the gain of the forward path, $A(z)$, is large in the signal bandwidth so that the STF is approximately unity (assuming $B[z] = 1$). The NTF , on the other hand, will approach zero, ideally, in the bandwidth of interest. Note that the signal spectrum passes through the modulator essentially unchanged, while the quantization noise spectrum is shaped (and thus the name *noise-shaping*). No precision filter or analog components are required, as discussed earlier, except, perhaps, for the DAC in the feedback path of the modulator. We'll see in the next chapter that if $A(z)$ is an integrator, the quantization noise is pushed to higher frequencies so that it can be removed with the averaging filter. This is a very important concept, as a noise-shaping modulator *does not* reduce the quantization noise to attain higher resolutions, but rather pushes the noise to frequencies outside of the signal bandwidth of interest.

REFERENCES

- [1] R. J. Baker, H. W. Li, and D. E. Boyce, *CMOS: Circuit Design, Layout, and Simulation*, Wiley-IEEE, 1998. ISBN 0-7803-3416-7
- [2] L. W. Couch, *Modern Communication Systems: Principles and Applications*, Prentice-Hall, 1995. ISBN 0-02325286-3
- [3] S. Haykin, *An Introduction to Analog and Digital Communications*, John Wiley and Sons, 1989. ISBN 0-471-85978-8
- [4] P. A. Lynn and W. Fuerst, *Introductory Digital Signal Processing*, Second Edition, John Wiley and Sons, 1998. ISBN 0-471-97631-8
- [5] E. P. Cunningham, *Digital Filtering: An Introduction*, John Wiley and Sons, 1995. ISBN 0-471-12475-3
- [6] R. K. Hester, *Introduction to Oversampled Data Conversion*, Notes from a tutorial at the 1995 International Solid-State Circuits Conference (ISSCC-95).
- [7] W. R. Bennett, "Spectra of Quantized Signals," *Bell System Technical Journal*, Vol. 27, pp. 446-472, July 1948.
- [8] J. C. Candy and G. C. Temes (eds.), *Oversampling Delta-Sigma Data Converters*, IEEE Press, 1992. ISBN 0-87942-285-8
- [9] S. R. Norsworthy, R. Schreier, and G. C. Temes (eds.), *Delta-Sigma Data Converters: Theory, Design, and Simulation*, IEEE Press, 1996. ISBN 0-7803-1045-4
- [10] S. K. Tewksbury and R. W. Hallock, *Oversampled, Linear Predictive and Noise-Shaping Coders of Order N>1*, IEEE Trans. Circuits and Sys., Vol. CAS-25, pp. 436-447, July 1978.

LIST OF SYMBOLS/ACRONYMS

- ACF - Autocorrelation Function, see Eq. (31.30)
- ADC - Analog-to-Digital Converter
- B* - Bandwidth of the input signal, see Eq. (31.69)
- DAC - Digital-to-Analog Converter
- dBc - Decibels with respect to the carrier, see Eq. 31.9
- DR - Dynamic Range, see Eq. (31.10)
- ΔT_s - Peak-to-peak amount of clock jitter
- ΔV_s - Uncertainty in the sampled voltage
- f_{clk} - Clock frequency. Also, sometimes called sampling frequency, f_s
- f_{in} - Input sinewave frequency

f_n - Nyquist frequency, which is $f_s/2$. Sometimes also called the folding frequency.

f_{res} - Resolution of a DFT

f_s - Sampling frequency. Also, sometimes called clock frequency, f_{clk} , or Nyquist rate.

$H(f)$ - Transfer function of a system

$H(z)$ - Z-Domain representation of $H(f)$

K - Oversampling factor or number of points averaged. See Eq. (31.22) or Eq. (31.51)

L - Order of sinc averaging filter

LSB - Least Significant Bit, see Eq. (31.2)

M - Order of a noise-shaping modulator (see Ch. 32.)

N - Ideal data converter resolution (number of bits)

N_{eff} - Effective number of bits, see Eq. (31.5)

N_{Final} - Final data converter resolution after averaging, see Eq. (31.74)

N_{Inc} - Increase in data converter resolution, see Eq. (31.53)

N_{Loss} - Number of bits lost because of sampling jitter. see Eq. (31.17)

PDF - Probability Density Function, see also $\rho(t)$

P_{AVG} - Total average power in a waveform

$P_{in}(f)$ - Power Spectral Density (PSD), see Eq. (31.35)

$P_{jitter}(f)$ - PSD of the sampling error voltage due to jitter, see Eq. (31.49)

$P_{osc}(f)$ - PSD of the output of an oscillator, see Eq. (31.50)

PSD - Power Spectral Density

$P_{Qe}(f)$ - Quantization noise power spectral density

ppm - Parts per million, a multiplier of 10^{-6}

L - Number of sections used in an averaging filter

$R_{in}(t)$ - Autocorrelation function, see Eq. (31.30)

$\rho(t)$ - Probability Density Function, PDF

σ - Standard deviation or square root of the variance

σ^2 - Variance of a PDF, see Eq. (31.44)

$R_{in}(t)$ - Autocorrelation function, see Eq. (31.30)

RMS - Root Mean Square

SFDR - Spurious Free Dynamic Range, see Eq. (31.9)

SINAD - SInal-to-Noise And Distortion. Same as SNDR.

SNR - Signal-to-noise ratio

$\text{SNR}_{\text{ideal}}$ - Ideal signal-to-noise ratio for a data converter, see Eq. (31.1)

SNR_{meas} - Measured signal-to-noise ratio for a data converter, see Ex. 31.1

SNDR - Signal-to-Noise plus Distortion Ratio, see Eq. (31.7)

T_s - Sampling period

$V_{\text{DFT}}(f)$ - Discrete Fourier transform of V

V_{LSB} - The voltage weighting of a least-significant bit, see Eq. (31.2)

V_p - Peak value of a sinewave

$V_{\text{in}}(t)$ - Input signal, see Eq. (31.29)

$V_{Qe}(f)$ - Power spectral density of the quantization noise, see Eq. (31.54)

$V_{Qe,RMS}$ - Root mean squared quantization noise, see Ch. 30.

$V_{Qe+D,RMS}$ - Root mean squared quantization noise and distortion, see Eq. (31.7)

$V_{\text{REF}+}$ - Positive reference voltage used in a data converter

$V_{\text{REF}-}$ - Negative reference voltage used in a data converter

V_{RMS} - An RMS voltage

\bar{y} - Average value of y , see Eq. (31.44)

z - Defined as $e^{j2\pi f_s} = e^{j2\pi f T_s}$

QUESTIONS

- 31.1** Develop an expression for the effective number of bits in terms of the measured signal-to-noise ratio if the input sinewave has a peak amplitude of 40% of ($V_{\text{REF}+} - V_{\text{REF}-}$).
- 31.2** Determine a data conversion system's SNR if the measured $V_{Qe,RMS}$ is 1 mV and the maximum peak-to-peak amplitude of an input sinewave is 1 V.
- 31.3** Repeat Ex. 31.2 if the sampling frequency is increased to 200 MHz. Does the SNR change?
- 31.4** Why is the amplitude of the tone at 45 MHz in the DAC output spectrum shown in Fig. 31.3b smaller than the amplitude of the ADC input signal? What is the origin of the noise added to the DAC output signal in Fig. 31.3b?
- 31.5** When using Eq. (31.8) what is the assumed ADC input signal? Put your answer in terms of the ADC reference voltages.
- 31.6** Describe in your own words the difference between specifying SNR and SNDR.

- 31.7** Suppose a perfectly stable clock is available (ΔT_s is zero in Eq. [31.12]). Would we still have a finite aperture window if the clock has a finite rise time? Describe why or why not?
- 31.8** How do the number of bits lost because of aperture jitter change with the frequency of an ADC input sinewave? If the ADC input is a DC signal is aperture jitter a concern? Why?
- 31.9** Show the time domain signal that generates the spectrum shown in Fig. 31.10. Verify in the time domain that the signal's rising and falling edges do indeed vary from their ideal positions.
- 31.10** Describe in your own words the problems with simulating clock jitter using SPICE.
- 31.11** What does the autocorrelation function (ACF) tell us about a signal? What is the ACF of a 1 V DC signal. Show the simple calculations leading up to your answer.
- 31.12** Plot the power spectral density of a sinewave. From this plot show how to determine the average and RMS values of the sinewave. Show the procedure for both one-sided and two-sided spectrums.
- 31.13** Sometimes the average power specified by Eq. (31.37) is termed *total average normalized* power of a signal. Why?
- 31.14** When WinSPICE generates a plot from a DFT the units on the y-axis are volts peak (the peak value of a sinewave at a given frequency). How do we change this plot into RMS voltages, voltage spectral density, and power spectral density vs. frequency?
- 31.15** Repeat Ex. 31.12 if the sinewaves are first sampled.
- 31.16** Suppose the jitter in a clock signal can be characterized using the PDF shown in Fig. 31.12. Further if $\Delta T_s = 100$ ps estimate the RMS value of clock jitter, standard deviation, and variance of the jitter.
- 31.17** Suppose that a noise voltage has the PDF shown in Fig. 31.12. If the maximum voltage deviation from the ideal value is 10 μ V estimate the RMS value of the noise (the standard deviation) and the noise power (the variance).
- 31.18** Repeat question 31.17 if the noise voltage has a Gaussian PDF as seen in Fig. 31.13.
- 31.19** Repeat Ex. 31.1 if we want to include an error from sampling jitter, $P_{AVG,jitter}$ of 1 μ W.
- 31.20** If a DC signal is input to a data conversion system, is Eq. (31.51) valid? Name three conditions on the input signal in order for this equation to be valid.

- 31.21** Suppose the standard deviation of the quantization noise in a data conversion system is 1 mV. Using Eq. (31.56) plot the PSD of the quantization noise. Comment on the assumption that the noise power is limited to the Nyquist frequency. Does this result in an over- or underestimate for the actual noise power in the spectrum of interest?
- 31.22** Show why averaging two 8-bit words, as seen in Fig. 31.19, must result in a 9-bit word. (Why isn't the sum of the two words divided by two [the average] another 8-bit word?)
- 31.23** Why must Bennett's criteria be valid for the averaging to reduce the quantization noise in Fig. 31.19? Give an example where averaging will not reduce quantization noise.
- 31.24** Show a figure similar to Fig. 31.20 where an input sinewave with a frequency of $f_s/4$ is sampled at f_s . Show that the magnitude of the resulting fundamental (indicating that the sinewave lies in the frequency range of DC to f_n) sinewave is $\sqrt{2}$, as indicated in Fig. 31.22.
- 31.25** Assuming Eq. (31.68) is valid rederive Eq. (31.4) including the effects of averaging K ADC output samples. Is Eq. (31.4) or the equation derived here valid for a slow or DC input signal? Comment on why or why not.
- 31.26** Assuming Bennett's criteria are valid, does averaging ADC outputs (or DAC inputs) put any restrictions on the bandwidth of the input signal? Why? Give an example.
- 31.27** Comment on the statement "The factor of 2 in the magnitude response of Fig. 31.22 at low-frequencies simply indicates that the digital word length increases by one bit."
- 31.28** What is the magnitude response of $z^{-2} + z^{-3}$.
- 31.29** Repeat Ex. 31.15 if 16 ADC outputs are averaged, that is, $K = 16$.
- 31.30** How accurate does an 8-bit ADC have to be in order to use a digital filter to average 16 output samples for a final output resolution of 10-bits (see Eq. [31.53]). Assume the ideal LSB of the 8-bit converter is 10 mV. Your answer should be given in both mV and % of the full-scale.
- 31.31** If a DC signal is applied to a data converter can a digital averaging filter be used to increase the system's resolution? What about if a dither signal is added to the DC input? Use simple time domain drawings to illustrate your answers.
- 31.32** Name three characteristics of all digital filters.
- 31.33** Plot Eq. (31.59) on a z-plane. Using this plot show how to graphically determine the magnitude and phase responses shown in Fig. 31.22.
- 31.34** The magnitude response shown in Fig. 31.34 becomes infinite as the input signal approaches DC. Since the filter is digital, what is the maximum output of the filter?

- 31.35** Show that the peak (+127) and valley (-128) amplitudes of the two's complement signals in Fig. 31.37 sum to -1.
- 31.36** Summarize the method of changing a number from binary offset to two's complement. Demonstrate addition and subtraction using two's complement numbers. Show how, in two's complement, 8, 33, and 111 sum to 152. Assume a 10-bit word size.
- 31.37** Suppose a digital filter sums 16 inputs and then outputs the total. If the filter is clocked at 100 MHz, plot the magnitude response of the filter.
- 31.38** Comment on the benefits and drawbacks of using an averaging filter with and without decimation.
- 31.39** Verify the z-domain function specified by Eq. (31.100) has a frequency response given by Eq. (31.101). How are the typical input and output signals in the time domain related for this filter?
- 31.40** What is the magnitude response of $(1 - z^{-1})^3$. Sketch a block diagram implementation for this filter.
- 31.41** Resketch Fig. 31.53 if, in each transfer function $H(z)$, a pole is added at DC.
- 31.42** Show the problem with not using a MUX at the input of the adders in Fig. 31.55.
- 31.43** Is it possible for the accumulate-and-dump circuit to output a spectrum with aliasing if the input signal is bandlimited to f_s ? Why or why not?
- 31.44** In the discussions in this chapter we assumed the digital signals are much larger than an LSB of a data converter. What happens if this is not the situation for the sinc averaging filter?
- 31.45** Is it possible to decimate a digital waveform down to $2B$ and then later, with some other hardware or software digital filter, remove all of the aliased signals from the desired signal?
- 31.46** Suppose the waveform shown in Fig. 31.66 is the input to a decimator. If $K = 8$, what would the output of the decimator look like? Use integers to illustrate your understanding.
- 31.47** Suppose a digital word is clocked into a hold register and held there for eight clock cycles before another word is clocked into the hold register. Is this similar to the analog sample-and-hold? If the sampling rate (clock frequency) is increased by a factor of 8 after the hold register what kind of digital filter can we think of the hold register as being?
- 31.48** Show that the digital resonator of Fig. 31.76 can be modified if we add a multiplier to the circuit, so that Eq. (31.120) can be implemented.

- 31.49** It is more correct to write our DAI continuous time input signals in Fig. 31.78 as

$$v_1(t) + V_{CM} \text{ and } v_2(t) + V_{CM}$$

Knowing this rederive Eq. (31.130).

- 31.50** Repeat question 31.49 for Eq. (31.134).

- 31.51** Using the results from question 31.49, derive the transfer function, Eq. (32.139), for the circuit shown in Fig. 32.92 (in the next chapter).

- 31.52** Show the detailed derivation of Eq. (31.138).

- 31.53** Summarize the advantages and disadvantages of predictive and noise-shaping data converters.

Chapter

32

Noise-Shaping Data Converters

In this chapter we discuss the design of noise-shaping (NS) data converters. Our approach will be to develop NS theory along with SPICE behavioral models to illustrate, using simulations, the operation of NS ADCs and DACs. Our goals are to discuss the fundamentals of NS data converter design and to put a framework together for SPICE simulations. Having a simulation framework available will allow us to (1) perform a behavioral simulation using nearly ideal components to determine fundamental performance limitations of a particular NS converter topology, and (2) replace behavioral models with actual, MOSFET-based circuits in steps to design and simulate the operation of a NS data converter in stages. While we can replace the behavioral models with MOSFET-based circuits, we will delay this discussion (MOSFET-based circuit design in a submicron process) until the next chapter. This chapter will focus on NS theory and examples, using simulations, to illustrate the use of the theory.

32.1 Noise-Shaping Fundamentals

In this section we develop SPICE behavioral models to illustrate NS data converter design and then we present the theory behind the design of first- and second-order NS data converters using single-bit DACs and ADCs.

32.1.1 SPICE Models

Because data converters employing averaging, such as an NS data converter, can require a significant number of samples for meaningful operation (the simulation time may be relatively long), we need to be careful not to develop simulation models that are inefficient or too complex. At the same time it is desirable to have models complex enough to include the nonideal effects that occur in the actual circuits used. For example, an op-amp will have finite gain and an offset voltage while the switches will have nonzero "on" resistance.

In the material that follows we attempt to develop models that are robust, including fundamental limitations of the circuits used, while at the same time attempting to generate simple models for fast simulations.

Nonoverlapping Clock Generation and Switches

In this chapter, as we did in Chs. 30 and 31, we assume $VDD = 1.5$ V (the positive power supply voltage), $VSS = 0$ V (the negative power supply voltage), $V_{REF+} = 1.5$ V (the positive data converter reference voltage), $V_{REF-} = 0$ V (the negative data converter reference voltage), and $f_s = f_{clk} = 100$ MHz (the sampling, or clock frequency, of the data converter). The SPICE pulse statements used to generate two 100 MHz nonoverlapping clocks can be written as

```
Vphi1 phi1 0 DC 0 Pulse 0 1.5 0 200p 200p 4n 10n
Vphi2 phi2 0 DC 0 Pulse 0 1.5 5n 200p 200p 4n 10n
R2 phi1 0 1MEG
R3 phi2 0 1MEG
```

where the resistors ensure that the clocks are not floating (not the only elements connected to the nodes phi1 and phi2 as the clocks may be used exclusively to control switches in a simulation). The statements used to set up the power supply voltages, reference voltages (if used), common-mode voltage, and switch trip points can be written as

```
VDD VDD 0 DC 1.5
Vtrip Vtrip 0 DC 0.75
VCM VCM 0 DC 0.75
VREFP VREFP 0 DC 1.5
VREFMVREFM0 DC 0
```

The trip voltage is used in simulating the operation of the switches to indicate when the switch should be opened or closed. Figure 32.1a shows the use of the basic switch in SPICE. When ϕ_1 (ϕ_1) is above the trip voltage (0.75 V here), the S1 switch is closed. When the node phi2 is above the trip voltage, S2 is closed. The SPICE statements specifying the operation of these switches, in the manner described, are

```
S1 1 2 phi1 Vtrip switmod
S2 2 3 phi2 Vtrip switmod
.model switmod SW RON=1k
```

The parameter RON can be used to model the switches' on resistance as shown in Fig. 32.1b. This may be useful when simulating finite settling time effects in a data converter.

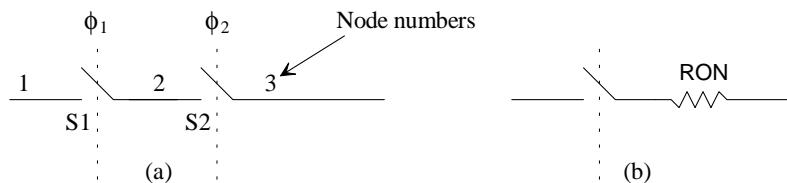


Figure 32.1 Using a switch in SPICE.

Op-Amp Modeling

Behavioral modeling of op-amps could take up an entire chapter by itself. Here we introduce a trivial model that is easily modified to account for real op-amp imperfections. Figure 32.2 shows the basic op-amp symbol and a voltage-controlled-voltage-source used to simulate the operation of an op-amp. The SPICE statement that specifies the op-amp is

```
Ein      3      0      2      1      100MEG
```

where the open-loop gain of the op-amp is 100 million.



Figure 32.2 Simple SPICE op-amp model.

Example 32.1

Determine and simulate the gain of the circuit shown in Fig. 32.3.

This circuit is our discrete analog integrator (DAI) shown in Fig. 31.78 with the v_2 input connected to V_{CM} . The transfer function of this circuit in the z-domain is

$$H(z) = \frac{z^{-1}}{1 - z^{-1}} \quad (32.1)$$

From Fig. 31.34 and Eq. (31.84) the magnitude of Eq. (32.1), in the frequency domain, is (noting the z^{-1} in the numerator of Eq. [32.1] is a delay of T_s that adds to the phase of the integrator but doesn't affect the magnitude response)

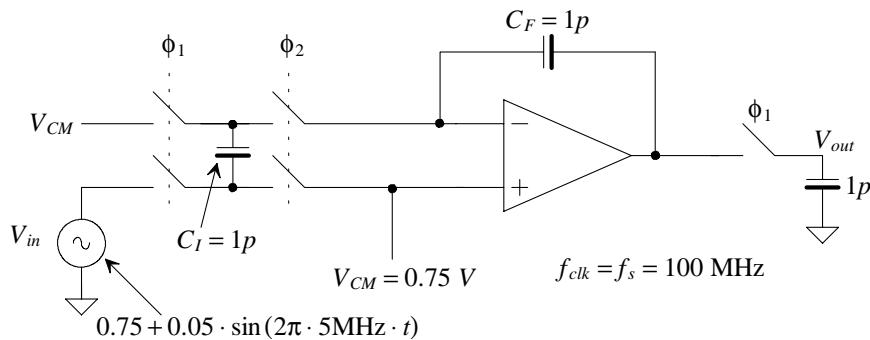


Figure 32.3 Circuit used in Ex. 32.1.

$$|H(f)| = \frac{1}{\sqrt{2\left(1 - \cos 2\pi \frac{5}{100}\right)}} = 3.2 \quad (32.2)$$

resulting in a peak output voltage of 160 mV (peak-to-peak voltage of 320 mV). The simulation results are shown in Fig. 32.4. Notice how the output, at DC, is defined by the initial state of the feedback capacitor, C_F . In this case the input and output of the integrator start at the same voltage. The phase shift can be calculated, using Eq. (31.85) and knowing the zero is not present, as -99° . Finally, the SPICE netlist used to generate this plot is listed below.

* Figure 32.4 CMOS: Mixed-Signal Circuit Design *

```
.tran 1n 500n 0 1n UIC

*WinSPICE command scripts
*#destroy all
*#run
*#plot Vout Vin ylimit 0.6 1.2

*Input power and references
Vtrip Vtrip 0 DC 0.75
VCM VCM 0 DC 0.75

*Input Signal
Vin Vin 0 DC 0 Sin 0.75 50m 5MEG

*Clock Signals
Vphi1 phi1 0 DC 0 Pulse 0 1.5 0 200p 200p 4n 10n
Vphi2 phi2 0 DC 0 Pulse 0 1.5 5n 200p 200p 4n 10n
R2 phi1 0 1MEG
R3 phi2 0 1MEG

*Use a VCVS for the op-amp
Eopamp Voutop 0 VCM Vinm 100MEG

*Setup switched capacitors and load
C1 Vtop Vbot 1p
CF Voutop Vinm 1p
Cload Vout 0 1p

*Setup switches for the integrator
S1 VCM Vtop phi1 VTRIP switmod
S2 Vin Vbot phi1 VTRIP switmod
S3 Vtop Vinm phi2 VTRIP switmod
S4 Vbot VCM phi2 VTRIP switmod
S5 Voutop Vout phi1 VTRIP switmod
.model switmod SW RON=100

.end
```



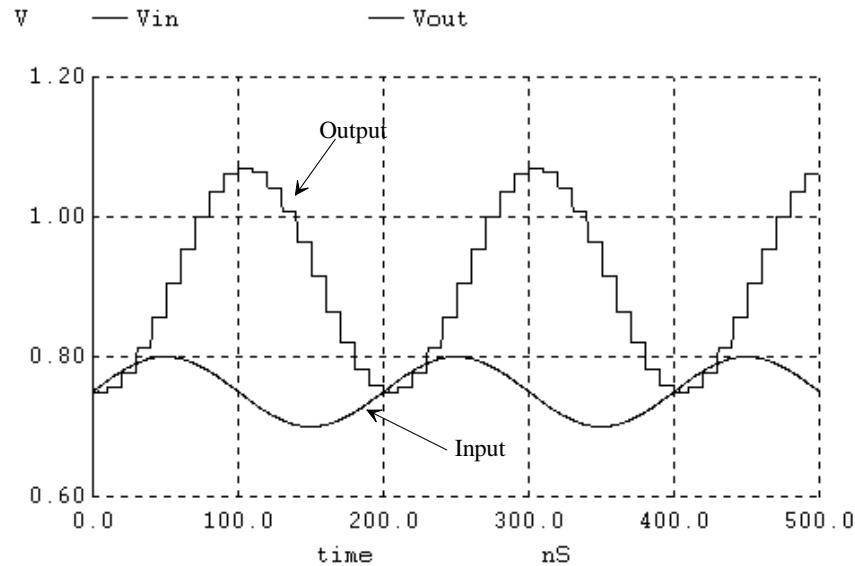


Figure 32.4 Integrator input and output for Ex. 32.1.

SPICE Modeling a 1-Bit ADC (A Comparator)

Modeling a nonclocked comparator is straightforward using switches as seen in Fig. 32.5. When the positive comparator input is greater than the negative input, the output of the comparator is high. When the negative input is greater than the positive input, the output of the comparator is low. In the implementation we might need to connect large, dummy resistors (or small capacitors) to the comparator inputs to keep the nodes from floating. For a clocked comparator we will add, before the continuous comparator S/H. The basic topology of our S/H was shown back in Fig. 30.24.

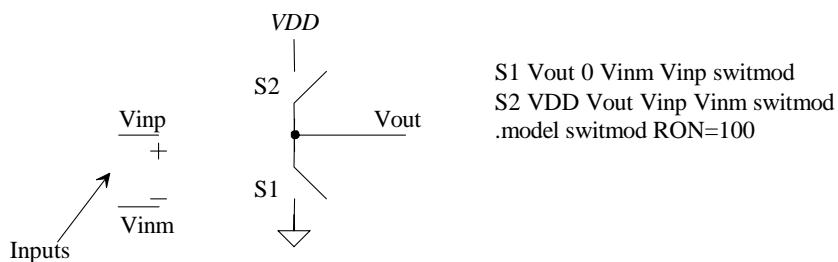


Figure 32.5 Modeling a comparator in SPICE.

32.1.2 First-Order Noise Shaping

The block diagram of a NS feedback modulator is shown in Fig. 32.6. In the end of Ch. 31 we showed that the output of the modulator, $Y(z)$, can be related to the input, $X(z)$, and the ADC's quantization noise, $E(z)$, by

$$Y(z) = \underbrace{\frac{A(z)}{1+A(z)} \cdot X(z)}_{STF(z)} + \underbrace{\frac{1}{1+A(z)} \cdot E(z)}_{NTF(z)} \quad (32.3)$$

where $STF(f)$ is the signal's transfer function and $NTF(f)$ is the noise's transfer function.

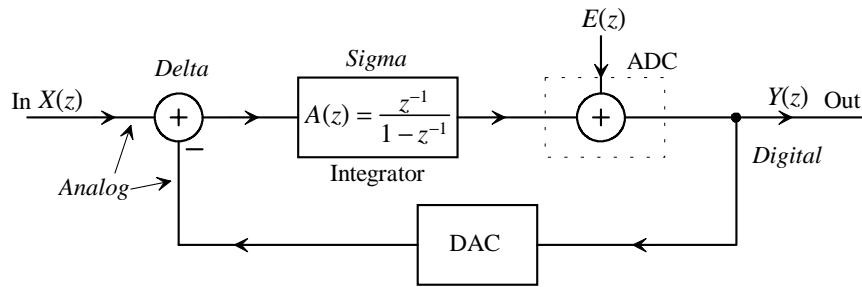


Figure 32.6 Block diagram of a noise-shaping (NS) modulator.

Consider what happens if $A(z)$ is an integrator (implemented using a DAI) as shown in the figure. Equation 32.3 becomes

$$Y(z) = z^{-1}X(z) + (1 - z^{-1})E(z) \quad (32.4)$$

This equation is important! It shows the input signal simply passes through the modulator with a delay while the quantization noise is differentiated (see Fig. 31.51 for the magnitude response of a digital differentiator with a transfer function $1 - z^{-1}$). We can think of the noise differentiation as pushing the quantization noise to higher frequencies. We'll come back to how NS affects the quantization noise spectral density, $V_{Qe}(f)$, in a moment. But first let's attempt to understand what's happening here.

In Fig. 32.6 the summer takes the difference (Delta) between the input signal and the fed back signal. The integrator accumulates or sums (Sigma) this difference and feeds the result back, via the ADC and DAC, to the summer. This forces the output of the modulator to track the average of the input. Sometimes the fed back signal will have a value greater than the input signal, while at other times the fed back signal will be less than the input signal. The *average* signal fed back, however, should ideally be the same as the input signal. Note that this type of NS modulator is often called a *Delta-Sigma* or *Sigma-Delta* modulator. Also, at this point, we should see the need for the averaging filters discussed in the last chapter.

A circuit implementation of a first-order NS modulator is shown in Fig. 32.7. For the moment we use a single-bit ADC and DAC (both implemented using the clocked comparator) for gain linearity reasons (discussed in more detail later). The analog voltage coming out of the integrator is compared to the common-mode voltage (this is our 1-bit ADC) using the comparator. For the 1-bit DAC a logic-0 has an analog voltage of 0 V, while a logic-1 has an analog voltage of VDD ($= 1.5$ V here) so that the comparator's output can be used directly (fed back to the DAI).

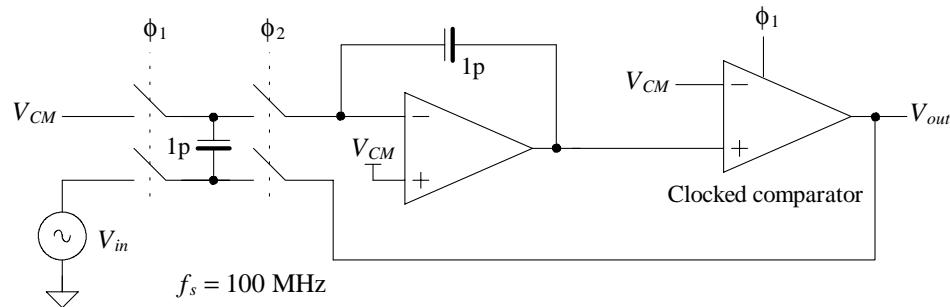


Figure 32.7 Circuit implementation of a first-order NS modulator.

The comparator is clocked on the rising edge of ϕ_1 resulting in a T_s delay (z^{-1}) in series with the feedback signal and a delay of $T_s/2$ ($z^{-1/2}$) in series with the input signal. To understand this statement, remember that the nonoverlapping clock dead time (the time both ϕ_1 and ϕ_2 are low) is short and, practically, the falling edge of ϕ_2 occurs at the same instance as the rising edge of ϕ_1 (and so we could also use $\bar{\phi}_2$ to clock the comparator). This results in a transfer function (see Eq. [31.130] and Table 31.2) to the input of the comparator (which can also be thought of as the ADC output since the fed back signal and modulator output are the same signal) of

$$\text{Desired ADC input/output} = \frac{z^{-1}}{1-z^{-1}}(V_{in} - V_{out}) \quad (32.5)$$

After careful review we should see that the circuit implementation of Fig. 32.7 corresponds to the NS modulator represented by the block diagram shown in Fig. 32.6.

We can use the SPICE models developed in the last section to demonstrate the operation of the NS modulator of Fig. 32.7. Assuming our sampling frequency is 100 MHz, the input is a 500 kHz sinewave centered around V_{CM} ($= 0.75$ V) with a peak amplitude of 0.7 V. The input and output of the modulator are shown in Fig. 32.8. It's important to understand the signals in this figure. When the sinewave is at its peak amplitude, the output of the modulator stays high, a logic one, most of the time. When the sinewave is moving through the common mode voltage, the output bounces back and forth between VDD and ground so that its average value, $VDD/2$, matches the input value.

To construct a (higher resolution) ADC the output of the modulator is connected to a digital averaging filter, as shown in Fig. 32.9. The output of the digital filter is a

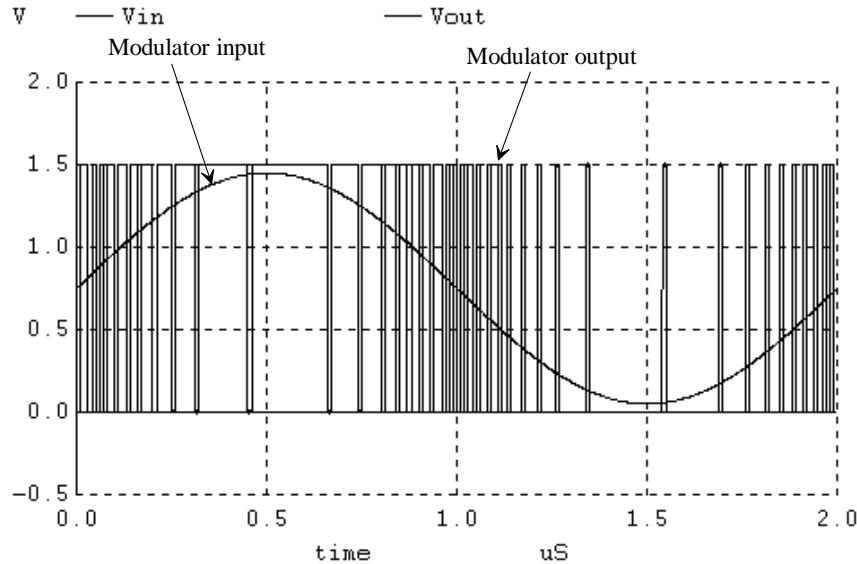


Figure 32.8 Modulator, Fig. 32.7, input and output.

digital word representing the analog input voltage. For a detailed discussion of the requirements placed on the anti-aliasing filter (AAF) and the digital filter (or digital-decimation filter if decimation is used), see Ch. 31.

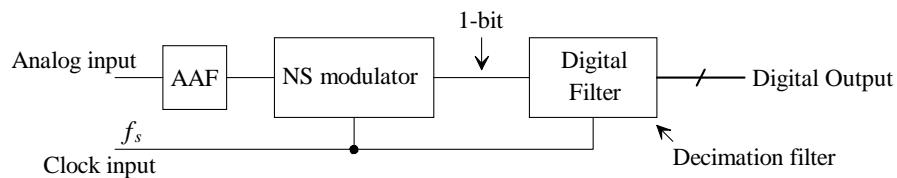


Figure 32.9 ADC using a NS modulator and digital filter.

We might wonder if we can use an analog filter, instead of a digital filter in the topology of Fig. 32.9, to remove the high-frequency quantization noise. The output of the resulting circuit will be analog, so it can't be used as an ADC. While this may not be of practical use at the moment, it does help in understanding how the NS modulator functions. Passing the modulator output of Fig. 32.8 through a simple RC lowpass filter, with a time constant of 100 ns, results in the waveform shown in Fig. 32.10. Increasing the time constant results in a smoother output signal. However, increasing the time constant too much can affect the amplitude of the desired signal. Also, note the phase shift through the modulator and filter.

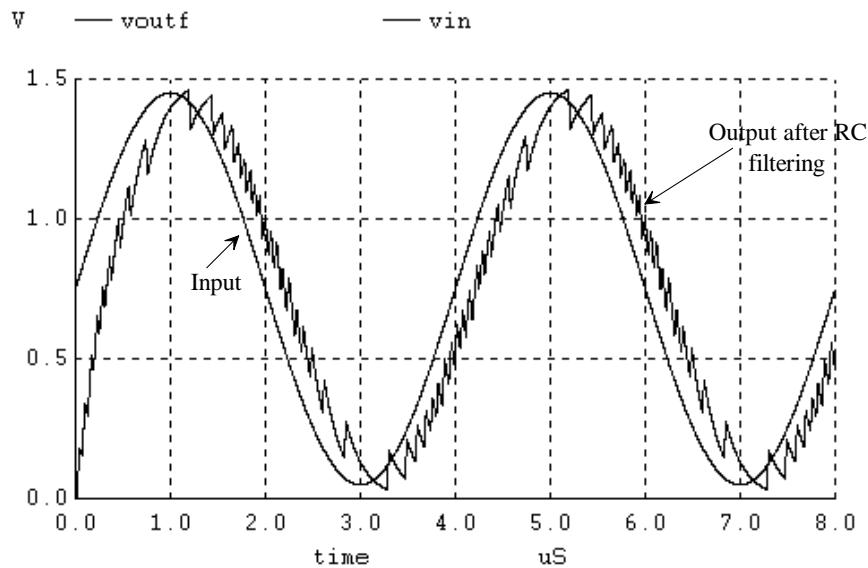


Figure 32.10 Using a simple RC lowpass filter on the output of the NS modulator of Fig. 32.7.

A Digital First-Order NS Demodulator

So far our noise-shaping discussion has centered around analog-to-digital conversion. Figure 32.11 shows a first-order NS demodulator-based topology for digital-to-analog conversion. While, once again, the discussion concerning selection of the digital interpolating and reconstruction filters is given in Ch. 31, we are interested here in the topology of the first-order NS demodulator for use in a DAC.

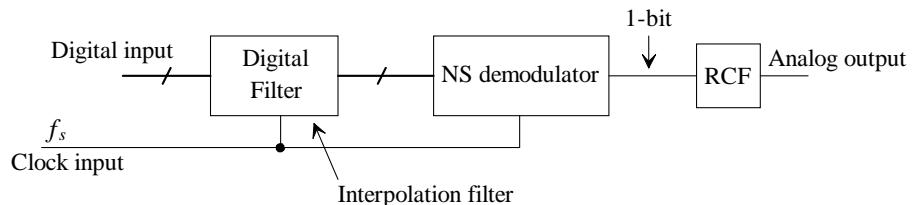


Figure 32.11 DAC using a NS modulator and digital filter.

Figure 32.12a shows a block diagram of a first-order NS demodulator for use in a DAC. Figure 32.12b shows the practical implementation. The only differences between this circuit and the circuit of Fig. 32.6 is that the DAI is replaced with an all-digital integrator (see Fig. 31.49), and the quantizer (comparator) is replaced with a circuit that performs quantization by selecting (using the MSB of the accumulator output word) digital V_{REF+} ($= 011111\dots$ in two's complement, see Fig. 31.37) or V_{REF-} ($= 10000\dots$) which, for our current discussion, are VDD and ground.

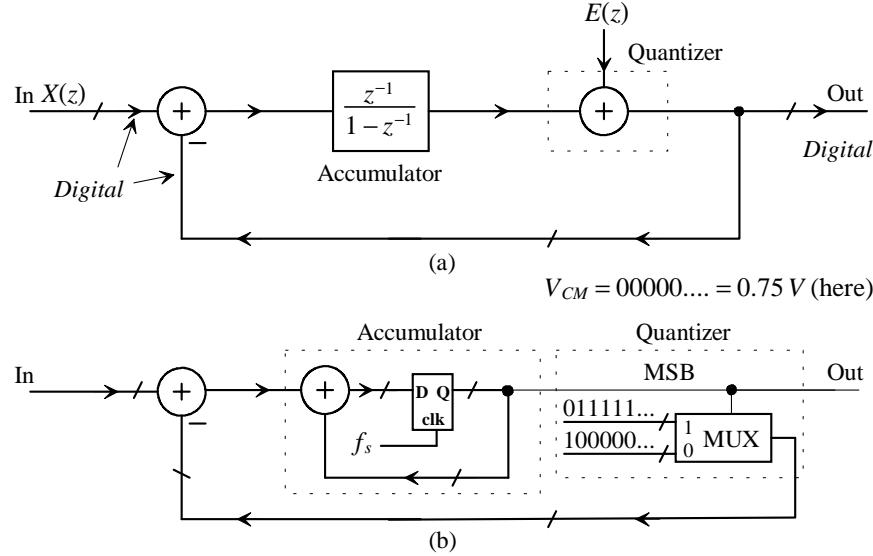


Figure 32.12 Block diagram of (a) a NS demodulator and, (b) a more detailed implementation for use in a DAC.

Modulation Noise in First-Order NS Modulators

Here we present a more detailed discussion of the quantization noise spectrum for a first-order NS modulator. To begin let's write Eq. (32.4) in the time domain,

$$Y[nT_s] = X[(n-1)T_s] + E[nT_s] - E[(n-1)T_s] \quad (32.6)$$

which shows the output is a function of the first difference (order) of the quantization noise $E[nT_s] - E[(n-1)T_s]$. Intuitively note that the smaller we make T_s (the faster we sample since $T_s = 1/f_s$), the closer our digital output $Y[nT_s]$ approaches the analog input $X[nT_s]$.

Next, using Eq. (32.4), let's write the product of the noise transfer function and $E[z]$ (the modulation noise) of the first-order NS modulator in the frequency domain as

$$NTF(z)E(z) = (1-z^{-1})E(z) \rightarrow NTF(f)V_{Qe}(f) = \left(1 - e^{-j2\pi\frac{f}{f_s}}\right) \cdot \frac{V_{LSB}}{\sqrt{12f_s}} \quad (32.7)$$

where we have used, see Fig. 30.57,

$$E(f) = V_{Qe}(f) = \frac{V_{LSB}}{\sqrt{12f_s}} \left(\text{units, V}/\sqrt{\text{Hz}} \right) \text{ for } 0 \leq f \leq f_s/2 \quad (32.8)$$

and

$$V_{LSB} = \frac{V_{REF+} - V_{REF-}}{2^N} \quad (32.9)$$

where N is the number of bits used in the low-resolution ADC/DAC in the modulator. Using a single-bit ADC/DAC in a NS modulator, $N = 1$, results in $V_{LSB} = 1.5$ V (see Prob. 30.14 for a discussion of when Eq. [32.9] isn't valid). This again shows that we are not reducing the quantization noise, but are rather pushing it to higher frequencies so that it can be filtered out. Using Eq. (31.107), we can write the PSD of the NTF (the PSD of the first-order modulator's modulation noise) as

$$|NTF(f)|^2 \cdot |V_{Qe}(f)|^2 = \frac{V_{LSB}^2}{12f_s} \cdot 2\left(1 - \cos 2\pi \frac{f}{f_s}\right) \quad (\text{units, } V^2/\text{Hz}) \quad (32.10)$$

Figure 32.13 shows the PSD of the first-order NS modulation noise for $V_{LSB} = 1.5$ V and $f_s = 100$ MHz. Note how now we are discussing modulation noise instead of quantization noise. *The modulation noise is the quantization noise after being differentiated by the NS modulator.* The modulation noise is the unwanted signal added to the input signal. After reviewing Fig. 32.13 we see that the magnitude of the modulation noise is significant. However, after passing this signal through a lowpass filter, we can remove the higher frequency noise resulting in a lower value of data converter RMS quantization noise, $V_{Qe,RMS}$. Figure 32.14 shows the PSD of the noise if we limit our view to 1 MHz. The point here is that by restricting the bandwidth of the modulation noise we can, theoretically, drive the RMS quantization noise in our signal to zero. Of course, by lowering the bandwidth of the digital filter on the output of the modulator we also limit the possible bandwidth, B , of the input signal. Notice that we have violated Bennett's criteria by utilizing a quantizer with an LSB that is comparable to the input signal. Now, however, we are using feedback that adds or subtracts a signal from the input and ultimately affects the quantizer input.

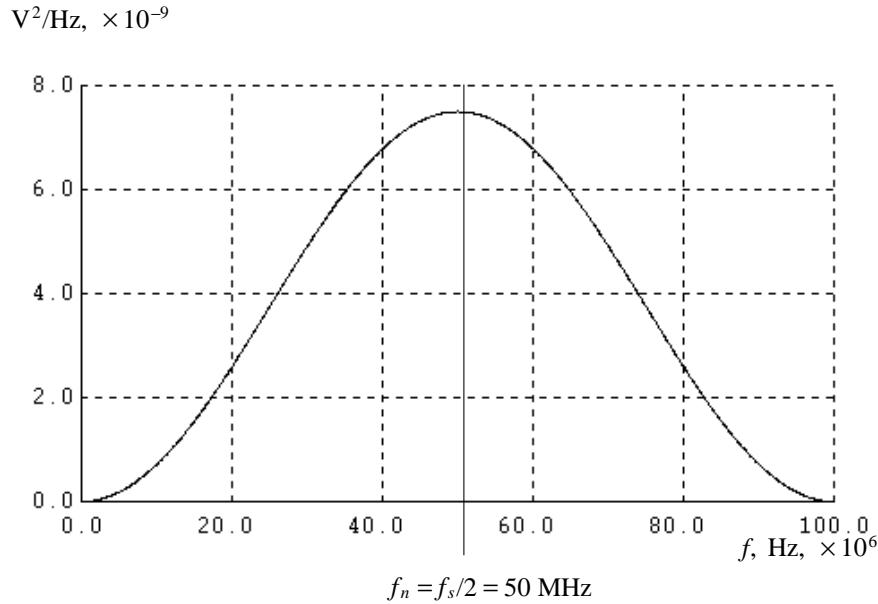


Figure 32.13 Modulation noise for a first-order NS modulator.

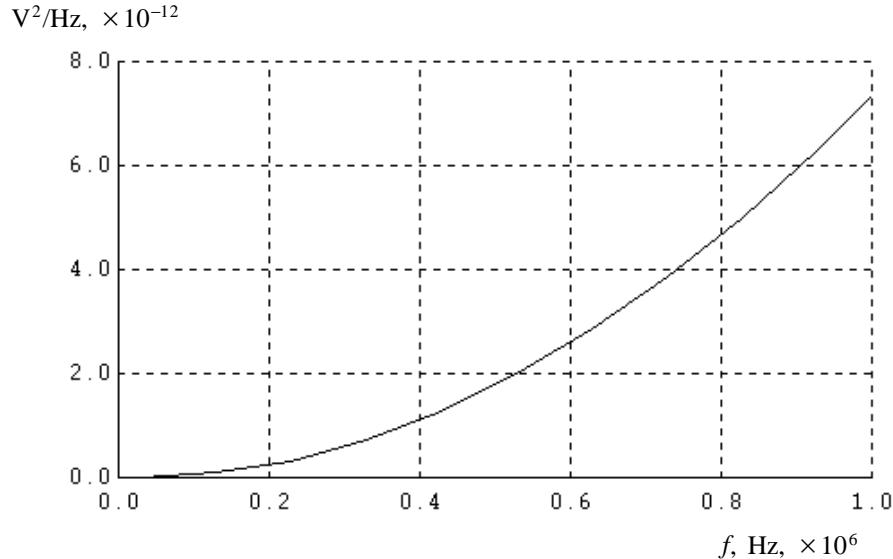


Figure 32.14 A limited view of the modulation noise of Fig. 32.13.

Example 32.2

Using SPICE, show the modulation noise spectrum associated with the NS modulator of Fig. 32.7. Compare the simulation results to the theoretical results shown in Fig. 32.13.

Following the procedure given back in Sec. 30.3.1, Fig. 30.46, to determine a data converter's quantization noise spectrum, we apply a slowly moving voltage ramp to the input of the modulator. Then we look at the difference between the input and output of the modulator (the modulation noise). The simulation results are shown in Fig. 32.15a and 32.15b. We used the following WinSPICE commands to generate these plots (added directly into the netlist)

```
*#plot Vout Vin
*#let Vqev=Vout-Vin
*#linearize Vqev
*#spec 0 100MEG 200k Vqev
*#let Vqedb=db(Vqev)
*#plot Vqedb
```

The first command is used to generate Fig. 32.15a. The second command is used to generate the difference between the modulator's input and output (the modulation noise). Notice how, in Fig. 32.15a, the output of the modulator stays low most of the time, when the input to the modulator is close to ground, while the output stays high most of the time when the input is close to VDD . The third and

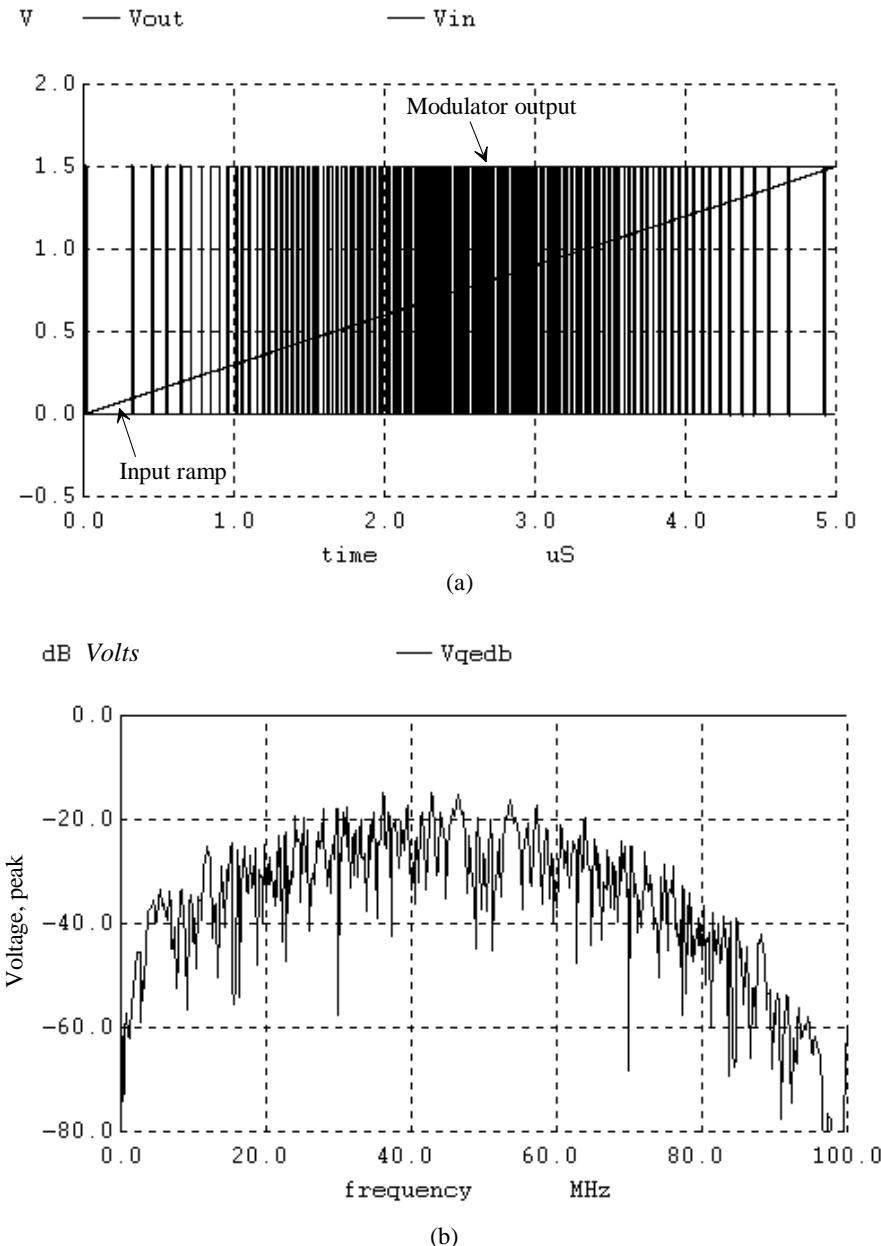


Figure 32.15 (a) Input and output of the NS modulator of Fig. 32.7, and (b) modulation noise output spectrum.

fourth commands in the above list generate the spectrum of the modulation noise (the spectrally shaped quantization noise). The last commands are used to plot, Fig. 32.15b, the spectrum of the modulation noise (units of Volts). It may be helpful, at this point, to review Fig. 30.48 and the associated discussion. To change this plot (Fig. 31.15b) into a power spectral density (units of V^2/Hz) we can square the magnitude of the modulation noise and then divide the result by the resolution of the Fourier transform ($f_{res} = 200$ kHz in Fig. 32.15). The list of commands used to generate a *power spectral density* from Fig. 32.15b would be

```
*#let mrms=mag(Vqev)/1.414
*#let Vqepsd=10*log10(mrms*mrms/200k)
*#plot Vqepsd
```

Using this sequence of commands results in a spectrum with amplitude values that are similar to the values given in Fig. 32.13 (and have the same units). However, the shape would remain essentially unchanged from Fig. 32.15b.

Notice that the shape of the modulation noise shown in Fig. 32.15b (in decibels) matches fairly well with the spectrum shown in Fig. 32.13. This is the case even though the quantization noise spectral density, $E(f)$, is not flat (is not white), as was assumed in Eq. (32.7). The important thing to notice in Fig. 31.15b is that the modulation noise spectrum decreases with decreasing frequency (at 20 MHz and below), as was predicted using Eq. (32.10) and shown in Fig. 32.13. ■

RMS Quantization Noise in a First-Order Modulator

If we limit the range of frequencies we look at to calculate the quantization noise to values below f_s , then we can rewrite Eq. (32.10) as

$$|NTF(f)| \cdot |V_{Qe}(f)| = \frac{V_{LSB}}{\sqrt{12f_s}} \cdot 2 \sin \pi \frac{f}{f_s} \quad (\text{units, } V/\sqrt{\text{Hz}}) \quad (32.11)$$

The RMS quantization noise present in a bandwidth, B , can be calculated, see Eq. (30.44), using

$$V_{Qe,RMS}^2 = 2 \int_0^B |NTF(f)|^2 |V_{Qe}(f)|^2 \cdot df = 2 \cdot \frac{V_{LSB}^2}{12f_s} \cdot 4 \cdot \int_0^B \sin^2 \pi \frac{f}{f_s} \cdot df \quad (32.12)$$

Remembering that the maximum bandwidth of our input signal is related to the sampling frequency, f_s , and the oversampling ratio, K , by

$$B = \frac{f_s}{2K} \quad (32.13)$$

and, for small values of x ,

$$\sin x \approx x \quad (32.14)$$

then

$$V_{Qe,RMS} \approx \frac{V_{LSB}}{\sqrt{12}} \cdot \frac{\pi}{\sqrt{3}} \cdot \frac{1}{K^{3/2}} \quad (32.15)$$

This equation should be compared to Eq. (31.51). Further we can describe the ideal data converter SNR using first-order NS, see Eqs. (31.1) - (31.4) as

$$\text{SNR}_{ideal} = 20 \cdot \log \frac{V_p/\sqrt{2}}{V_{Qe,RMS}} = 6.02N + 1.76 - 20 \log \frac{\pi}{\sqrt{3}} + 20 \log K^{3/2} \text{ (in dB)} \quad (32.16)$$

or

$$\text{SNR}_{ideal} = 6.02N + 1.76 - 5.17 + 30 \log K \text{ (in dB)} \quad (32.17)$$

This equation should be compared to Eq. (31.52) where we saw every doubling in the oversampling ratio, K , results in a 0.5-bit increase in resolution (called simple oversampling). Here we see that *every doubling in the oversampling ratio results in 1.5 bits increase in the resolution* (or a 9 dB increase in SNR_{ideal}). A first-order NS modulator's performance is compared to simple oversampling in Fig. 32.16.

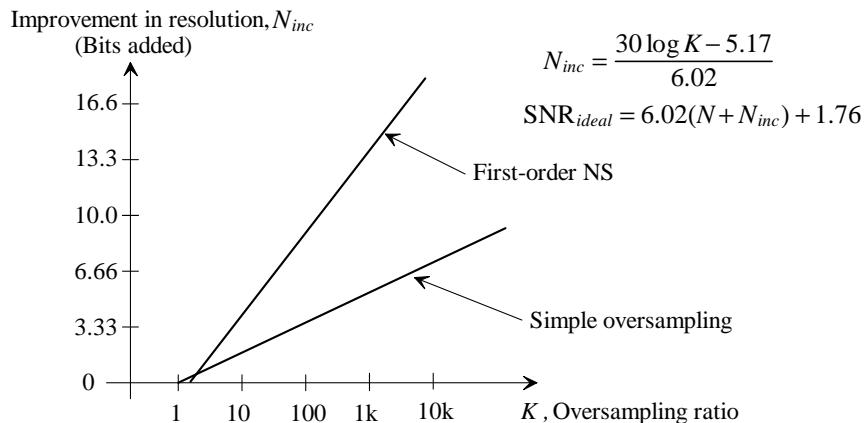


Figure 32.16 Comparing simple oversampling to first-order noise-shaping.

Examples 32.3

Determine the ideal signal-to-noise ratio and the maximum signal bandwidth allowed, B , for the first-order NS modulator of Fig. 32.7 if 16 of its output samples are averaged ($K = 16$).

Because the sampling frequency, f_s , is 100 MHz, we can use Eq. (32.13) to determine the maximum input signal bandwidth, B , is 3.125 MHz. Using Eq. (32.17) we can solve for the SNR_{ideal} as (knowing that the NS modulator of Fig. 32.7 uses a 1-bit quantizer) 38.73 dB. This corresponds to an equivalent data converter (ADC) resolution, using Eq. (31.4), of 6.14 bits (number of bits added is

5.14). Note that the ADC is made with the components, modulator, and digital filter, shown in Fig. 32.9. ■

Decimating and Filtering the Output of a NS Modulator

It's important to note that Eq. (32.17) was derived assuming the output of the modulator was passed through a perfect lowpass filter with a bandwidth of B . Passing the output through a sinc averaging filter, see Fig. 31.41, will result in a poorer SNR because the higher frequency noise components will not be entirely filtered out. In this section we want to answer two questions: (1) what order, L (see Eq. [31.104]), of sinc averaging filter should be used in the digital filter on the output of the NS modulator, and (2) assuming we use only this filter (no half-band filter or additional filtering), how will the ideal SNR of the first-order NS modulator be affected.

We begin to answer to the first question by writing the increase in the number of bits, N_{inc} , as

$$N_{inc} = \frac{30 \log K - 5.17}{6.02} \quad (32.18)$$

If our NS modulator uses a 1-bit ADC, then the final, after the digital filter, resolution of the resulting data converter is $N_{inc} + 1$ bits. (An NS modulator using a 5-bit ADC [often called a multibit NS modulator] would ideally have an output resolution of $N_{inc} + 5$ bits.) Further, we saw in Ex. 31.22 and Fig. 31.55 that the word size increased by $\log_2 K$ bits in each stage so that we can require

$$L \cdot \log_2 K \geq \frac{30 \log K - 5.17}{6.02} \quad (32.19)$$

Solving this equation results in L being greater than or equal to 2. In general, we can write

$$L = 1 + M \quad (32.20)$$

where M is the order of the modulator. For a first-order modulator we use two stages in the averaging filter, or,

$$H(z) = \left[\frac{1}{K} \frac{1-z^{-K}}{1-z^{-1}} \right]^2 \quad (32.21)$$

In the next section we discuss second-order NS modulators ($M = 2$). For these modulators we use a sinc averaging filter with $L = 3$.

Example 32.4

Sketch the implementation of the digital decimation filter for the modulator described in Ex. 32.3. Assume the final output clocking frequency is 100 MHz/16 or 6.25 MHz. Do not be concerned with aliasing (use only the averaging filter).

The transfer function of the digital filter is (see Eq. [31.93])

$$H(z) = \left[\frac{1 - z^{-16}}{1 - z^{-1}} \right]^2 = |H(f)| = \left[\frac{1}{16} \cdot \frac{\sin(16 \cdot \pi \frac{f}{f_s})}{\sin(\pi \frac{f}{f_s})} \right]^2$$

The block diagram of the filter is shown in Fig. 32.17. The increase in resolution through each accumulator (integrator) stage is $\log_2 16 = 4$ bits. The resolution calculated in Ex. 32.3 was 6.14 bits, which we round up to 7-bits. Because the output of the digital filter is 9-bits, we drop the lower two bits (divide by 4) to get our final 7-bit resolution. ■

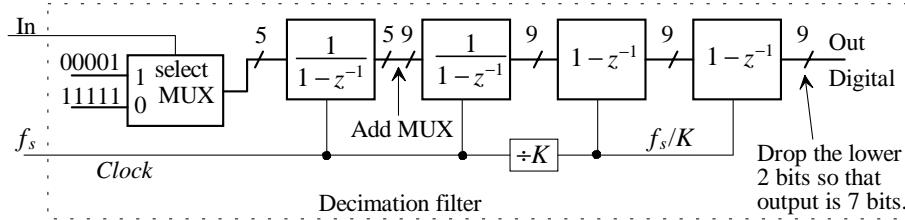
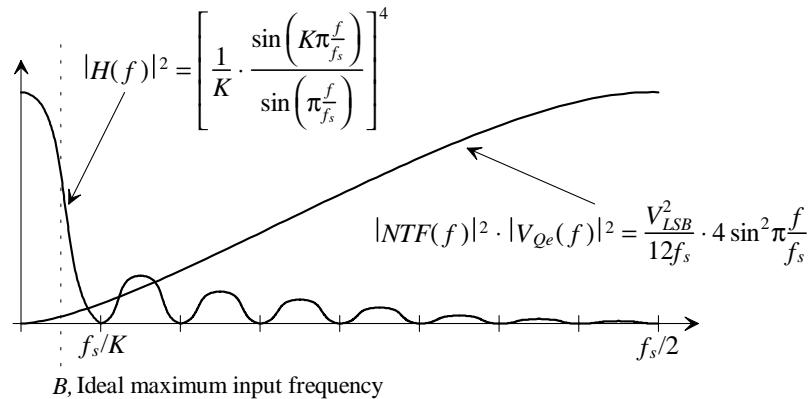


Figure 32.17 Sinc filter used for decimating the output of the NS modulator of Fig. 32.7.

Next let's answer how filtering with a sinc filter affects the SNR of the data converter. Remember the $\text{SNR}_{\text{ideal}}$ was calculated in Eq. (32.17) assuming the modulation noise was strictly bandlimited to B . Figure 32.18 shows the PSD of the $NTF^2(f) \cdot |V_{Qe}(f)|^2$ (the modulation noise) of the first order NS modulator. Also shown in this figure is the shape of the averaging filter's magnitude response squared (see Eqs. [32.22], [31.90] and [31.93]). Here we are showing the shape of a filter with $L = 2$ (set by Eq. [32.20] for a first-order modulator) and $K = 16$. We limit our range to $f_s/2$.



B , Ideal maximum input frequency

Figure 32.18 Showing modulation noise and filter response.

We can calculate the RMS quantization noise resulting from a cascade of a first-order modulator and an averaging filter using

$$V_{Qe,RMS}^2 = 2 \int_0^{f_s/2} |NTF(f)|^2 \cdot |V_{Qe}(f)|^2 \cdot |H(f)|^2 \cdot df \quad (32.22)$$

$$V_{Qe,RMS}^2 = \frac{V_{LSB}^2}{12f_s} \cdot \frac{8}{K^4} \cdot \int_0^{f_s/2} \frac{\sin^4\left(K\pi\frac{f}{f_s}\right)}{\sin^2\left(\pi\frac{f}{f_s}\right)} \cdot df \quad (32.23)$$

If we let $\theta = \pi\frac{f}{f_s}$, then this equation can be written as

$$V_{Qe,RMS}^2 = \frac{V_{LSB}^2}{12f_s} \cdot \frac{8}{K^4} \cdot \frac{f_s}{\pi} \cdot \overbrace{\int_0^{\frac{\pi}{2}} \frac{\sin^4(K\theta)}{\sin^2\theta} d\theta}^{= K \cdot \frac{\pi}{4}} \quad (32.24)$$

and finally,

$$V_{Qe,RMS} = \frac{V_{LSB}}{\sqrt{12}} \cdot \frac{\sqrt{2}}{K^{3/2}} \quad (32.25)$$

This equation should be compared to Eq. (32.15), which was derived assuming the digital filter was ideal with a bandwidth of B . The SNR resulting from using a first-order ($M = 1$) NS modulator and a second-order ($L = 2$) sinc averaging filter is

$$\text{SNR}_{sinc} = 6.02N + 1.76 - 3.01 + 30 \log K \quad (\text{in dB}) \quad (32.26)$$

Comparing this to SNR_{ideal} given in Eq. (32.17), we see that using a sinc filter for averaging results in only a 2.16 dB difference (increase) in SNR over the ideal filter. If we remember that using a sinc filter results in a droop in the desired signal, see Fig. 31.46, the SNR will be lower than what is predicted by Eq. (32.26). (Note that an analysis of higher order modulators using sinc averaging filters would show that as long as Eq. [32.20] is valid the deviation from SNR_{ideal} is negligible.)

We have not talked about the effects of sample rate reduction (decimation) on the SNR. The modulation noise aliased into the base spectrum is a concern when decimating the output of the modulator. The major difference between filtering the outputs of the data converters in the last chapter and filtering the output of an NS modulator is the spectral characteristics of the noise. In the last two chapters we assumed the quantization noise spectral density was white (see Fig. 30.57). As we've seen in this chapter the modulation noise increases with increasing frequency (see Fig. 32.18). This can mean that the amount of modulation noise aliased into the base, or desired, spectrum can be more of a concern. The major concern when decimating, in most situations, is the input signals that reside at frequencies $> B$ and the resulting aliasing degradation of the SNR (see, for example, Fig. 31.59).

Implementing the Sinc Averaging Filter Revisited

We chose to implement the sinc averaging filter for the first-order NS modulator of Fig. 32.7 with the topology shown in Fig. 32.17. We might wonder what the implementation of the averaging filter would look like if we had used a cascade of accumulate-and-dumps similar to what is shown in Fig. 31.44 (assuming the larger reduction in output clocking frequency [f_s/K^2 instead of f_s/K] and the possible aliasing isn't a concern [as discussed in Sec. 31.2.2]). For a first-order NS modulator ($M = 1$) the desired transfer function of the averaging filter ($L = 2$) was given by Eq. (32.21).

A cascade of two accumulate-and-dumps is shown in Fig. 32.19. Since $z \equiv e^{j2\pi f_s t}$ the overall transfer function of the averaging filter is

$$H(z) = \frac{1 - z^{-K^2}}{1 - z^{-1}} \quad (32.27)$$

This equation shows that adding a second accumulate-and-dump results in averaging K^2 samples while not providing a reduction in the first sidelobe's amplitude. Because the bandwidth of the filter is reduced using this topology, we get a corresponding reduction in quantization noise on the output of the filter (and corresponding increase in the attenuation when looking at a fixed frequency in the stop band). While we could modify the cascade of accumulate-and-dumps to operate properly, with a final output clocking frequency of f_s/K , the simplicity and ease of designing the averaging filter using the topology of Figs. 32.17, 31.54, or 31.56 makes them the topology of choice, in most situations, for a NS modulator averaging filter.

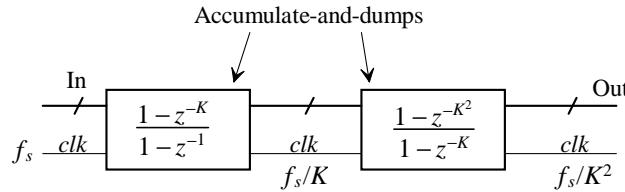


Figure 32.19 Problems with cascading accumulate-and-dump circuits.

The time interval we average the output of the modulator over is related to the order of sinc averaging filter as shown in the following two examples. A time interval longer than KT_s is used when $L \geq 2$.

Example 32.5

Determine the time domain impulse response of a first-order averaging filter ($L = 1$) with $K = 8$. Assume decimation is not employed in the filter.

The transfer function of the filter is given, after reviewing Eqs. (31.89) and (31.90), by

$$H(z) = \frac{1-z^{-8}}{1-z^{-1}} = 1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + z^{-5} + z^{-6} + z^{-7}$$

The time domain relationship between the input and the output is then

$$y[nT_s] = x[nT_s] + x[(n-1)T_s] + x[(n-2)T_s] + \dots + x[(n-7)T_s]$$

The time-domain impulse response of the first-order averaging filter is shown in Fig. 32.20. Note the rectangular shape. ■

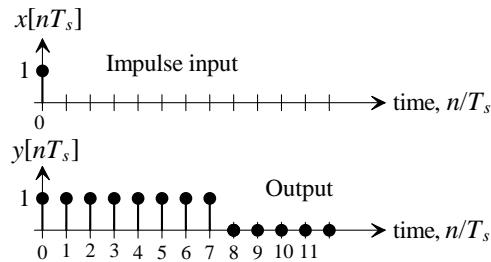


Figure 32.20 Impulse response of an $L = 1$ averaging filter.

Example 32.6

Repeat Ex. 32.5 if a second-order averaging filter is used.

The transfer function of the filter is

$$H(z) = \left[\frac{1-z^{-8}}{1-z^{-1}} \right]^2 = 1 + 2z^{-1} + 3z^{-2} + 4z^{-3} + 5z^{-4} + \dots + 3z^{-12} + 2z^{-13} + z^{-14}$$

The time domain relationship is

$$y[nT_s] = x[nT_s] + 2x[(n-1)T_s] + 3x[(n-2)T_s] + \dots + 2x[(n-13)T_s] + x[(n-14)T_s]$$

The impulse response of the second-order averaging filter is shown in Fig. 32.21. Note the triangular shape of the curve and how the impulse response of the second-order filter lasts twice as long as the first-order's response. ■

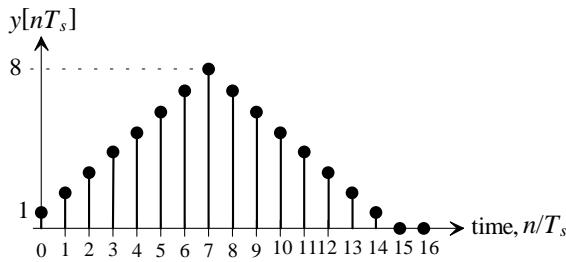


Figure 32.21 Impulse response of an $L = 2$ averaging filter.

Analog Sinc Averaging Filters Using SPICE

It can be very useful to simulate the operation of a sinc averaging filter of the form

$$H(z) = \left[\frac{1}{K} \frac{1-z^{-K}}{1-z^{-1}} \right]^L \quad (32.28)$$

Implementing this filter using digital circuits can result in long simulation times. Also, the output of the filter will be digital requiring the use of a DAC to reconstruct the analog input voltage. While in an actual chip design we may (will) want to simulate the actual circuit fabricated (the digital implementation of the filter), it is still nice to have a computationally efficient ideal filter for fast simulations during the development of the modulator.

After reviewing Fig. 32.10, and the associated discussion, we see that we can pass the output of the modulator directly through an analog filter to reconstruct the analog input voltage (with the unwanted quantization noise). Using an analog filter eliminates the need for an additional DAC and makes observing the resulting modulator/filter output simulation spectrum straightforward. The question now becomes, "How do we make a simple sinc-shaped analog filter that models the digital implementation?"

We saw in Fig. 31.53 that a digital comb filter has a transfer function of $1 - z^{-K}$. We also saw in Ch. 30 that we can implement an analog comb filter using a transmission line, Fig. 30.11. Here we attempt to use the simple circuit shown in Fig. 32.22a to implement a comb filter in SPICE. The integrator, Fig. 31.33, can be implemented using a similar topology and is shown in Fig. 32.22b. To get a filter with a continuous-time

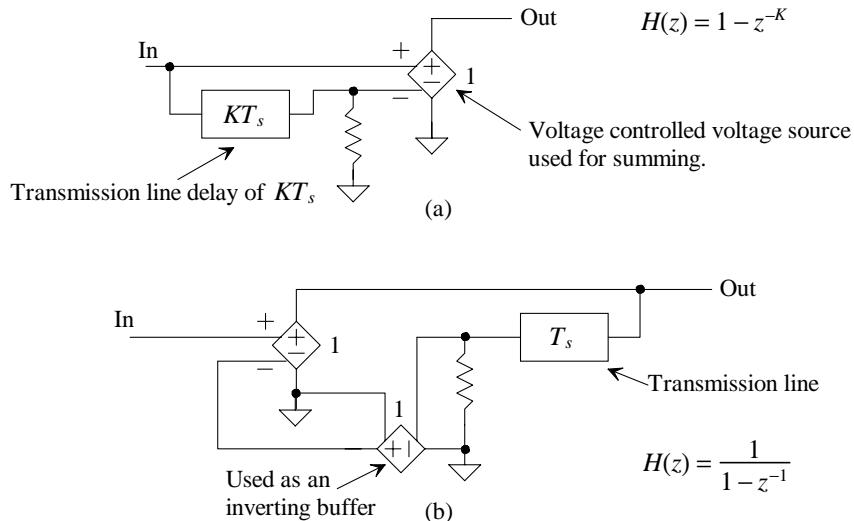


Figure 32.22 (a) Implementation of a comb filter in SPICE, and (b) implementation of an integrator.

transfer function equivalent to Eq. (32.28) (see Eqs. [31.100] and [31.101]), we can cascade L of these sections. The resistors used on the output of the transmission lines, in Fig. 32.22 are used to terminate the transmission lines. The following example demonstrates the implementation of a sinc averaging filter in SPICE.

Example 32.7

Generate a SPICE model for the digital filter used with the first-order modulator of Fig. 32.7. Assume, as in Ex. 32.4, that $K = 16$. Specify the droop in the output when an input sinewave has a frequency of B .

In this modulator the clocking frequency is 100 MHz ($T_s = 10$ ns). The delay in the comb filters is 160 ns, while the delay in the integrators is 10 ns. The SPICE netlist is shown below. Note how the input to the filter is isolated using a buffer, and the integrators are isolated by the comb filters. The output is scaled at the end of the netlist to $1/K^2$ to normalize the filter's gain (see Fig. 31.46).

The simulation results are shown in Fig. 32.23. The input to the filter is a 1-V sinewave that is swept from DC to 50 MHz ($= f_s/2$). The droop at B ($= 100$ MHz/ $(2 \cdot 16) = 3.125$ MHz) is 7.8 dB (which matches what was predicted in Fig. 31.46).

* Figure 32.23 CMOS: Mixed-Signal Circuit Design *

```
.AC LIN 1000 1k 50MEG

*WinSPICE command scripts
*#destroy all
*#run
*#set units=degrees
**#plot db(vo2) ylimit 30 0
*#plot db(vout) ylimit 0 -60

Vin      Vin      0      DC      0      AC      1
*Input buffer
Ebuf1   Vobuf   0      Vin      0      1
*Comb filter 1
EC1      Vo1      0      Vobuf   Vf1      1
TC1      Vobuf   0      Vf1      0      ZO=50 TD=160n
RC1      Vf1      0      50
*Integrator filter 1
EI1      Vo2      0      Vo1     Vb1      1
TI1      Vo2      0      Vf2      0      ZO=50 TD=10n
RI1      Vf2      0      50
EB1      Vb1      0      0      Vf2      1
*Comb filter 2
EC2      Vo3      0      Vo2     Vf3      1
TC2      Vo2      0      Vf3      0      ZO=50 TD=160n
RC2      Vf3      0      50
```

```

*Integrator filter 2
EI2    Vo4    0      Vo3    Vb2    1
TI2    Vo4    0      Vf4    0      ZO=50 TD=10n
R12    Vf4    0      50
EB2    Vb2    0      0      Vf4    1

*Scale the output by 1/K^2 (1/256=0.00390625)
Ebuf2  Vout   0      Vo4    0      .00390625

.end
■

```

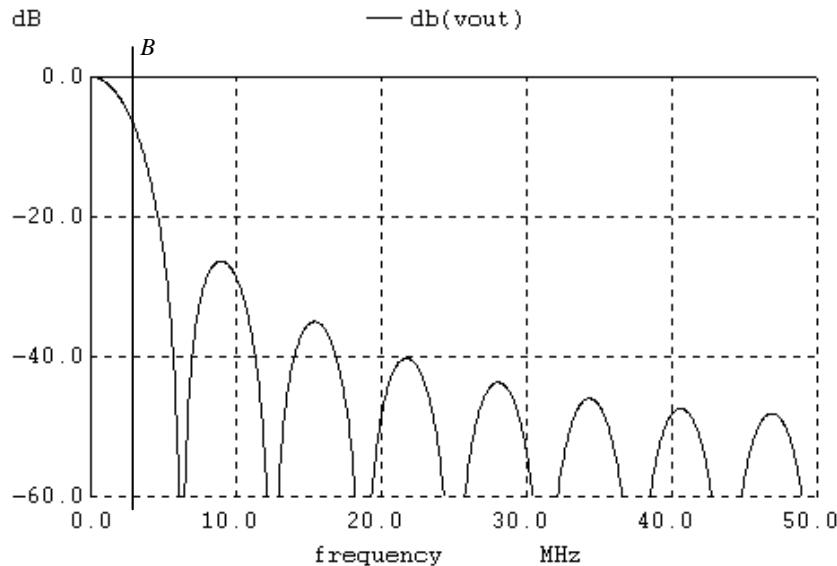


Figure 32.23 Frequency response of the $L = 2$ sinc averaging filter.

Using our SPICE Sinc Filter Model

The analog sinc averaging filter model we've just developed is difficult to use in a practical simulation for several reasons. To begin with, the actual shape of the pulses (their rise and falltimes together with the over- and undershoot in the pulse shape) coming out of the modulator is now important where, in a digital implementation, it is not. The frequency content of the analog pulse stream differs from the spectral content of the digital data. This leads to analog sinc filter output waveforms with distortion that wouldn't be found in the output of the digital sinc filter. For a digital implementation a modulator output rise time of 1 ns is no different from a risetime of 100 ps. In the analog implementation, however, the risetime does matter (consider what happens, in Fig. 32.10, if the risetime of the digital data coming out of the modulator is slowed down). Also, the SPICE implementation of the transmission lines isn't tolerant to fast pulse edge transitions. This leads to difficulty with convergence and, possibly, long simulation times.

Analog Implementation of the First-Order NS Modulator

A continuous-time implementation of a first-order NS modulator is shown in Fig. 32.24. If we write the sum of the currents through the resistors (and thus through the feedback capacitor) as

$$i_F = \frac{V_{CM} - V_{in}}{R} + \frac{V_{CM} + V_{out}}{R} \quad (32.29)$$

then the output of the integrator can be written, assuming the input and output are referenced to V_{CM} , as

$$V_{OI} = \frac{i_F}{j\omega C} = \frac{-1}{j\omega RC}(V_{in} - V_{out}) \quad (32.30)$$

where $\omega = 2\pi f$ and noting how we switched the inverting and noninverting inputs of the comparator to compensate for the inverting gain of the integrator. Knowing, from Eq. (31.82), that

$$\frac{z^{-1}}{1-z^{-1}} = \frac{1}{(-1 + \cos 2\pi \frac{f}{f_s}) + j \sin 2\pi \frac{f}{f_s}} \quad (32.31)$$

or, for $f \ll f_s$ (which must be valid for any oversampling converter) where $\cos 2\pi \frac{f}{f_s} \approx 1$ and $\sin 2\pi \frac{f}{f_s} \approx 2\pi \frac{f}{f_s}$, then

$$\frac{z^{-1}}{1-z^{-1}} \approx \frac{1}{j2\pi \frac{f}{f_s}} \quad (32.32)$$

Finally, we see that the topology of Fig. 32.24 behaves like a modulator with a block diagram of Fig. 32.6 when

$$f_s = \frac{1}{RC} \quad (32.33)$$

If this equation doesn't hold, the topology of Fig. 32.24 may still function correctly as a NS modulator, except that the above analysis would include an integrator gain. (Which,

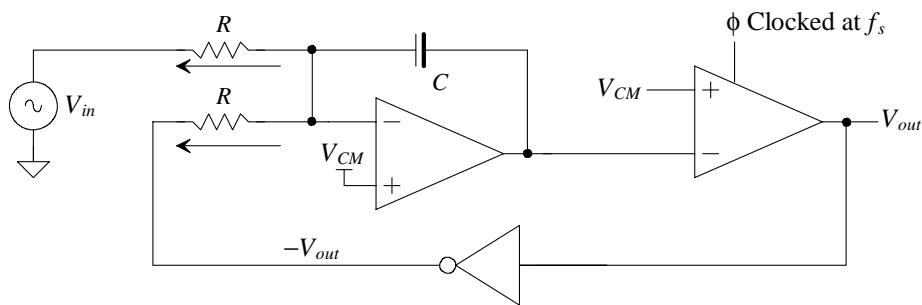


Figure 32.24 Analog circuit implementation of a first-order NS modulator.

combined with the high gain of the comparator, may still result in an overall forward path gain of one. We will discuss component gains in the modulator later.)

Analog integrator-based implementations of modulators can be simpler and easier to breadboard and test on the bench, lower power, and less susceptible to clocking noise (capacitive feedthrough and charge injection). The two main drawbacks are the difficulty in setting the integrator gain to a precise value (in integrated versions) and the integrator's susceptibility to the fed back pulse shape (a problem also encountered using the analog sinc filters of the last section).

Using a DAI, the gain of the integrator is set by a ratio of capacitors, see Fig. 31.79 (and Ch. 27). Variations in the absolute oxide capacitance for a given process run don't affect the integrator's gain. Using the analog integrator in a purely monolithic form, however, can result in RC time constant variations of 50% or more.

Figure 32.25 shows how the shape of the pulse affects the output of the analog integrator. In part (a) we see the ideal pulse shape and the ideal area under the pulse (the shaded area). In part (b) we see how the finite rise time and fall time can affect the actual area under the curve and thus the output of the integrator. To minimize these unwanted effects we can use wider pulses as shown in parts (c) and (d), which means we run the modulator at a slower clocking frequency. Increasing the width of the pulses minimizes the percentage of the area affected by the transition times. Note that the feedback signal directly subtracts from the input signal so that any noise or unwanted variation in the feed back signal, such as an amplitude variation, can be considered as adding noise to the input (and thus degrading the modulator's SNR). *This is important!* We will discuss the fed back signal, and how to isolate/implement the actual voltage fed back to the integrator, again in the next section.

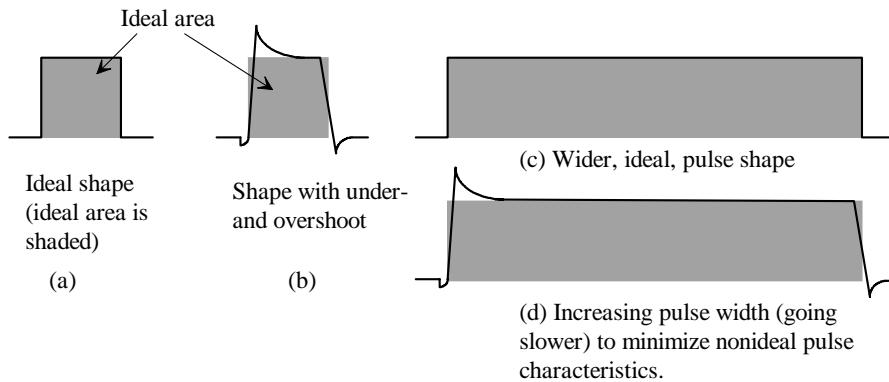


Figure 32.25 Comparator output pulse shapes, input to the integrator.

The DAI is less susceptible to the pulse shape fed back from the comparator. Reviewing Fig. 32.7 we see that as long as the output of the comparator can charge or discharge the switched capacitor to within the final resolution of the converter, before the ϕ_2 switches turn off, the circuit functions as expected. Note that the comparator, having a

digital output, doesn't limit the rate the switched capacitor is charged but rather the limiting factor is the op-amp (since the top plate is tied to the op-amp's inverting input and charged from the op-amp's output through the feedback capacitor).

The Feedback DAC

Up until this point we have been feeding the output of the comparator directly back to the integrator. This works fine as long as the logic "1" (VDD) or logic "0" (ground) voltages are clean (have no noise on them). Noise on these fed back voltages directly adds or subtracts from the input signal and thus decreases the modulator's SNR. In any practical mixed-signal integrated circuit the digital supply and return are commonly noisy with variations in the hundreds of mV. As discussed in Ch. 28, it is common to separate the analog and digital power supplies on-chip (and so the modulator should be powered with the analog supply). Because of this, and the desire to set the fed back voltage V_{REF+} and V_{REF-} independent of VDD and ground, the output of the comparator is often connected to the simple 1-bit DAC circuit shown in Fig. 32.26. It's important to remember that the output of the DAC must be able to charge the integrator's input switching capacitance (see Fig. 32.7) to within the final desired resolution of the converter in half a clock cycle (before the ϕ_2 switches open).

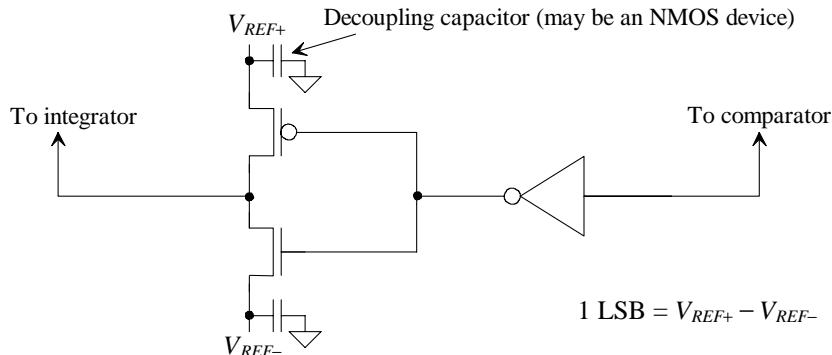


Figure 32.26 One-bit DAC for use in a NS modulator.

Understanding Averaging and the Use of Digital Filtering with the Modulator

In the following discussion we assume, as before, that $V_{REF+} = VDD = 1.5$ V and $V_{REF-} = \text{ground}$ (1 LSB = 1.5 V). We also assume the input voltage to the modulator falls within VDD and ground so that the output of the modulator doesn't saturate in a string of ones or zeroes (more on this below). In this section we want to discuss, intuitively, the operation of the digital filter used on the output of the modulator (see Fig. 32.9).

Before discussing the digital filter let's remember that we can recover the analog input voltage using an analog filter as shown in Fig. 32.10. This means that if the output of the modulator is a continuous string of zeroes, then the output of the analog filter will be zero volts. Other possible modulator outputs and their averages are shown in Fig. 32.27.

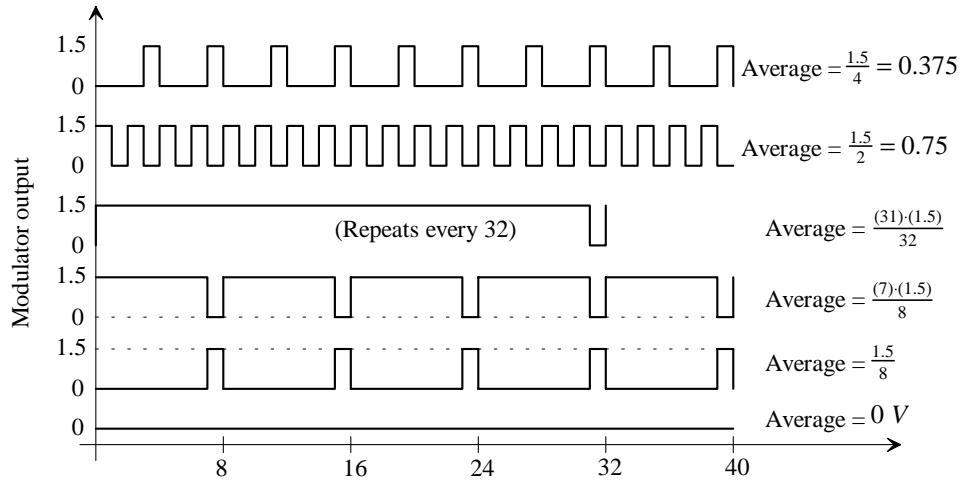


Figure 32.27 Modulator outputs and their corresponding averages.

Example 32.8

Plot the ideal I/O transfer curve for the 1-bit DAC. Also plot the non-ideal transfer curve if $V_{REF+} = 1.45$ V (instead of 1.5 V). Comment on how the output of the NS-modulator/decimator (the data converter) will be affected.

The transfer curves are shown in Fig. 32.28. The offset in the positive reference voltage results in a gain error in the data converter (but no nonlinearity). ■

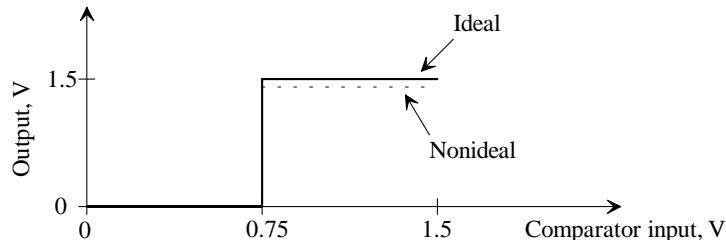


Figure 32.28 Ideal and nonideal transfer curves for the 1-bit DAC and comparator cascade.

Next let's consider the case where the modulator's input is a DC signal and our data converter's resolution is only limited by the number of samples we can take in a given time, KT_s . Equation (32.13) shows that as K approaches infinity, B approaches zero (DC). Figure 32.14 shows that at DC the spectral density of the modulation noise is zero. Feeding the output of our first-order modulator to a single accumulate-and-dump, see Figs. 31.39 and 31.41, can provide the needed digital filtering. As seen in Fig. 31.41,

increasing K causes the filter's amplitude response at DC to increase (the number of bits coming out of the counter increases) and the sidelobes move toward DC. Also, as seen in Fig. 31.58b, we don't have aliasing at DC, so the reduction in output clocking frequency (decimation) to f_s/K can be accomplished with the single accumulate-and-dump stage. However, there may still be aliasing from higher frequencies. Note that we have just described a first-order NS modulator driving a counter where the counter is reset and read-out every KT_s clock cycles and clocked at a rate of f_s . Practically, for large K , the limiting factor in the resolution of the NS modulator is the noise inherent in the circuit (mainly thermal and flicker noise sources from the MOSFETs) and, more importantly, the finite gain and linearity of the op-amp (more on this later). Note that in our perfect modulator a single output going high out of one-million outputs would correspond to an average input voltage of only 15 μV . This would also mean that we would need to average at least one million and one (1,000,001) modulator outputs with our simple counter for a constant output.

In our cascade of two accumulate-and-dumps, Eq. (32.27) and Fig. 32.19, we average K^2 samples. The cascade behaves, from a frequency response point of view, like a single accumulate-and-dump. For the cascade of L sinc filters, however, the number of modulator outputs averaged is

$$\text{Number of modulator outputs averaged} = L \cdot K - 1 \quad (32.34)$$

As we saw in Ex. 32.6, the cascade of sinc filters results in a weighted average of the filter's inputs. For our first-order modulator of Fig. 32.7 and the digital filter of Fig. 32.17 with a transfer function, once again, of

$$H(z) = \left[\frac{1-z^{-16}}{1-z^{-1}} \right]^2 \quad (32.35)$$

we perform a weighted average on 31 of the modulator's outputs. The time domain impulse response of this filter (without decimation), again see Ex. 32.6, is given by

$$\begin{aligned} y[nT_s] &= x[nT_s] + 2x[(n-1)T_s] + 3x[(n-2)T_s] + \dots + 15x[(n-14)T_s] + 16[(n-15)T_s] + \\ &\quad 15x[(n-16)T_s] + 14x[(n-17)T_s] + \dots + 2x[(n-29)T_s] + x[(n-30)T_s] \end{aligned} \quad (32.36)$$

For a continuous filter input of "1" the output of the filter is 256. Note that this is the same result we get with the cascade of two accumulate-and-dumps and is the "gain" (K^L) of the filter at DC. For our current discussion the minimum resolution we can represent with the maximum output value of 256 (realizing the minimum output value, which corresponds to a continuous modulator output of all zeroes, is 0) is $(1.5)/256 = 5.86 \text{ mV}$. We want this value to be less than the resolution calculated in Ex. 32.3, which was 6.14 bits ($1.5/2^{6.14} = 21.27 \text{ mV}$) so that our modulation noise, for a given bandwidth, limits the data converter's resolution and not the digital filter. This is why adding an additional sinc filter stage (say, $L = 3$) will not increase the data converter's resolution. The fundamental way to increase the first-order modulator's resolution is to increase the number of samples averaged (the oversampling ratio), K . Note also, that increasing L will have the undesirable effect of increasing droop in the bandwidth of interest.

Example 32.9

If the desired input to an *ideal* first-order NS modulator is a DC signal, would it be better to use a single sinc filter (a counter or accumulate-and-dump) or a cascade of two sinc filters of the form given by Eq. (32.35)?

Equation (32.20) was derived assuming we wanted to maximize the input signal bandwidth, B . If we are measuring a DC signal with, ideally, zero-bandwidth, then we want to minimize the digital filter's bandwidth to remove unwanted noise that may corrupt the DC signal. This means, for higher resolution with correspondingly longer conversion time, the single-stage filter is the best choice. ■

In Fig. 32.17 we indicated that the word size coming out of the filter, after dropping the lower two-bits, is seven-bits. This means, in two's complement, that outputs of 011 1111 (+63), 100 0000 (-64), and 000 0000 (0) correspond to the maximum input ($V_{REF+} - 1$ LSB), minimum input (V_{REF-}), and common-mode voltage ($V_{CM} = 0.75$ V) respectively, see Fig. 31.37. A continuous modulator output of "1" would correspond to an input of V_{REF+} , which would be outside the possible digital filter output words and result in the *incorrect* filter output code of 100 0000. Note that here $1 \text{ LSB} = (V_{REF+} - V_{REF-})/2^7 = 11.719 \text{ mV}$, which is, again, below the 21.27 mV fundamental RMS noise limit of the modulator in a bandwidth, B .

Next, let's consider the situation where one out of every 64 modulator outputs is a logic 1 (and the sequence repeats indefinitely). As we saw in Fig. 32.27, averaging this output results in an analog voltage of $1.5/64$ or 23.44 mV. Using Eq. (32.36), we can write the sequence of digital filter outputs (in decimal form and assuming our single-pulse input to the filter marks the beginning of the output) without decimation as

1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0, (followed
by 32 more zeroes and then the sequence repeats itself).

If decimation is employed, as in Fig. 32.17, then the outputs of the filter look like

1,15,0,0 (summed = 16)

noting that shifting in time doesn't change the sum of the sequence if the samples output by the decimation process are spaced apart by 16 (K). For example, the sequence

4,12,0,0 (summed = 16)

also has the same sum of 16. Figure 32.29 shows plots of our modulator input and the reconstructed filter output. Our modulator input is a constant DC signal of 23.44 mV while at the same time the digital filter output is a repeating sequence. What we are seeing is the *ripple* associated with passing the output of the modulator through the filter. Additional filtering, including the RCF, will reduce the ripple (because of the reduction in bandwidth). We should point out that this ripple represents a major difference between Nyquist-rate data converters and oversampling converters. Note how averaging 4 output samples after decimation or 64 samples before decimation results in the digital reconstructed output value matching the input value.

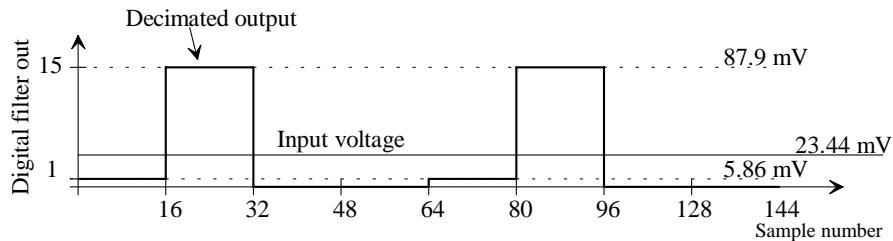


Figure 32.29 Ripple in the output of a digital filter. Note how the average of the filter output is equal to the input voltage.

Figure 32.30 shows the output of an RC filter, with a time constant of $1 \mu\text{s}$, when connected to the output of the modulator of Fig. 32.7. The modulator input voltage is 23.44 mV. The analog RC filter, like the digital filter, will not totally filter out the higher frequency components of the modulation noise and thus there will be some ripple on the output signal. Again, as in the digital filter, reducing the bandwidth of the filter (increasing the time constant) will reduce the peak-to-peak amount of ripple.

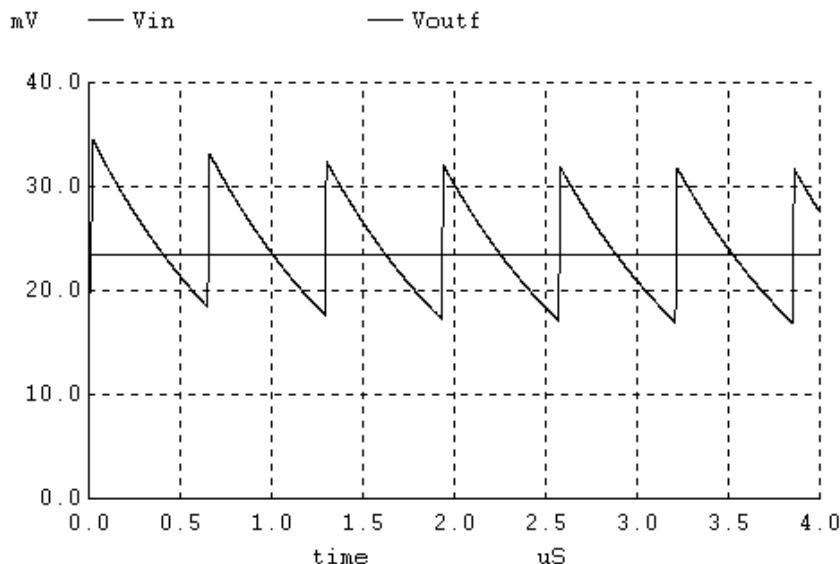


Figure 32.30 Showing how we have ripple on the output of an analog filter connected to a modulator with a DC input.

Example 32.10

What are the 7-bit, two's complement, representations of the numbers in Fig. 32.29 assuming they are originally represented using 9-bit, two's complement, words as in Fig. 32.17? What is the four-sample average of these words?

The 9-bit two's complement representations of 0, 1, and 15 are 1 0000 0000, 1 0000 0001, and 1 0000 1111, respectively. Dropping the lower two bits and knowing, for our 7-bit representation that 1 LSB = 11.719 mV, results in 100 0000 (0), 100 0000 (0), and 100 0011 ($V_{CM} - 61 \cdot 11.719$ mV = 35.157 mV). Averaging the filter outputs, as we did in Fig. 32.29, results in a value of 8.79 mV, which is different from the input voltage of 23.44 mV. By dropping the lower 2 bits we actually lost resolution. This can be confusing until we remember that at DC the possible resolution of the ideal modulator and digital filter is infinite. Over a bandwidth, B , however the resolution of the converter is limited to less than 7 bits as discussed in Ex. 32.3. Not throwing out the two lower bits is useful if additional digital filtering is used in the mixed-signal system. Otherwise the two lower bits are just random values (noting that when we averaged the four samples in Fig. 32.29 it was equivalent to passing the digital data through an additional lowpass, sinc averaging filter with $K = 4$). ■

Pattern Noise from DC Inputs (Limit Cycle Oscillations)

The ripple on the output of the filter can cause noise in the base spectrum of interest. The frequency of the ripple and the amplitude of the ripple depend on the DC input value. As compared with Fig. 32.30, Fig. 32.31 shows the RC filter output if the modulator input is changed to 0.75 V (the common-mode voltage). The frequency of the ripple is higher and the peak-to-peak amplitude of the ripple is smaller. The modulator output, with an input of 0.75, is a square wave of alternating ones and zeroes. The first harmonic of this signal is at half the clocking frequency or, in this example, 50 MHz. Since the ripple frequency lies

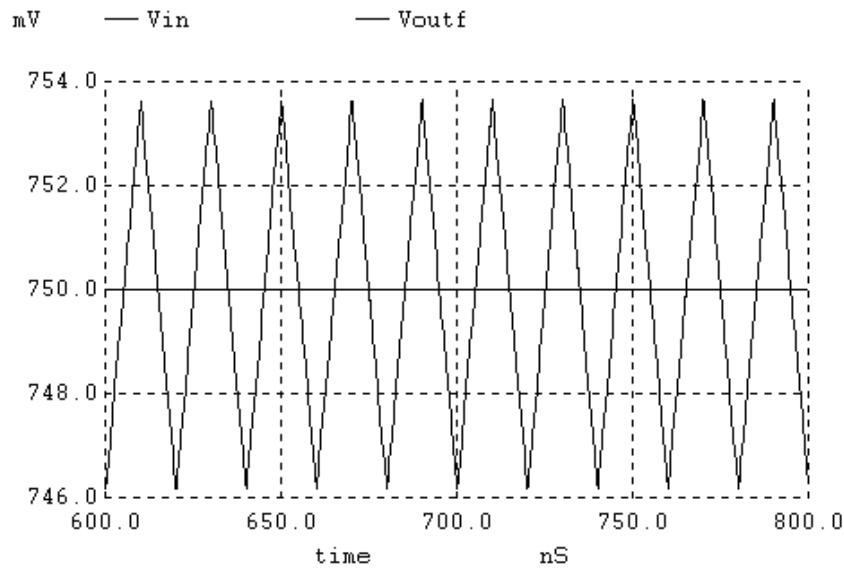


Figure 32.31 Filter ripple when input is the common-mode voltage of 0.75 V.

outside our base spectrum (which, from Ex. 32.3, is from DC to 3.125 MHz) it will not, in a significant way, affect the SNR. (In the digital filter a zero in the digital filter's transfer function will most likely fall at half the clocking frequency eliminating the ripple altogether and resulting in a constant filter output value.) The frequency of the ripple in Fig. 32.30, however, is 1/640 ns or 1.564 MHz, which is well within our base spectrum. The resulting tone will lower the SFDR and the SNR of the data converter. The question now becomes, "How do we minimize the possibility of unwanted tones appearing in the data converter's output spectrum?"

If we look at the digital filter output data shown in Fig. 32.29 we see that the "ripple" amplitude of the digital data is fully 87.9 mV peak-to-peak or significantly above the data converter's LSB value (noting that after the RCF/half-band filter this ripple value will be reduced). Looking at this figure, we see that it would be better to spread or flatten the data out over all four cycles of the repeating waveform. Although we may still have a tone, or repeating sequence, at a frequency in the base spectrum, the amplitude of the tone will be well below the LSB of the data converter (and so it won't affect the SFDR of the data converter). To accomplish this spread or randomization we can add a noise dither source (see the last chapter) to our basic NS modulator, as seen in Fig. 32.32. By applying the dither to the input of the comparator (quantizer) the dither will be noise-shaped like the quantization noise (the spectral content of the dither, Eq. [31.76], is less important).

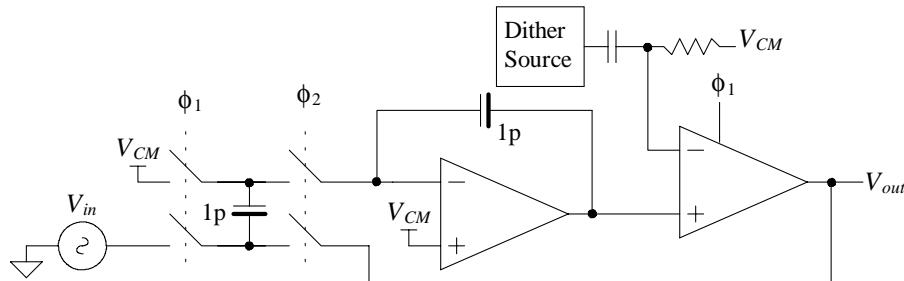


Figure 32.32 Adding a dither source to a first-order NS modulator.

The output of the modulator in our discussion and generation of Fig. 32.29 was a single bit going high followed by 63 zeroes as shown in Fig. 32.33a. Note how the period of the output is repetitive. If our dither source is used, the output may look something like what is seen in Fig. 32.33b. The average of the waveform, over several cycles, is the same as in part (a), while the period of the waveform varies and randomizes the power contained in a particular frequency (tone). What this means is that the output spectrum of a data converter, with a DC input, will not contain tones at specific frequencies sticking up above the noise floor and resulting in a decrease in the SFDR. However, the peak-to-peak amplitude of the ripple in the time domain may actually get worse. As seen in Fig. 32.33b, the occasional shorter spacing between the output ones can result in a larger digital filter output code. Again, we should point out that this variation, or ripple, in the output code is a basic difference between a NS modulator-based data converter and a Nyquist-rate data converter.

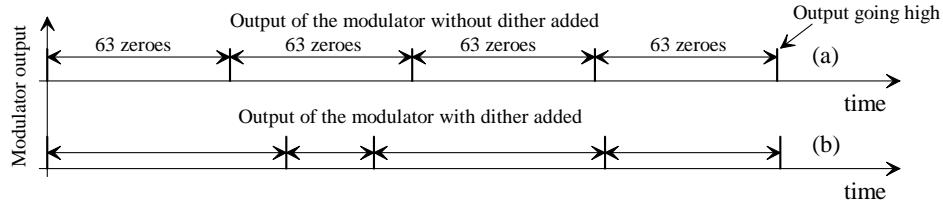


Figure 32.33 Output of our modulator (a) without and (b) with a dither source.

As a final example, let's consider how a tone can occur in a modulator output that has heavy transition densities (numerous one-zero transitions). If the input to the modulator is the common mode voltage, V_{CM} , of 0.75 V, then the output of the modulator is an alternating sequence of ones and zeroes. Changing the input voltage upwards by a small amount will result in the output of the modulator staying high once in a while instead of going low (resulting in two consecutive logic one outputs). An example is seen in Fig. 32.34 where the input to the modulator was increased to 0.77 V. As the double ones are spaced apart by approximately 350 ns, we can estimate a tone in the resulting output spectrum at a frequency of 1/350 ns or 2.86 MHz.

Finally, note that unwanted tones are usually not a problem if the input signal is busy and random (not DC as discussed in this section). Later in the chapter, we discuss second-order modulators that utilize two integrators. The second integration helps to spread the repeating sequences out over a longer period of time so that, hopefully, negligible unwanted tone energy is present in the base spectrum.

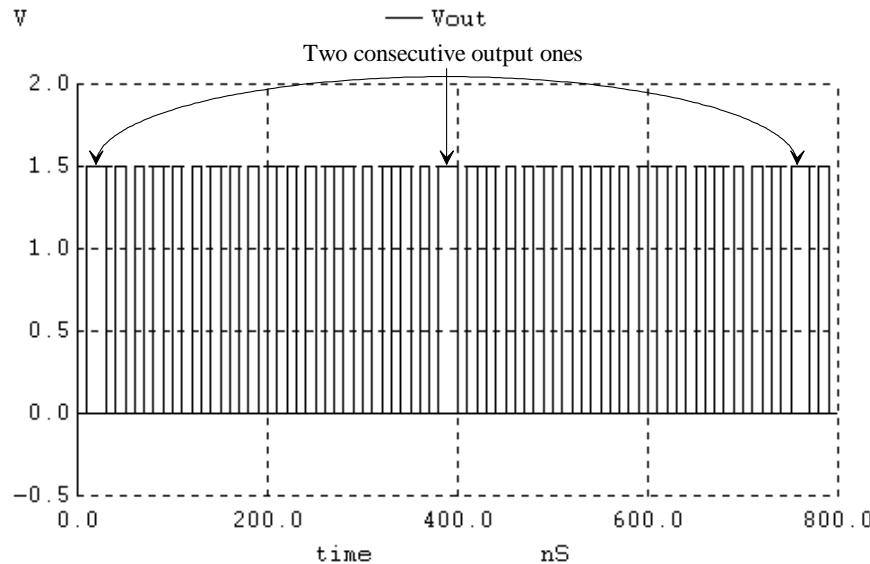


Figure 32.34 Modulator output showing how tones can occur with higher transition density.

Integrator and Forward Modulator Gain

So far we haven't discussed the shape or amplitude of the integrator's output. Because we are using near-ideal components in our simulations, we haven't seen any limitations due to the finite op-amp output swing. Figure 32.35 shows the integrator's output for the input and output signals shown in Fig. 32.8 (using the modulator of Fig. 32.7). Clearly the output swing of the op-amp is beyond the power supply rails. If the transistor-level model of the op-amp were to replace the ideal op-amp, the integrator's output would *saturate* at voltages less than VDD ($= 1.5$ V here) or greater than ground. While in some situations op-amp saturation is not necessarily bad (the gain of the integrator goes to zero), it is desirable to understand how decreasing forward loop gain affects the performance of the modulator. Note also, in Fig. 32.35, how the output of the integrator makes the largest change when it passes through the comparator reference voltage, $V_{CM} = 0.75$ V, since the fed back signal, the comparator output (a full-scale signal), is input to the integrator.

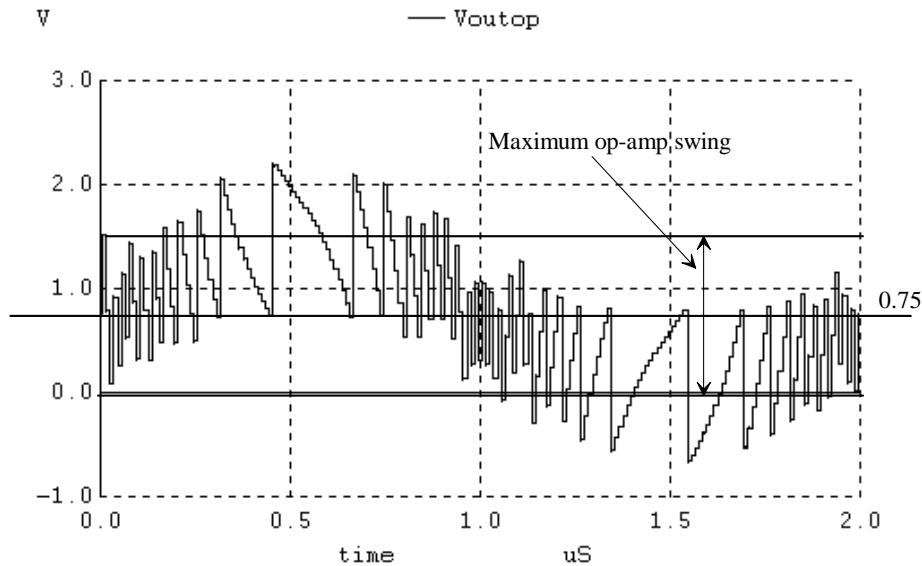


Figure 32.35 Output swing limitations in the op-amp (integrator).

Consider the linearized model of our first-order NS modulator shown in Fig. 32.36. The gain of the integrator, see Eq. (31.135) or Fig. 31.79, is given by

$$G_I = \frac{C_I}{C_F} \quad (32.37)$$

We have also drawn the comparator with a gain. Up until this point we have assumed the gain of the comparator is unity. We'll comment on this more in a moment. Let's define the modulator's forward gain as

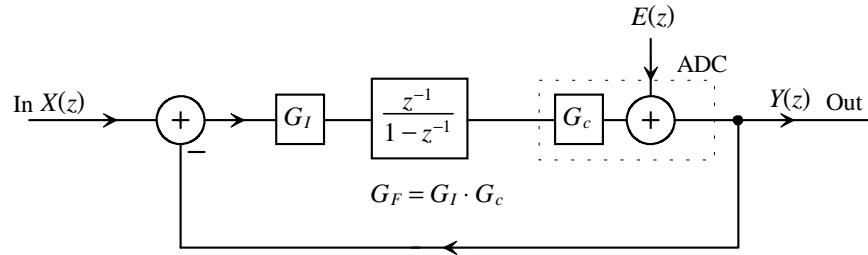


Figure 32.36 Block diagram of a NS modulator showing forward gains.

$$G_F = G_I \cdot G_c \quad (32.38)$$

We can rewrite Eq. (32.4) using this gain as

$$Y(z) = \frac{z^{-1} \cdot G_F}{1 + z^{-1}(G_F - 1)} \cdot X(z) + \frac{1 - z^{-1}}{1 + z^{-1}(G_F - 1)} \cdot E(z) \quad (32.39)$$

If G_F approaches zero (the integrator saturates while the comparator gain stays finite), then the output of the modulator is the sum of the integrated input and the quantization noise. (This is bad.) Since the quantization noise is not spectrally shaped it will be difficult to filter the modulator's output to recover the input signal. If the forward gain is greater than two, then, as seen in Fig. 31.62 and the associated discussion, the poles of the transfer function reside outside the unit circle and the modulator will be unstable. We can restrict the values of the forward gain to

$$0 \leq G_F \leq 2 \quad (32.40)$$

Ideally, however, the gain is one.

Example 32.11

Show, using SPICE simulations and the modulator of Fig. 32.7, that an integrator gain of 0.4 will result in an op-amp output range well within the power supply range.

Figure 32.37a shows a schematic of the modulator with $G_I = 0.4$. Figure 32.37b shows the output of the integrator (the output of the op-amp) in the modulator of part (a) with the input sinewave shown in Fig. 32.8. The output swing is limited to roughly 80% of the supply range. *For general design it is desirable to set our integrator gain to 0.4.* This ensures our integrator doesn't saturate unless the input to the modulator goes outside the supply voltage range.

It's interesting to note that in both modulators, Fig. 32.7 and Fig. 32.37, the forward gain is unity. This is a result of the effective gain of the comparator changing forcing the forward gain, controlled by the fed back signal, to unity. What this means is that our modulator functions as expected with a signal gain of one (Eq. [32.4] is valid) whether G_I is 1 or 0.4. We discuss how this change in comparator gain occurs next. ■

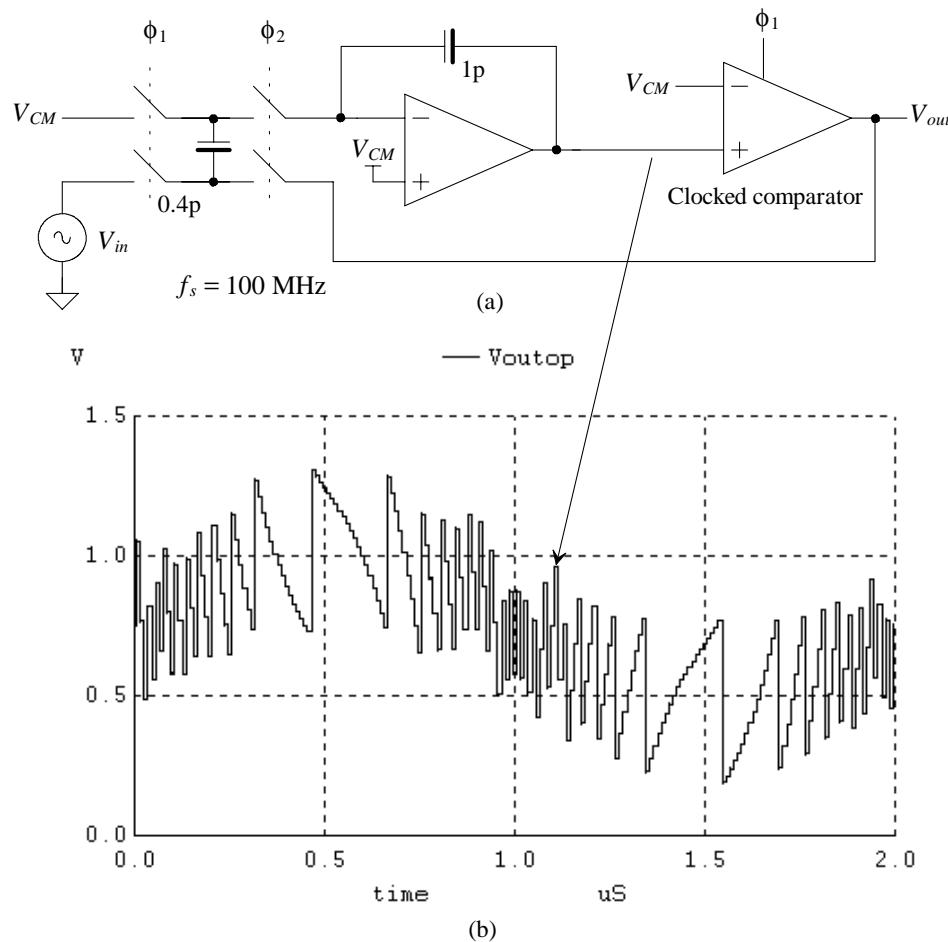


Figure 32.37 (a) First-order NS modulator with an integrator gain of 0.4, and (b) the output of the op-amp.

Figure 32.38 shows the transfer curves for the comparator. The x-axis, the comparator input, is the output of the integrator in our modulator. Shrinking the integrator's output swing while holding the output swing of the comparator at the supply rails (1.5 V) results in an increase in effective comparator gain. This gain variation, with the integrator output swing, helps to set the forward gain of the modulator to precisely 1. We can write this using equations as

$$\underbrace{\frac{\text{Integrator gain, } G_I}{\text{Modulator input}}}_{\text{Integrator output}} \cdot \underbrace{\frac{\text{Comparator gain, } G_c}{\text{Integrator output}}}_{\text{Comparator(modulator) output}} = \underbrace{\frac{G_F}{\text{Modulator output}}}_{\text{Modulator input}} \quad (32.41)$$

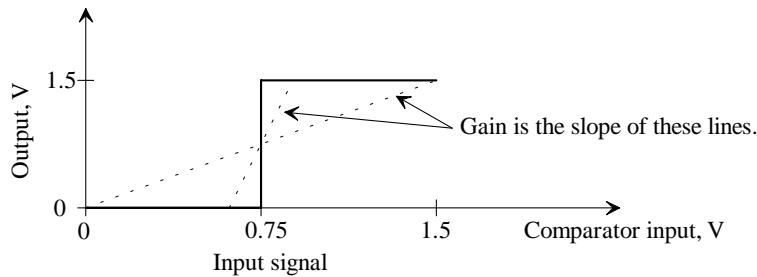


Figure 32.38 Comparator gain as a function of input voltage.

If the modulator is functioning properly, then the average value of the modulator output will be equal to the modulator input and thus $G_F = 1$. It's interesting to note that this result (*precise integrator gain isn't important*) will apply to any integrator that is directly followed by an ADC.

Before leaving this section, let's point out a couple of problems with a noise-shaping modulator that uses a multibit ADC, Fig. 32.39. Since the output of the integrator is the input signal to the ADC, the limited integrator output swing will directly affect the range of ADC output codes. Limiting the range of ADC output codes will then limit the allowable range of modulator inputs unless scaling is used (shifting the output codes or sizing of capacitors in the DAI). Next, notice in Fig. 32.39 how the variation in the gain of the ADC, with input signal, is more limited than the gains attainable with the simple comparator of Fig. 32.38. Limiting the range of ADC gains can result in modulator

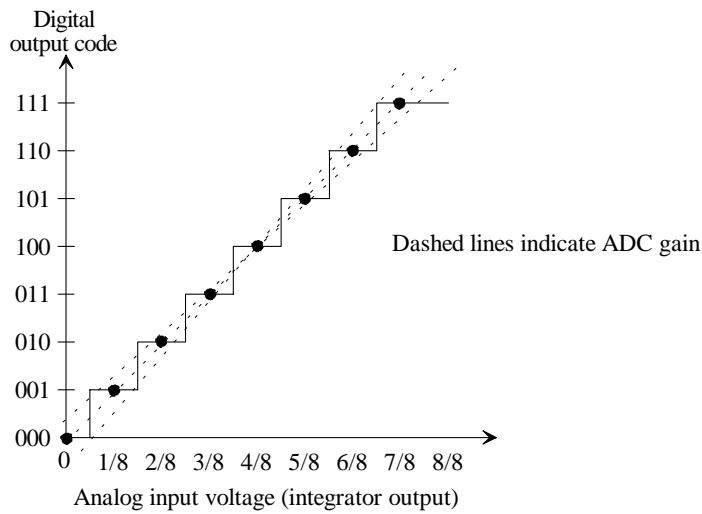


Figure 32.39 A 3-bit ADC.

forward gains that are not exactly unity. This is especially true at high input frequencies where the gain of the integrator is low. However, if the integrator gain is high, the effective gain of the ADC is not important. The point here is that using a multibit ADC will increase the open-loop gain requirements of the op-amp used in the integrator.

Comparator Gain, Offset, Noise, and Hysteresis

It's of interest to determine how the performance of the comparator affects the operation of the modulator. Both the comparator's offset and input-referred noise, Fig. 32.40a, can be referred back to the modulator's input, Fig. 32.40b. By doing so we can determine how they effectively change the input signal seen by the modulator. As seen in Fig. 32.40, the high gain of the integrator, $A(f)$, reduces the effect of the comparator's noise and offset on the input signal. For example, if the gain of the integrator at DC is 1,000 and the offset voltage of the comparator is 50 mV, then the input-referred offset is only 50 μ V.

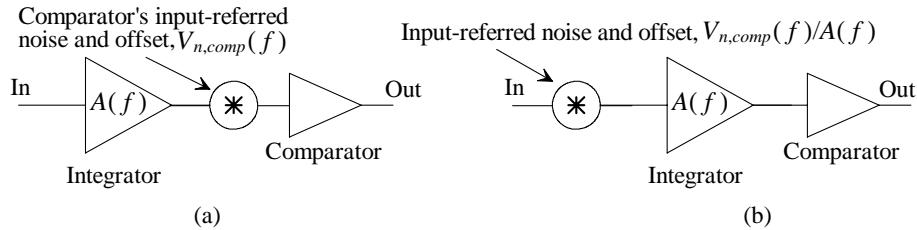


Figure 32.40 (a) Referring the comparator offset and noise to (b) the input of the modulator.

To determine the minimum gain and maximum allowable hysteresis requirements of the comparator, let's review Fig. 32.37. We see that when the output of the comparator changes states, the output of the integrator changes by at least

$$\text{Change in integrator output} = G_I \cdot (VDD - V_{CM}) = \frac{C_I}{C_F} \cdot \frac{V_{REF+} - V_{REF-}}{2} \quad (32.42)$$

For the modulator of Fig. 32.27 this equation can be evaluated as 0.3 V. As long as the hysteresis is much less than this value and the gain of the comparator (1.5/0.3 or 5) is large enough so that the comparator can make a full output transition with this input difference, then the modulator will function properly. Very simple, low-performance comparator designs can be used while not affecting the modulator's performance.

Op-Amp Gain (Integrator Leakage)

Now that we've discussed the gain of the comparator, let's determine how high the open-loop gain of the op-amp must be for proper integrator action. With low op-amp gain, some of the charge stored on the integrator's input capacitor, C_I , is not transferred to the feedback capacitor, C_F . This loss of charge is sometimes referred to as *integrator leakage*. The charge on the input capacitance effectively leaks off when it is transferred to the feedback capacitance.

We can write the open-loop, frequency-dependent gain of the op-amp as $A_{OL}(f)$. The output voltage of the op-amp is then $v_{out} = A_{OL}(f)(v_+ - v_-)$, where v_+ is our common-mode voltage V_{CM} (the noninverting terminal of the op-amp), see Fig. 31.78, and v_- is the op-amp's inverting input terminal. Following the procedure to derive Eq. (31.134), we can rewrite Eq. (31.132) with finite op-amp gain as

$$Q_2 = C_I \left(V_{CM} - \frac{v_{out}[nT_s]}{A_{OL}(f)} - v_2[nT_s] \right) \quad (32.43)$$

or, rewrite Eq. (31.134) to include the effects of finite op-amp gain to get

$$V_{out}(z) = \frac{C_I}{C_F} \cdot \frac{V_1(z) \cdot z^{-1/2} - V_2(z)}{\left(1 + \frac{C_I}{C_F A_{OL}(f)} \right) - z^{-1}} \quad (32.44)$$

Using this result in Eq. (32.3) and, as discussed in the last section, assuming the forward gain of the modulator, G_F , is one gives

$$Y(z) = \frac{z^{-1}}{1 + \frac{C_I}{C_F A_{OL}(f)}} \cdot X(z) + \frac{1 + \frac{C_I}{C_F A_{OL}(f)} - z^{-1}}{1 + \frac{C_I}{C_F A_{OL}(f)}} \cdot E(z) \quad (32.45)$$

The gain error term

$$\epsilon_{gain} = \frac{C_I}{C_F} \cdot \frac{1}{A_{OL}(f)} \quad (32.46)$$

is ideally zero so that Eq. (32.45) reduces to Eq. (32.4). Note how reducing the integrator's gain, C_I/C_F , reduces the gain error while increasing the gain required of the comparator. Note also how the denominator term is common in both the signal and the noise. This term results in a data converter gain error (it behaves as if it were an op-amp offset voltage that is a function of the integrator's output amplitude [which results in the gain error] and frequency), but it will not affect the modulator's SNR. To determine the increase in the modulator's output noise (the change in the shape of the modulation noise) we need to look at the noise transfer function including the effects of the gain error

$$NTF_e(z) = (1 + \epsilon_{gain}) - z^{-1} \quad (32.47)$$

or, in the frequency domain,

$$|NTF(f)|^2 = 2(1 + \epsilon_{gain}) \left(1 - \cos 2\pi \frac{f}{f_s} \right) + \epsilon_{gain}^2 \quad (32.48)$$

Following the same procedure used to arrive at Eq. (32.15) and assuming constant op-amp gain, $A_{OL}(f)$, from DC to B , results in

$$V_{Qe,RMS}^2 = 2 \cdot \frac{V_{LSB}^2}{12f_s} \cdot \left[4(1 + \epsilon_{gain}) \frac{\pi^2}{f_s^2} \cdot \frac{1}{3} \cdot \left(\frac{f_s}{2K} \right)^3 + \epsilon_{gain}^2 \cdot \frac{f_s}{2K} \right] \quad (32.49)$$

noting that if $\epsilon_{gain} = 0$, this equation reduces to Eq. (32.15).

If we assume the contribution to the noise from the error term squared, ε_{gain}^2 , is small, which is valid for op-amp gain

$$A_{OL}(f) > K \text{ (the oversampling ratio)} \quad (32.50)$$

over the frequency range of DC to B , then we can rewrite Eq. (32.16), to include the effects of finite op-amp gain, as

$$\text{SNR}_{gerr} = 6.02N + 1.76 - 20 \log \frac{\pi}{\sqrt{3}} + 20 \log K^{3/2} - 10 \log (1 + \varepsilon_{gain}) \quad (32.51)$$

The largest degradation in the SNR, resulting from integrator leakage, can be estimated as 0.5 dB if $K \geq 8$ ($\varepsilon_{gain} \approx 1/8$ neglecting G_I). The minimum gain-bandwidth product of the op-amp is estimated as

$$\text{Op-amp unity gain frequency, } f_u = K \cdot B = f_s/2 \quad (32.52)$$

assuming the op-amp is rolling off at 20 dB/decade at B (a dominant pole compensated op-amp). Otherwise, the minimum gain of the op-amp can be estimated simply as the oversampling ratio, K .

To illustrate typical op-amp requirements, let's consider the modulator of Fig. 32.37 with $K = 16$ and $B = 3.125$ MHz (see Ex. 32.3). The f_u of the op-amp is estimated, using Eq. (32.51), as 50 MHz. If the open-loop response of the op-amp starts to roll off at 10 kHz, then the DC gain of the op-amp must be at least 5,000. However, we could also use an op-amp with a DC gain of 100 (remembering low integrator gain increases the undesirable effects [noise and offset] of the comparator on the performance of the modulator) that rolls off at 500 kHz.

Op-Amp Settling Time

Equation (32.52) can be used, for the moment, to provide an estimate for the settling time requirements of the op-amp in a first-order modulator. Assuming the settling time is linear, and not slew-rate limited, we can write the change in the op-amp's output (assuming a dominant pole compensated op-amp, see Eqs. [27.37] and [27.38]) as

$$v_{out} = V_{outfinal}(1 - e^{-t/\tau}) \text{ where } \tau = \frac{1}{2\pi f_u \cdot \beta} \quad (32.53)$$

where, for the DAI (see Fig. 32.41), the feedback factor is

$$\beta = \frac{C_F}{C_F + C_I} \quad (32.54)$$

The feedback factor is 0.714 in the modulator of Fig. 32.37. The output of the DAI, v_{out} , must settle in a time, $t (< T_s/2)$, to some percentage of an ideal value, $V_{outfinal}$. Solving for this percentage using Eqs. (32.52), (32.53), and (32.54) and assuming $T_s/2 = t$ results in

$$\frac{v_{out}}{V_{outfinal}} \times 100\% = 1 - \exp \left[-\frac{\pi}{2} \cdot \frac{C_F}{C_F + C_I} \right] \times 100\% \quad (32.55)$$

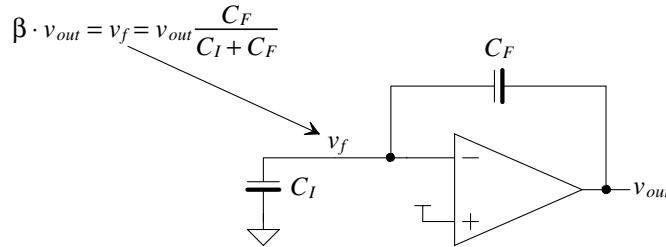


Figure 32.41 The feedback factor in the DAI.

The output will only reach 67% of its ideal final value in the modulator of Fig. 32.37 when the op-amp used has a unity gain frequency of $f_s/2$.

In deriving Eq. (32.55) we used an op-amp unity gain frequency specified by Eq. (32.52) to determine the settling response of the integrator. If the settling is linear then incomplete settling will result in a constant DAI gain error (0.67 above). Every time the output changes it will change by some constant percentage of its ideal value. Rewriting the transfer function of our DAI to include this constant gain error results in

$$V_{out}(z) = \frac{C_I}{C_F} \cdot \overbrace{\left(1 - e^{-\pi\beta(f_u/f_s)}\right)}^{\text{Settling gain error, } G_s} \cdot \frac{V_1(z) \cdot z^{-1/2} - V_2(z)}{1 - z^{-1}} \quad (32.56)$$

Full, or complete, settling requires that the op-amp's unity gain frequency, f_u , be much larger than the sampling frequency, f_s (in other words we can't use Eq. [32.52] to specify the required bandwidth of the op-amp if settling time is important). The constant gain error, resulting from incomplete settling, can be tolerated in the first-order modulator because the integrator is directly followed by a comparator, as discussed earlier. In some of the modulator topologies, though, the integrator is not followed by a comparator so settling time becomes more important. To determine to what percentage the integrator output must settle in these topologies, a gain term, say G_s , is added to the linearized block diagram of the modulator (integrator). The transfer function of the modulator is then evaluated to determine the allowable values of G_s for the application.

It's important to realize that we are assuming the op-amp doesn't experience slew-rate limitations. If slewing is present, then the added gain term, in Eq. (32.56), will not be a constant and will introduce *distortion* into the modulator's output spectrum (whether a comparator follows the integrator or not).

Op-Amp Offset

The operation of the DAI is subject to the op-amp's offset. It can be shown that this offset will effectively add (or subtract) from the common-mode voltage, V_{CM} , and thus effectively shift the input signals upwards or downwards. The resulting modulator output will then show an offset equal to the op-amp's offset. To circumvent this problem, offset storage can be used in the integrator.

Op-Amp Input-Referred Noise

We'll discuss the calculation of the DAI input-referred noise (PSD of $V_{n,DAI}^2(f)$), at the transistor level, in the next chapter. Here we discuss how the DAI's unwanted noise contributions affect the SNR of the modulator, assuming we know $V_{n,DAI}^2(f)$. Figure 32.42 shows the modulator's input-referred noise source, $V_{n,ckt}(f)$, in series with the input signal. This noise source, with units of $\text{V}/\sqrt{\text{Hz}}$, includes both the integrator's and the comparator's contributions. However, as discussed earlier, the noise contributions from the comparator are usually negligible.

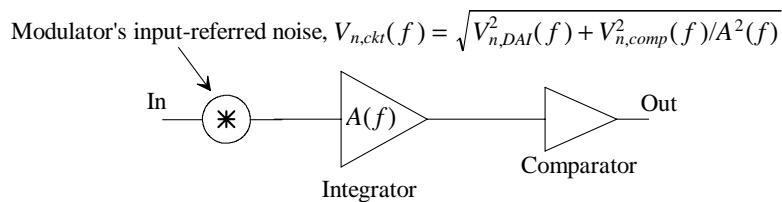


Figure 32.42 The modulator's input-referred noise contributions from both the comparator and the integrator.

Because the modulator's input-referred noise adds directly to the input signal, we can use the derivations developed earlier in the chapter. As specified in Eq. (32.4), the modulator's input, and thus its input-referred noise, pass through the modulator with a delay of z^{-1} . If we assume the modulator's input-referred noise is white and bandlimited to $f_s/2$ such that

$$V_{n,ckt}(f) = \frac{V_n}{\sqrt{f_s}} \quad \text{for } f < f_s/2 \quad (32.57)$$

then passing the output of the modulator through an ideal lowpass filter with a bandwidth of B ($= f_s/[2K]$) results in

$$V_{ckt,RMS} = \sqrt{2 \cdot \int_0^B \frac{V_n^2}{f_s} \cdot df} = \frac{V_n}{\sqrt{K}} \quad (32.58)$$

Noting that not passing the output of the modulator through a lowpass filter results in an RMS output noise of V_n , we see that the averaging filter (the lowpass filter) reduces the noise by the root of K . We could also think of the filtering as reducing the PSD of the modulator's input-referred noise by K . Remembering the jitter discussion from the last chapter, we see a direct parallel in the derivations of how averaging affects the RMS value of a random signal (noise or jitter).

Finally, as used in Ex. 31.15, we can estimate the finite SNR of a data converter from quantization noise, jitter, and circuit noise using

$$V_{n,RMS} = \sqrt{V_{Qe,RMS}^2 + V_{jitter,RMS}^2 + V_{ckt,RMS}^2} \quad (32.59)$$

and

$$\text{SNR} = 20 \cdot \log \frac{V_p/\sqrt{2}}{V_{n,RMS}} \quad (32.60)$$

where V_p is the peak amplitude of an input sinewave, see Eq. (31.1), and

$$V_{\text{jitter},\text{RMS}} = \sqrt{P_{\text{AVG,jitter}}} \quad (\text{see Eq. [31.47]}) \quad (32.61)$$

Practical Implementation of the First-Order NS Modulator

As discussed in Ch. 27, switched-capacitor circuits suffer from the problems of capacitive feedthrough and charge injection. To reduce these effects, fully-differential circuit topologies are used. It could be stated that if reasonable size capacitors and dynamic range are required, fully-differential topologies are a necessity simply because they subtract out, to a first-order, the voltage changes on the switched-capacitors resulting from these problems. In addition, again as discussed in Ch. 27, fully-differential topologies are used because they improve power supply and substrate-coupled noise rejection and improve distortion (even-order harmonics cancel).

Figure 32.43 shows the fully-differential implementation of the DAI of Fig. 31.78. The inputs are now differential, that is, now $v_1 = v_{1+} - v_{1-}$ and $v_2 = v_{2+} - v_{2-}$, as is the output of the integrator, $v_{out} = v_{out+} - v_{out-}$. The fully-differential DAI has the same transfer function as the single-ended DAI assuming the input signals are differential.

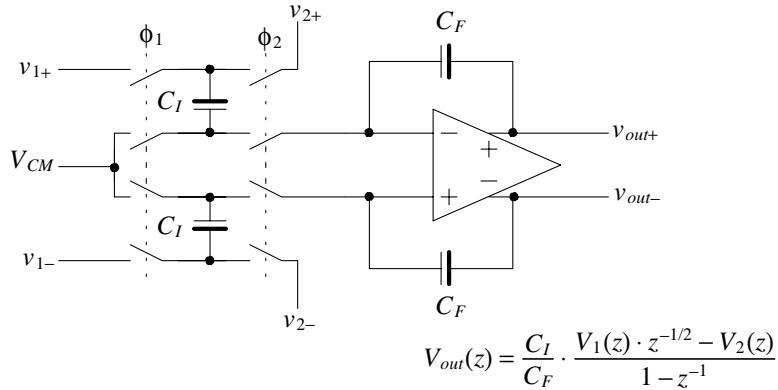


Figure 32.43 Fully-differential discrete-analog integrator (DAI) implementation.

It's important to understand the signal levels in the fully-differential DAI. Let's assume $V_{CM} = 0.75$ V and the input voltages can range in amplitude from 0 to 1.5 V. Assuming the input is balanced correctly if $v_{1+} = 0.85$ V, then v_{1-} must equal 0.65 V. The maximum input voltage is $v_{1max} = 1.5 - 0 = 1.5$ V. The minimum input signal, on the other hand, is $v_{1min} = 0 - 1.5 = -1.5$ V. The range of inputs, or outputs, is then 3 V or twice the range of the single-ended DAI.

Figure 32.44 shows the implementation of a first-order NS modulation utilizing a fully-differential DAI. Figure 32.45 shows the SPICE model used for a differential input/output op-amp (see also, Fig. 26.29). We'll use this model to simulate the operation of the modulator of Fig. 32.44 with the input signals and capacitor sizes used in Fig. 32.37 ($f_s = 100$ MHz, $C_I = 0.4$ pF, and $C_F = 1$ pF, $f_{in} = 250$ kHz, and a 0.7 V peak input sinewave [the input sinewave, $V_{in+} - V_{in-}$, has a peak amplitude of 2.8 V]).

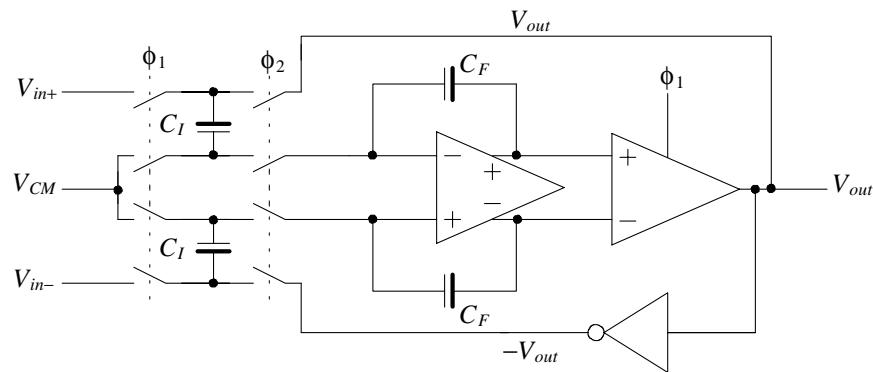


Figure 32.44 Fully-differential implementation of a first-order NS modulator.

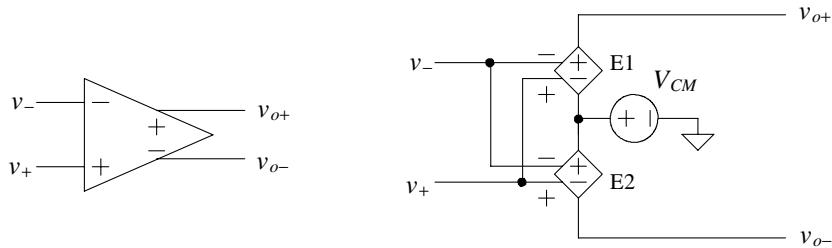


Figure 32.45 SPICE modeling a differential input/output op-amp with common-mode voltage.

Figure 32.46 shows the simulation results for the outputs of the modulator of Fig. 32.44 after being passed through two RC filters with time constants of 100 ns. Passing a single modulator output to the decimating filter would result in an output that is half the input signal amplitude, which can be compensated for at the output of the filter by a shift-left operation (multiply by two). Note, because the gain of the integrators is 0.4, the integrator output's swing is at most 80% of the supply rails. Also note that the input common-mode voltage of the op-amp remains at 0.75 V. This is important as the design of the op-amp becomes more challenging if the common-mode voltage is not constant. The finite op-amp common-mode rejection ratio (CMRR) can introduce distortion into the output of the modulator. Because many input signals will not be fully-differential, we briefly discuss differential modulator design with single-ended inputs next.

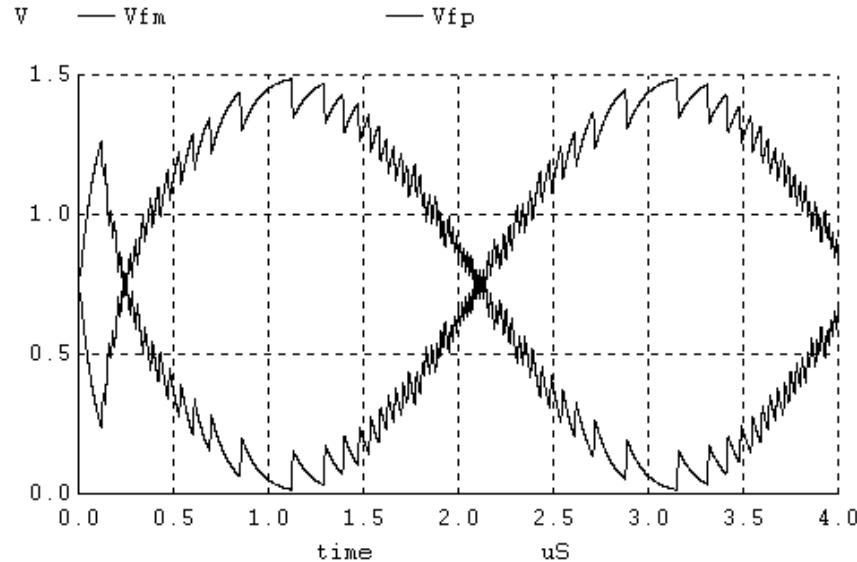


Figure 32.46 Outputs of the fully-differential first-order modulator after RC filtering.

Fully-Differential Modulator with a Single-Ended Input

If we connect our modulator's minus input, V_{in-} , to the common mode voltage, V_{CM} , we can apply a single-ended input to the modulator's V_{in+} input. We need to note several differences when the modulator is used with a single-ended input. The maximum input signal is now half of the modulator's input range. This means that we can increase our integrator's gain to 0.8 and still avoid DAI output saturation. It also means that our modulator output range will be at most half of the full-scale range, Fig. 32.47a. Finally, and probably most importantly, the op-amp's input common mode voltage now changes with the input signal, Fig. 32.47b, which may result in the input diff-amp used for the first stage in the op-amp shutting off (see also Eq. [34.8]).

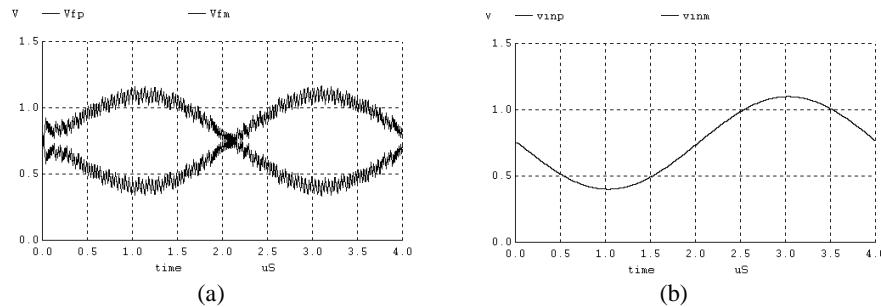


Figure 32.47 (a) Filtered modulator outputs with full-scale (1.5 V peak-to-peak) single-ended input, and (b) how the input common-mode voltage of the op-amp changes.

32.1.3 Second-Order Noise-Shaping

If we review Eq. (32.4), we might wonder if further filtering of the quantization noise, $E(z)$, can result in an improvement in the data converter's SNR over an input signal bandwidth B . The second-order modulator's output shows a double differentiation of the quantization noise

$$Y(z) = z^{-1}X(z) + (1 - z^{-1})^2 E(z) \quad (32.62)$$

The modulation noise may then be written, see Eqs. (32.10) and (32.11), as

$$|NTF(f)|^2 \cdot |V_{Qe}(f)|^2 = \frac{V_{LSB}^2}{12f_s} \cdot 4 \left(1 - \cos 2\pi \frac{f}{f_s}\right)^2 \quad (32.63)$$

Figure 32.48 shows a comparison between the modulation noise of first- and second-order NS modulators. Notice how the modulation noise is "flatter" in the bandwidth of interest.

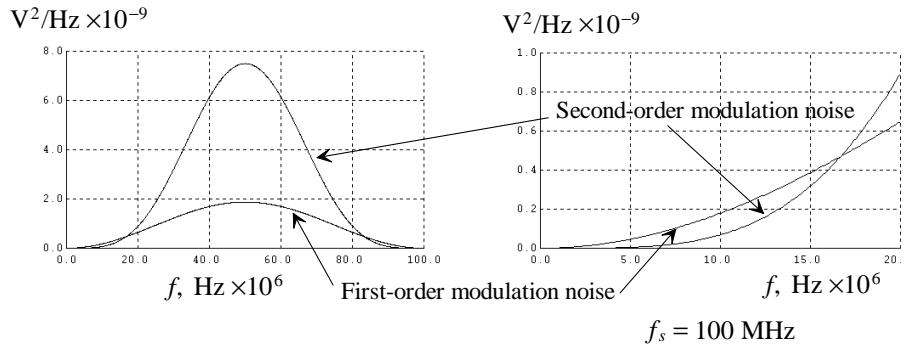


Figure 32.48 Comparing first- and second-order NS modulator's modulation noise.

If we restrict our frequency range to frequencies less than $f_s/2$, then we can rewrite Eq. (32.63) as

$$|NTF(f)| \cdot |V_{Qe}(f)| = \frac{V_{LSB}}{\sqrt{12f_s}} \cdot 4 \sin^2 \pi \frac{f}{f_s} \quad (32.64)$$

Calculating the RMS quantization noise in a bandwidth B results in

$$V_{Qe,RMS} \approx \frac{V_{LSB}}{\sqrt{12}} \cdot \frac{\pi^2}{\sqrt{5}} \cdot \frac{1}{K^{5/2}} \quad (32.65)$$

with an increase in the SNR of

$$\text{SNR}_{ideal} = 6.02N + 1.76 - 12.9 + 50 \log K \quad (32.66)$$

Every doubling in the oversampling ratio results in an increase in SNR of 15 dB or 2.5 bits increase in resolution! Figure 32.49 shows a comparison between simple oversampling, first-order NS, and second-order NS-based data converters. Note that, as discussed earlier, the oversampling ratio is generally greater than or equal to eight.

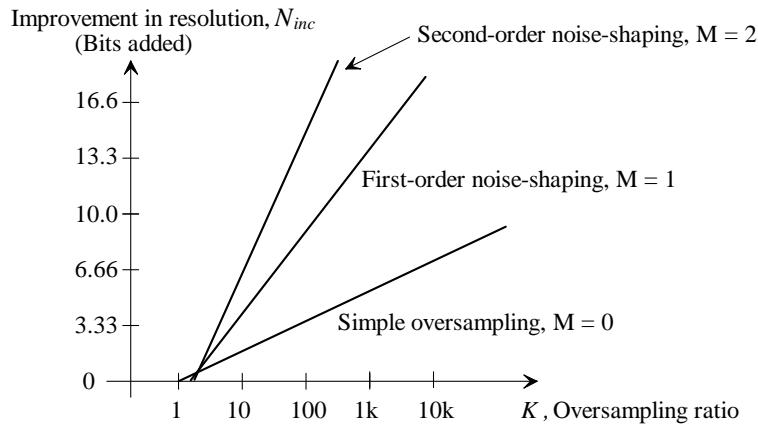


Figure 32.49 Comparing improvement in modulator resolution.

Second-Order Modulator Topology

Consider the block diagram of a NS modulator shown in Fig. 32.50 (see Fig. 31.82). The transfer function of this modulator may be written as

$$Y(z) = \frac{A(z)}{1 + A(z)B(z)} \cdot X(z) + \frac{1}{1 + A(z)B(z)} \cdot E(z) \quad (32.67)$$

Comparing this equation to Eq. (32.62), we can solve for the forward and feed-back circuit blocks, $A(z)$ and $B(z)$, by equating coefficients

$$STF(z) = \frac{A(z)}{1 + A(z)B(z)} = z^{-1} \quad (32.68)$$

and

$$NTF(z) = \frac{1}{1 + A(z)B(z)} = (1 - z^{-1})^2 \quad (32.69)$$

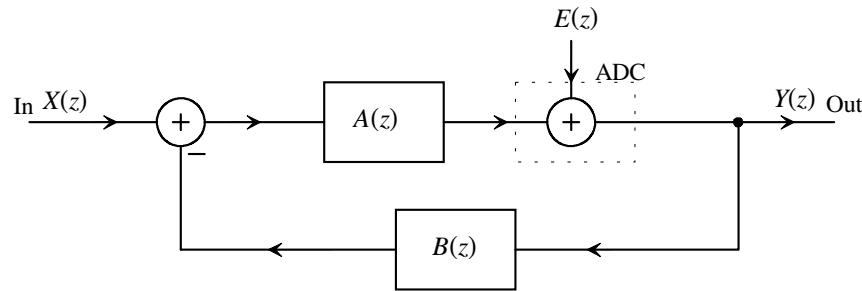


Figure 32.50 Block diagram of a feedback modulator.

The results are

$$A(z) = \frac{z^{-1}}{(1-z^{-1})^2} \quad (32.70)$$

and

$$B(z) = 2 - z^{-1} \quad (32.71)$$

The second-order modulator can be implemented using the topology shown in Fig. 32.51a. The output of $B(z)$ is the sum of the modulator output and the differentiated, $(1-z^{-1})$, modulator output. We can redraw the block diagram of Fig. 32.51a, as shown in Fig. 32.51b, resulting in the implementation of a second-order NS modulator shown in Fig. 32.51c.

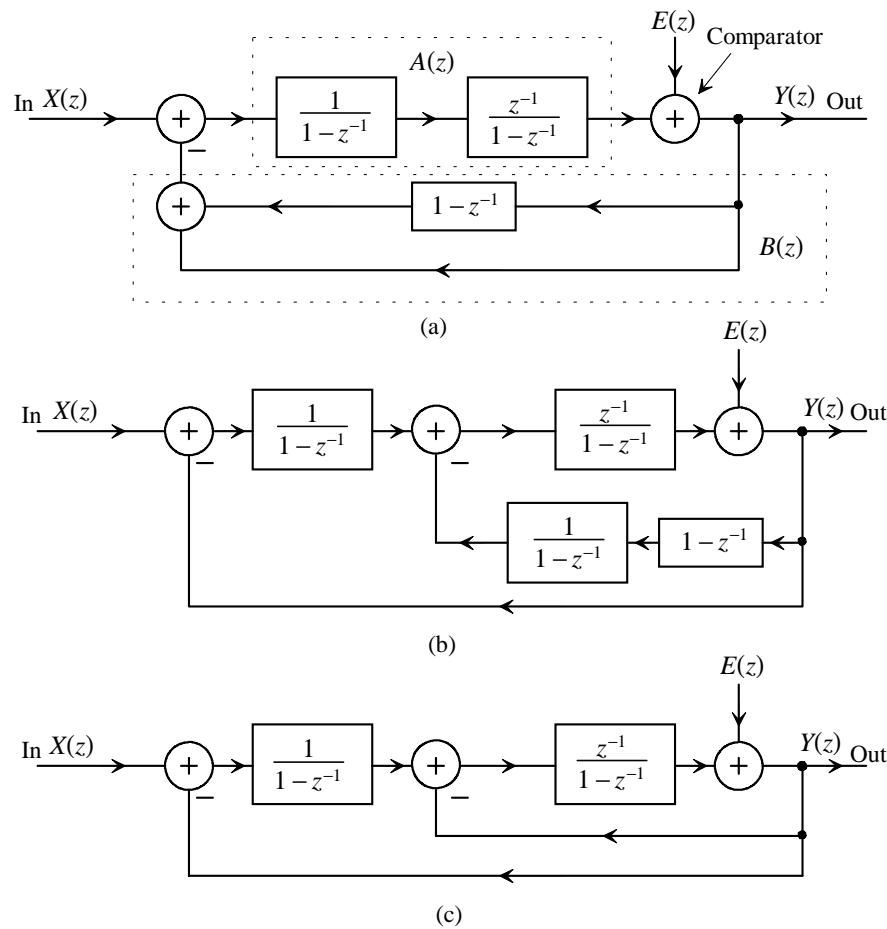


Figure 32.51 Block diagrams of second-order modulators.

The second-order (de) modulator topology of Fig. 32.51c can be used directly to implement a NS DAC (see Figs. 32.11 and 32.12). However, this topology doesn't lend itself directly to implementation using the DAI. The major concern, as discussed in the last section, is the op-amp's output going to the power-supply rails (integrator saturation). This is more of a concern in the second-order modulator since the output of the first integrator isn't connected directly to a comparator.

Figure 32.52a shows how we can add an integrator gain to the block diagram of Fig. 32.51c without changing the system's transfer function. Figure 32.52b shows pushing the gain, $1/G_I$, through the second summer so that it is directly preceding the second integrator. Notice how in Fig. 32.52b this (the second integrator's gain) is in series with the comparator's gain (not shown; see Fig. 32.36 and the associated discussion). This means we can arbitrarily change the second integrator's gain because of how the

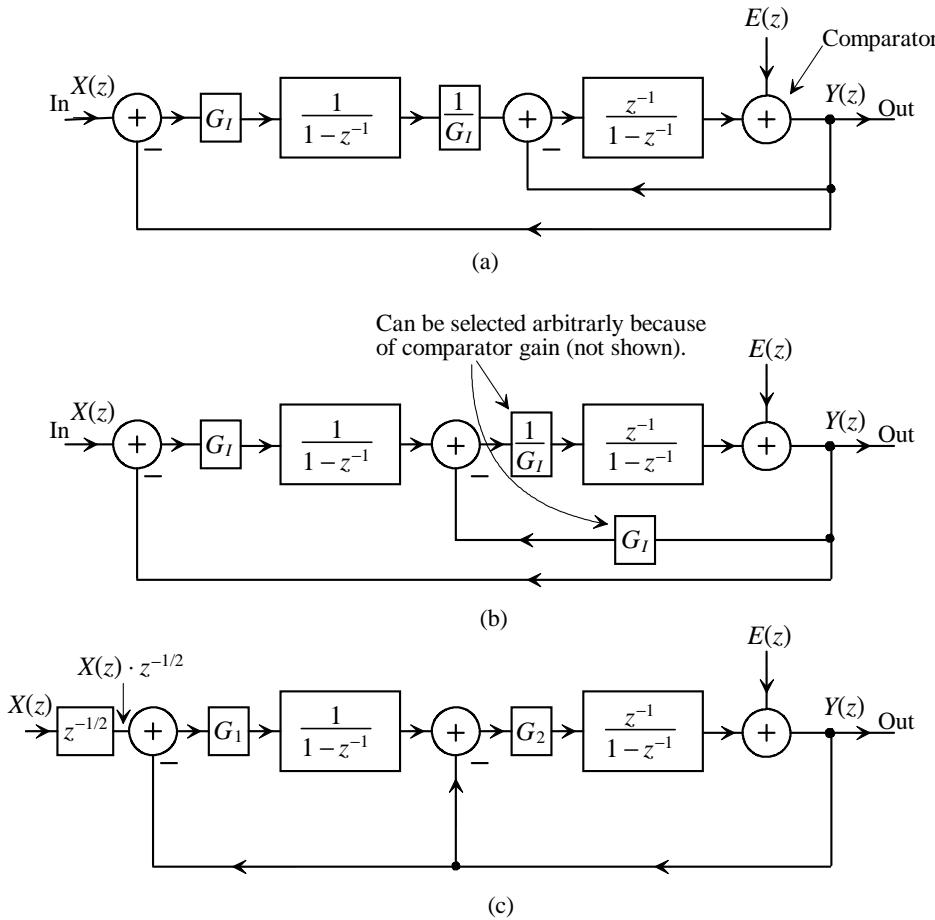


Figure 32.52 Block diagrams of second-order modulator introducing integrator gains.

comparator gain changes to force the loop gain to unity (see Fig. 32.38). Figure 32.52c shows the resulting configuration where the second integrator has a gain of G_2 and the first integrator has a gain of G_1 . Also notice how we have added a delay in series with the input signal. This delay was added to show how using a DAI results in an added delay in series with the input signal. The delay doesn't affect the magnitude of modulator's transfer function but rather indicates the input signal arrives half a clock cycle later.

Figure 32.53 shows the DAI implementation of the second-order modulator of Fig. 32.52c. Note how the output of the modulator is fed back and immediately passes through the first integrator and is applied to the second integrator (no delay as seen in Fig. 32.52c). This is a result of switching the phases of the clock signals in the first integrator. We should also see how the input signal sees an added half-clock cycle delay. Note that at this point it should be trivial to sketch the circuit implementation of the fully-differential, second-order modulator (see Fig. 32.44).

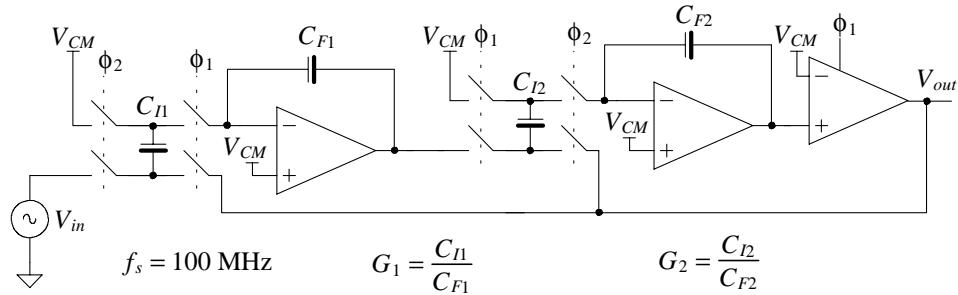


Figure 32.53 Implementation of the second-order modulator of Fig. 32.52c.

Integrator Gain

As we showed in Eq. (32.41) for the first-order modulator, the forward gain of a second-order modulator will be unity when the modulator is functioning properly. We now need to discuss how to select the integrator gains to avoid harmful integrator saturation. If noise and offsets were not a concern, as shown in Fig. 32.40 and the associated discussions, then we could make our integrator gains very small (ultimately limited by imperfections in the switches such as clock feedthrough and charge injection). In a practical modulator, integrator saturation (the integrator's gain going to zero) can also lead to modulator instability, as shown in Eq. (32.40), and the associated discussion.

Figure 32.54 shows the integrator outputs for the modulator of Fig. 32.53 if both integrator gains are set to 0.4. Notice how both outputs go outside the supply voltage range. If we replace the ideal op-amps in the simulation with transistor-based op-amps, the integrator outputs will saturate at some voltage within the supply range. This saturation can be thought of as noise and ultimately limits the data converter's SNR. Integrator saturation can be avoided by limiting the input signal range, designing with small integrator gain, and using op-amps that have a wide output swing.

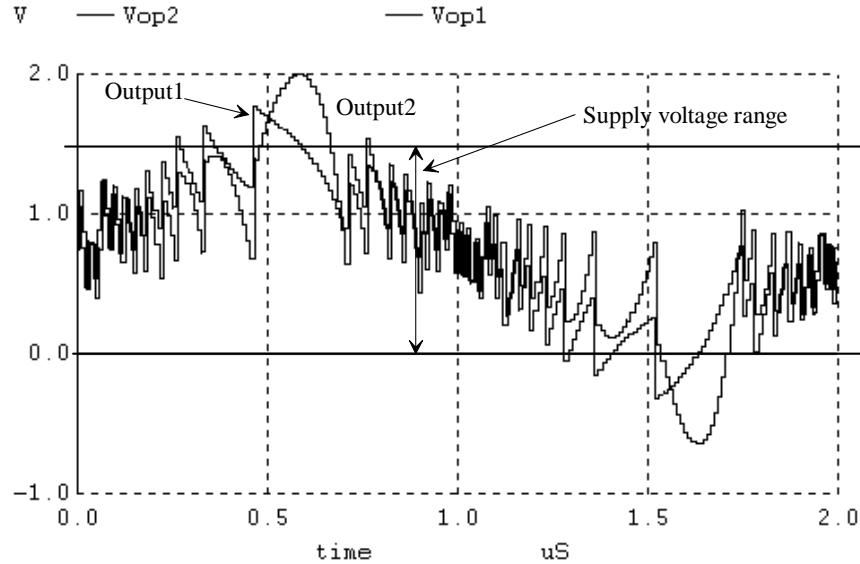


Figure 32.54 Showing integrator outputs using ideal components.

Example 32.12

Using SPICE simulations, show how an ideal second-order NS modulator can become unstable if the integrator gain is too low.

Because the second integrator is directly followed by a comparator, its gain is more tolerant to variations allowing it (the gain) to be made small. The first integrator's gain, however, is isolated from the comparator by the second integrator restricting its values. Figure 32.55a shows the (unstable) output of the modulator in Fig. 32.53 if $G_1 = 0.01$, while Fig. 32.55b shows the integrator outputs. ■

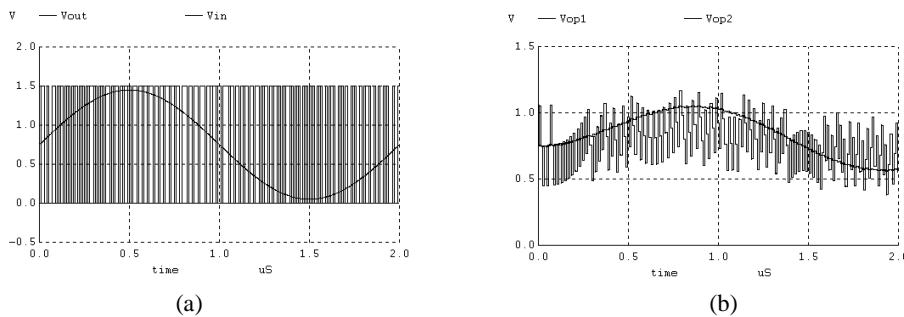


Figure 32.55 (a) Modulator output, and (b) integrator outputs if $G_1 = 0.01$

For a more quantitative view of how the gains in a second-order NS modulator affect performance, let's consider a couple of different topologies. Figure 32.56 shows the block diagram of the second-order NS modulator topology of Fig. 32.50, with an integrator gain coefficient, G_I , and a comparator gain, G_c , added. Deriving the transfer function of this linearized model with $G_F = G_I \cdot G_c$ results in

$$Y(z) = \frac{G_F \cdot z^{-1}}{1 + z^{-1} \cdot 2(G_F - 1) + z^{-2} \cdot (1 - G_F)} \cdot X(z) + \frac{(1 - z^{-1})^2}{1 + z^{-1} \cdot 2(G_F - 1) + z^{-2} \cdot (1 - G_F)} \cdot E(z) \quad (32.72)$$

The poles of this transfer function are located at

$$z_{p1,p2} = (1 - G_F) \pm \sqrt{(1 - G_F)^2 - (1 - G_F)} \quad (32.73)$$

We know that for the modulator to remain stable the poles must reside within the unit circle. This means that our values of forward gain are restricted to

$$0 \leq G_F \leq 1.333 \quad (32.74)$$

Again, if the modulator is functioning properly, $G_F = 1$ (because of the comparator's gain variation as seen in Fig. 32.28 and the associated discussion).

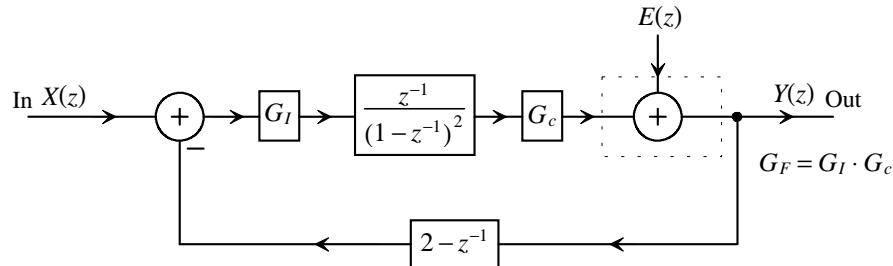


Figure 32.56 Block diagram of a second-order feedback modulator with gains.

We should make some observations at this point. Reviewing Eq. (32.40), we see that the allowable range of forward gain, in the first-order modulator, is larger than the allowable range in the second-order modulator. However, as long as the integrators don't saturate (G_I doesn't approach zero), stability for either modulator is easy to attain. An analysis of the stability of higher order modulators show that the range of allowable forward gains decreases with the order of the modulator. For example, a third-order modulator can have a forward gain of at most 1.15. Finally, notice that the input signal range is more restricted for the second-order modulator, in order to avoid integrator saturation, as seen in Fig. 32.54. We'll discuss methods to attain wider input signal range and more robust stability criteria by adjusting the feedback gains later in this section.

Notice that we are treating our modulator as a linear system even though it isn't linear; the comparator gain is a nonlinear variable. The linear approximation is useful to give an idea of the stability of the modulator under certain operating conditions. Generally, a DC input is applied to the modulator in the simulation, while lowpass filters are added to determine the average comparator gain, G_c . Figure 32.57 shows this schematically. Assuming we know G_i (the gain coefficient of the integrators), we can then look at the stability and forward gain of the modulator for varying DC input signal voltages.

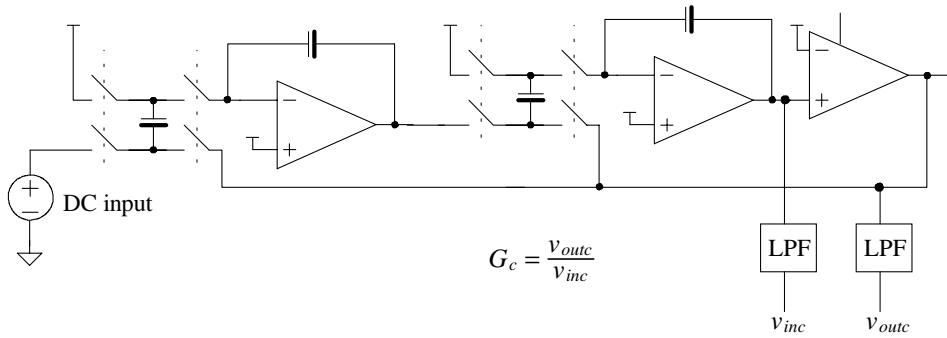


Figure 32.57 Simulating the gain of the comparator.

Next consider the more generic block diagram of the second-order NS modulator shown in Fig. 32.58. In a moment we'll discuss how to implement the feedback gain, G_3 , using the DAI. The transfer function of this topology can be written as

$$Y(z) = \frac{G_1 G_2 G_c \cdot z^{-1} X(z) + (1 - z^{-1})^2 \cdot E(z)}{1 + z^{-1} \cdot (G_1 G_2 G_c + G_2 G_3 G_c - 2) + z^{-2} \cdot (1 - G_2 G_3 G_c)} \quad (32.75)$$

Notice that if $G_1 = G_2 = G_3 = G_c = 1$ (where $G_1 G_2 G_c = G_F$), then this equation reduces to Eq. (32.62). The poles of this equation are located at

$$z_{p1,p2} = \frac{2 - G_1 G_2 G_c - G_2 G_3 G_c \pm \sqrt{(2 - G_1 G_2 G_c - G_2 G_3 G_c)^2 - 4(1 - G_2 G_3 G_c)}}{2} \quad (32.76)$$

When the modulator is functioning properly we require the (linearized) coefficient of the input, $X(z)$ in Eq. (32.75), to be unity

$$\left| \frac{G_1 G_2 G_c}{(z - z_{p1})(z - z_{p2})} \right| = 1 \quad (32.77)$$

Again, if we set $G_1 = G_2 = G_3 = 1$ (and $G_c = 1$), then the poles are located at DC, that is,

$$z_{p1,p2} = 0 \quad (32.78)$$

Equation (32.76) is useful to estimate the modulator's stability when scaling amplitudes by adjusting the integrator gain coefficients, G_1 , G_2 , and G_3 .

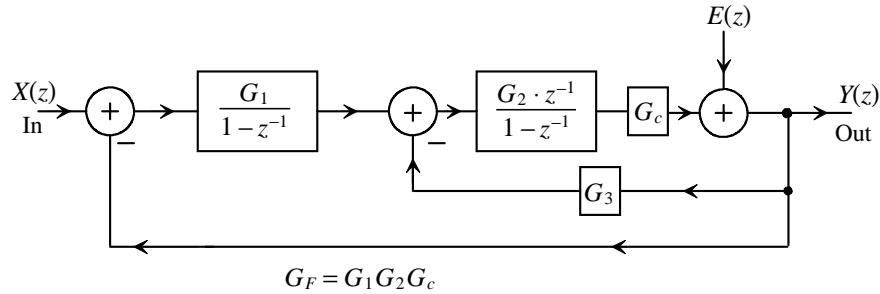


Figure 32.58 Generic block diagram of a second-order NS modulator.

Implementing Feedback Gains in the DAI

Consider the modified DAI shown in Fig. 32.59. Notice that if $C_{I2} = C_{I3}$, this topology reduces to the DAI shown in Fig. 31.78. Also note that some of the switches can be combined to simplify the circuitry. Assuming that the output is connected through the ϕ_2 switches (or that there are no switches connected to the output of the op-amp, see Eq. [31.136]) we can write the transfer function of the integrator as

$$V_{out}(z) = V_1(z) \cdot \frac{C_{I2}}{C_{F2}} \cdot \frac{z^{-1/2}}{1-z^{-1}} - V_2(z) \cdot \frac{C_{I3}}{C_{F2}} \cdot \frac{1}{1-z^{-1}} \quad (32.79)$$

The block diagram of this topology is shown in Fig. 32.60a. We want to implement a block diagram like the one shown in Fig. 32.60b. Because we have already defined

$$G_2 = \frac{C_{I2}}{C_{F2}} \quad (32.80)$$

we define our feedback gain, G_3 , as

$$G_3 = \frac{C_{I3}}{C_{F2}} \cdot \frac{1}{G_1} = \frac{C_{I3}}{C_{I2}} \quad (32.81)$$

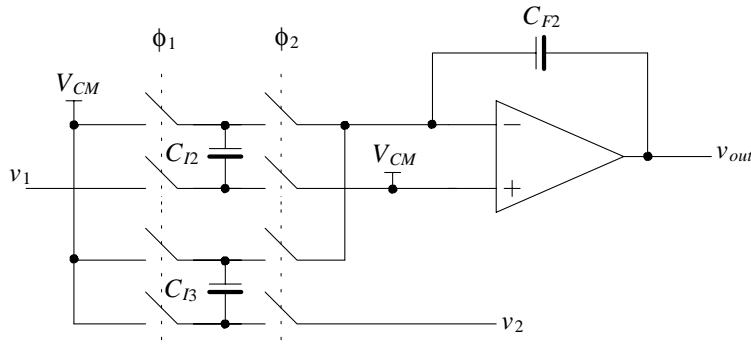


Figure 32.59 Adding an additional gain setting to our DAI.

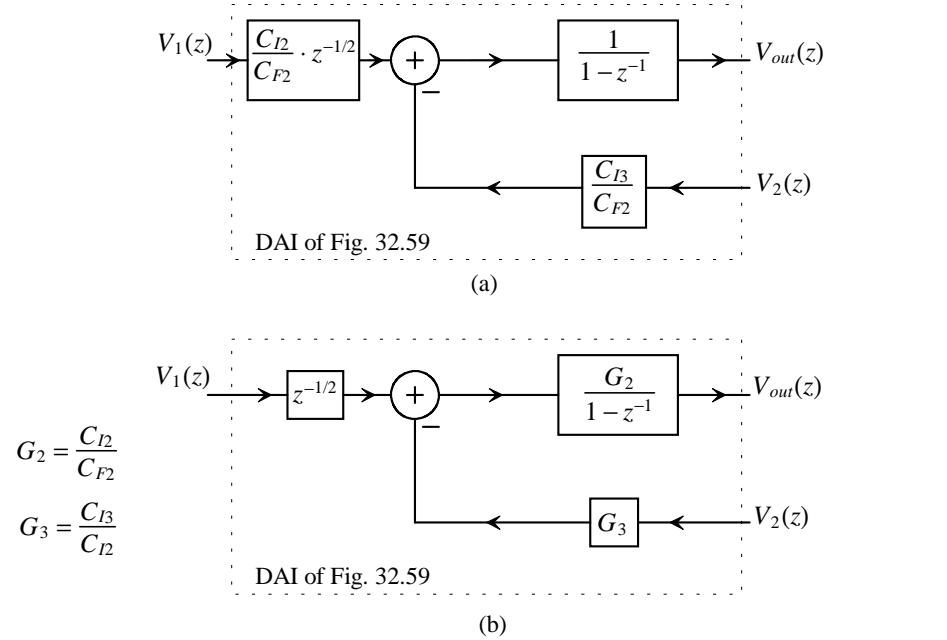


Figure 32.60 Block diagram of a DAI.

Example 32.13

Sketch the circuit implementation of a second-order NS modulator based on the topology of Fig. 32.58, where $G_1 = G_2 = G_3 = 0.4$. Comment on the stability of the resulting configuration. Simulate the design and show the integrator output swing.

The block diagram of the modulator is shown in Fig. 32.61. We could dissect Eq. (32.76) at this point to determine the transient properties of the modulator. However, before discussing the transient characteristics of the modulator, let's look at the integrator output swing.

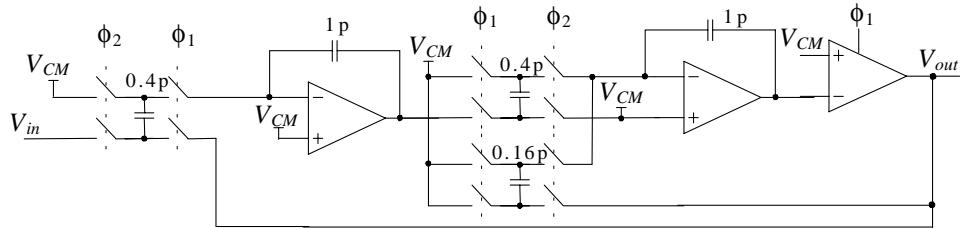


Figure 32.61 Implementation of a second-order modulator with feedback gain.

Figure 32.62 shows the output swing of the integrators. This figure should be compared with Fig. 32.54. The output of the first integrator now falls within the power supply range. The output of the second integrator is reduced but still exceeds the power supply range. This, as discussed earlier, has less impact on performance in the actual transistor-based modulator because the integrator is followed by a comparator.

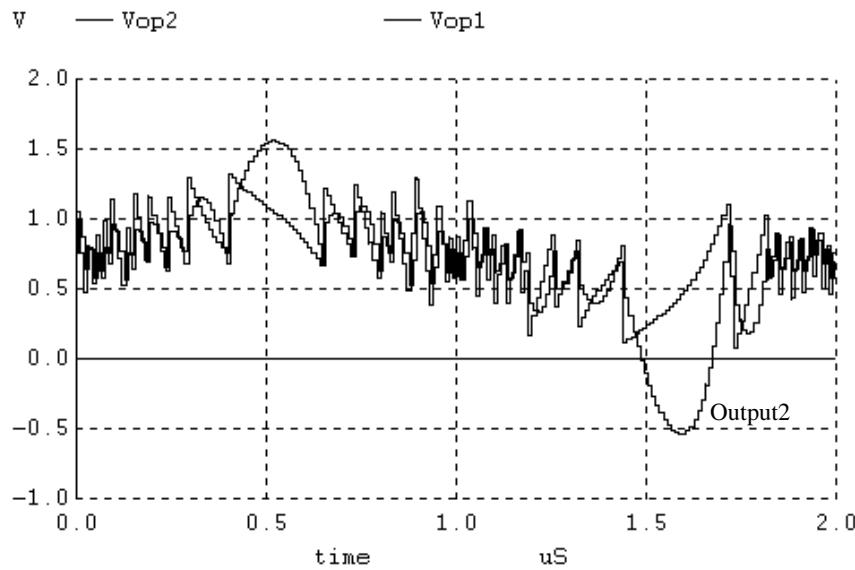


Figure 32.62 Integrator output signals in the modulator shown in Fig. 32.61.

Let's attempt to get an idea for the stability of the modulator by adding LPFs, as seen in Fig. 32.57, to the simulation (with a DC input) to measure G_c . Figure 32.63 shows how we will implement the LPFs. The voltage-controlled voltage source is used to keep from loading the modulator with the RC circuit when it is added into the general simulation. In our ideal modulator shown in Fig. 32.61 both the comparator output and integrator outputs are ideal voltage sources, so we don't need the isolation (and therefore we can add the RC LPF directly into the simulation).

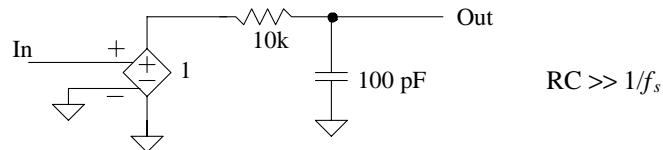


Figure 32.63 SPICE implementation of a LPF for determining comparator gain.

Figure 32.64 shows the comparator input and output, after lowpass filtering, for the modulator of Fig. 32.61 when the input signal is 0.1 V (DC). Longer simulation times reveal the average comparator input is 0.4 V. The resulting comparator gain is then only 0.25. Using Eq. (32.76) to calculate the location of the poles results in $z_{p1,p2} = 0.96 \pm j \cdot 0.195$. These poles are very close to the unit circle. Small shifts in the DAI gains can result in an unstable modulator. Increasing the input signal amplitude makes the modulator more stable. Increasing G_3 also increases the modulator's stability.

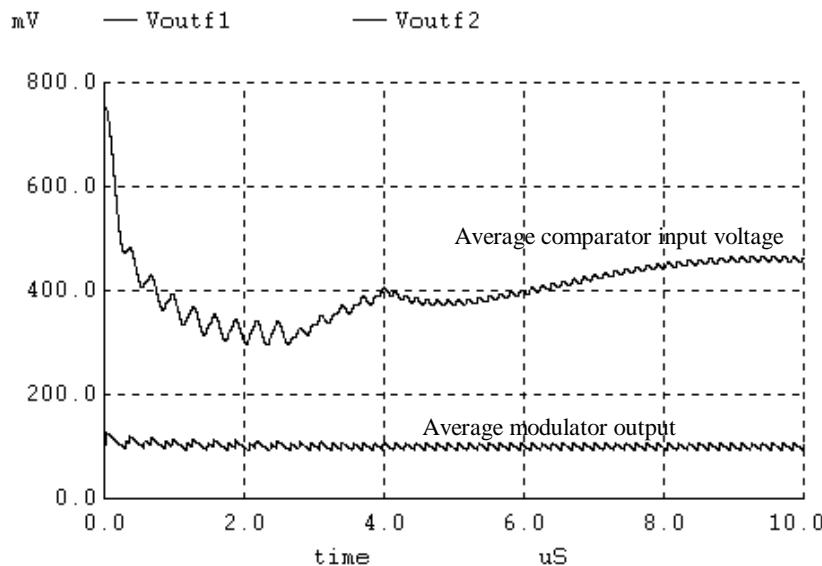


Figure 32.64 Average comparator input and output when using the modulator of Fig. 32.61 with an input signal of 0.1 V.

The simulation that generated Fig. 32.64 can be very useful in understanding basic second-order modulator's stability criteria. Changing the simulation variables and looking at the resulting simulation outputs can be very instructional. Note that increasing the simulation time in the netlist that generated Fig. 32.64 would reveal that the comparator input actually has small amplitude oscillations. Also note how Fig. 32.62 shows the output of the second integrator going way outside the power supply limits when transitioning negative (going well below 0 V) while staying bounded to the power-supply rail when transitioning positive (above 1.5 V). This is related to the stability of the modulator being a function of the input voltage. ■

Using Two Delaying Integrators to Implement the Second-Order Modulator

Consider the second-order modulator topology shown in Fig. 32.65. This topology can be implemented using the circuits of Figs. 32.53 or 32.61 by simply switching the phases of

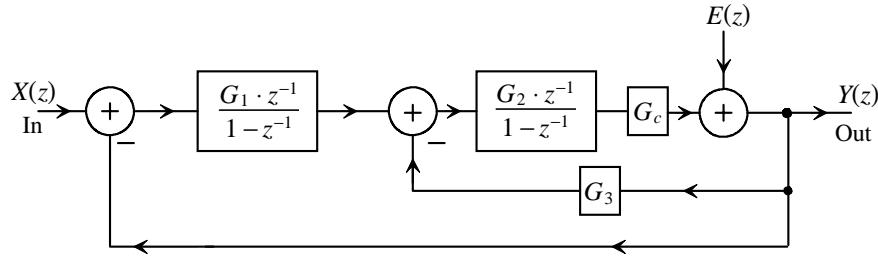


Figure 32.65 Second-order NS modulator using two delaying integrators.

the clocks in the first integrator (by making both integrators delaying). The transfer function of this topology is

$$Y(z) = \frac{G_1 G_2 G_c \cdot z^{-2} X(z) + (1 - z^{-1})^2 \cdot E(z)}{1 + z^{-1} \cdot (G_2 G_3 G_c - 2) + z^{-2} \cdot (1 - G_2 G_3 G_c + G_1 G_2 G_c)} \quad (32.82)$$

The poles are located at

$$z_{p1,p2} = \frac{2 - G_2 G_3 G_c \pm \sqrt{(2 - G_2 G_3 G_c)^2 - 4(1 - G_2 G_3 G_c + G_1 G_2 G_c)}}{2} \quad (32.83)$$

This equation should be compared to Eq. (32.76). Remembering that for a stable modulator the poles must be inside the unit circle, we see that using two delaying integrators will not result in a modulator that has as robust stability criteria as the general implementation of Fig. 32.58. Figure 32.66 shows the implementation of a second-order NS modulator using two delaying integrators. One advantage of this topology over the topology of Fig. 32.58 is the reduced slew-rate requirements of the op-amps since neither op-amp in Fig. 32.66 has to drive both the feedback capacitance and the switched input capacitance of the next stage during the same clock phase.

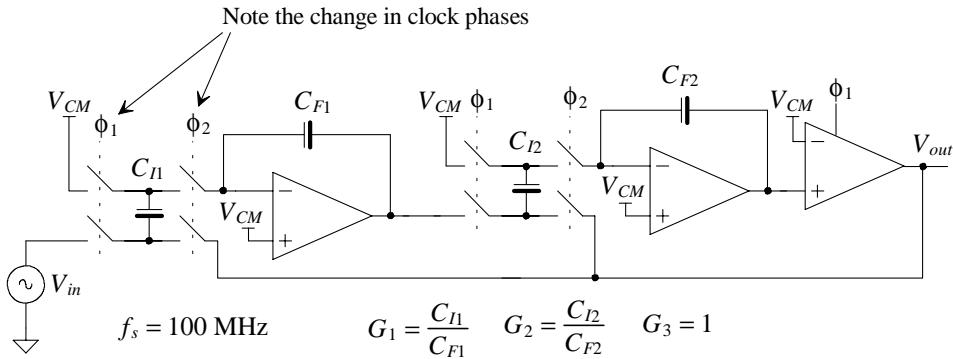


Figure 32.66 Implementation of a delaying second-order NS modulator.

Selecting Modulator (Integrator) Gains

Before leaving this section, let's discuss the general selection of modulator gains. In general, for good stability, the inner loop feedback gain, G_3 , should be made as large as possible. For general design, set $G_3 = 1$. This simplifies the design of the modulator circuitry and provides good flexibility when selecting the values of G_1 and G_2 . If $G_3 = 1$, then Eq. (32.76) may be rewritten to show the location of the poles as

$$z_{p1,p2} = \frac{2 - G_1 G_2 G_c - G_2 G_c \pm \sqrt{(2 - G_1 G_2 G_c - G_2 G_c)^2 - 4(1 - G_2 G_c)}}{2} \quad (32.84)$$

Keeping in mind that the reason we are not setting all gains to one is to avoid integrator saturation, we can look at Eq. (32.84) as a guide to determine how we can reduce G_1 and G_2 . Since G_2 is directly followed by the comparator, we can set its gain to 0.4 as discussed earlier. Practically then, we can reduce the value of G_1 to a very small number and still have a stable modulator (see Ex. 32.12). At the same time using small G_1 avoids integrator saturation. The practical problem with small G_1 , as discussed earlier, is the increase in the input-referred noise. Again trade-offs must be made for given design criteria. Figure 32.67 shows the integrator outputs for the modulator of Fig. 32.58 when $G_1 = 0.2$, $G_2 = 0.4$, and $G_3 = 1$. Note how, when compared to Figs. 32.54 and 32.62, the outputs are very well behaved. We don't have the abnormal transitions above the power-supply rails indicating that the modulator stability is becoming marginal with input signal values close to the power-supply rails.

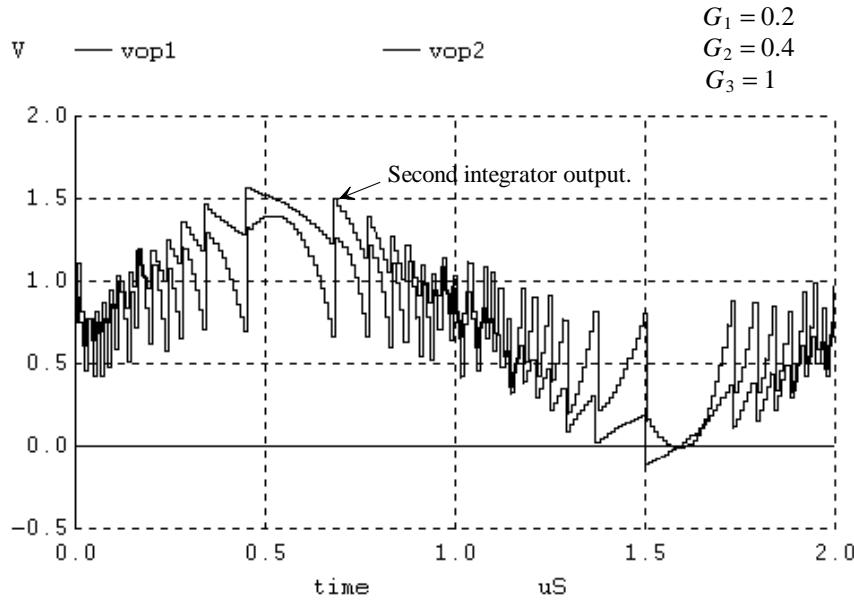


Figure 32.67 Integrator outputs for a modulator with first integrator gain of 0.2.

Understanding Modulator SNR

Figure 32.68 shows the output spectrum of a first-order NS modulator clocked at 100 MHz. This spectrum should be compared to the spectrum of the second-order NS modulator shown in Fig. 32.69. Note how the spectrum of the first-order modulator appears to contain more tones in the base spectrum of interest than the spectrum of Fig. 32.69. As discussed earlier, unwanted tones are less of a problem in second-order modulators.

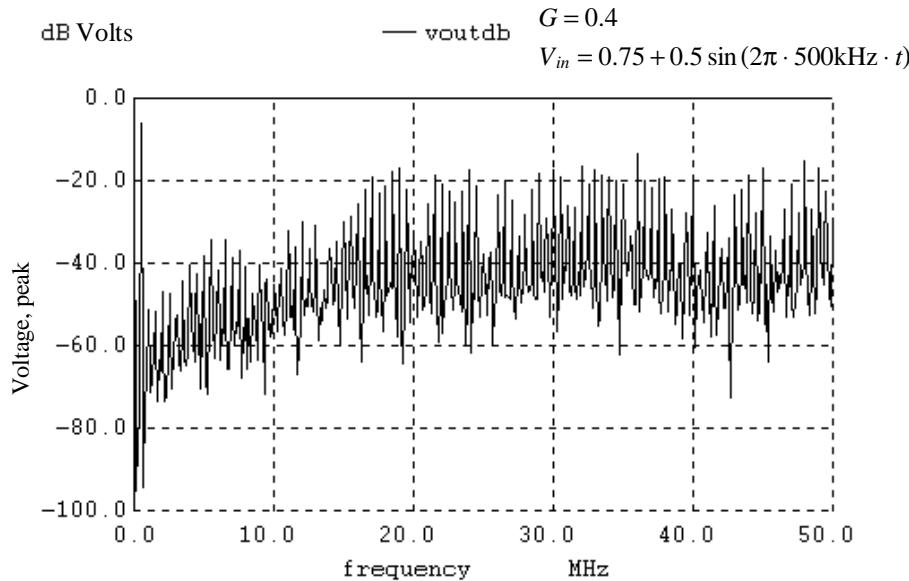


Figure 32.68 Output spectrum of a first-order modulator.

We know that for the modulator to be useful its output must be passed through a lowpass filter to remove the modulation noise. In simulations we can approximate a lowpass filter with a bandwidth B by limiting the spectral analysis range. To estimate the SNR from the simulations of Figs. 32.68 or 32.69, with $K = 16$, we perform the spectral analysis up to 50 MHz/16 or 3.125 MHz. The quantization noise plus distortion is calculated as discussed in the last chapter and shown in the SPICE netlists.

To demonstrate the calculation of a modulator's SNR let's use the second-order modulator simulation used to generate Fig. 32.69. Using Eq. (32.65) with $V_{LSB} = 1.5$ V and $K = 16$ results in an RMS quantization noise of 1.86 mV. The SNR_{ideal} is calculated, using Eq. (32.66) as 55 dB. However, Eq. (32.66) was derived assuming Eq. (32.9) was valid. For the 1-bit ADC/DAC it is not. For the 1-bit DAC/ADC V_{LSB} is twice the value given by Eq. (32.66) or $V_{REF+} - V_{REF-}$. The doubling in V_{LSB} results in a subtraction of 6 dB from Eq. (32.66). The SNR_{ideal} is 49 dB. To discuss this further consider a sinewave with a peak-to-peak amplitude of V_{LSB} ($= 1.5$ here). We can write the SNR of the modulator as

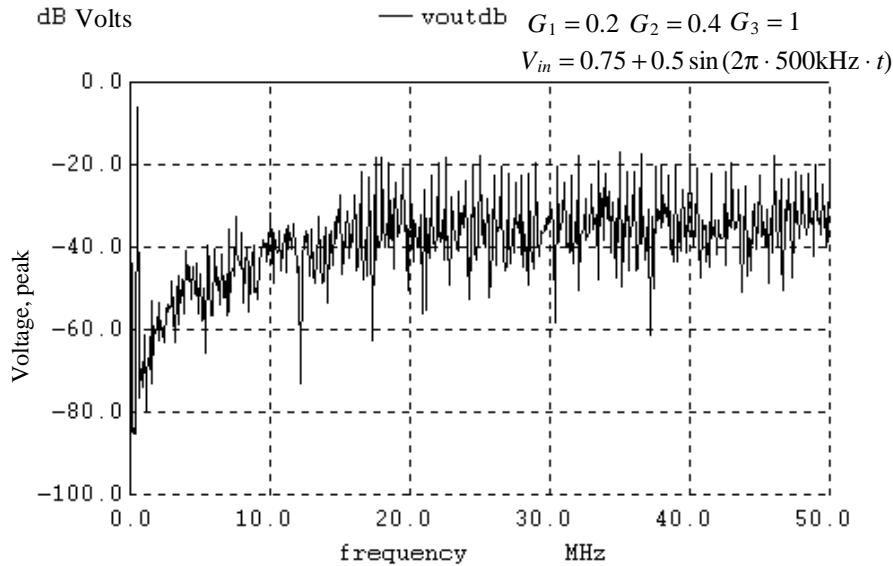


Figure 32.69 Output spectrum of a second-order modulator.

$$\text{SNR}_{\text{ideal}} = 20 \log \left[\left([1.5/2]/\sqrt{2} \right) / (1.86 \text{ mV}) \right] = 49 \text{ dB} \quad (32.85)$$

Modifying Eq. (32.66) for the 1-bit case and the increase in V_{LSB} results in

$$\text{SNR}_{\text{ideal}} = 50 \cdot \log K - 8.3 = 49 \text{ dB} \quad (32.86)$$

In practice the SNR (SNDR) is considerably worse than the ideal value. This variation comes from the fact that our assumed quantization noise spectrum isn't white and the modulator output can contain unwanted tones at multiples of the input sinewave signal frequency. Figure 32.70 shows the output of the second-order modulator of Fig. 32.69 when we limit the spectral analysis to B ($= 3.125$ MHz). Calculating the SNR using Eq. (32.85) with a peak input sinewave amplitude of 0.5 V gives 45.6 dB. Simulations using ideal components, however, give a 37 dB SNDR. Zeroing out the tones (see the commands in the netlist where we have already zeroed out the DC term and the fundamental at 500 kHz) will obviously increase the SNR. Note also that we didn't use a full-scale input sinusoid (peak amplitude of 0.75 V). Full-scale inputs inherently result in a reduction in SNR because our modulator has less output signal range to average over. For example, an input signal approaching the supply rails causes the output of the modulator to remain high most of the time, while a mid-scale signal results in more modulator output variation allowing better averaging. The reduction in SNDR with input signal amplitude was shown back in Fig. 31.5. Finally, note that by using a high-frequency input signal, which has the same effect as an added dither signal, the output modulation noise becomes more random and the SNR increases. To illustrate this we could resimulate the netlist used for Fig. 32.70 with an input frequency of 5 MHz (outside our signal band of interest).

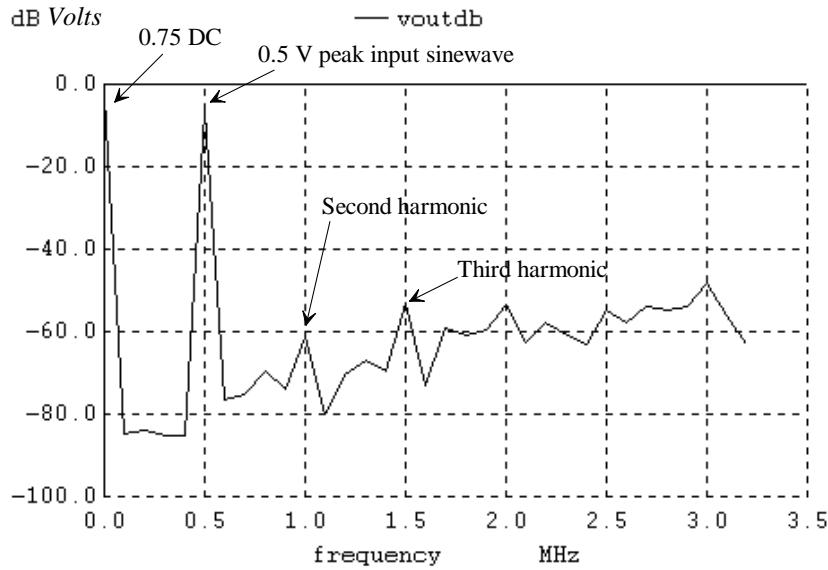


Figure 32.70 Same as Fig. 32.69 but with limited spectral range.

32.2 Noise-Shaping Topologies

The last section presented the fundamentals of NS data converters. It's important to understand this fundamental material before proceeding with the topics presented in this section.

In this section we cover (1) higher order NS modulators, (2) NS modulators using multibit ADCs and DACs (multibit modulators), (3) cascaded modulators (higher-order modulators built with a cascade of first- and/or second-order modulators), and (4) bandpass modulators (modulators that perform data conversion over a band of frequencies that doesn't include DC).

32.2.1 Higher-Order Modulators

We can take the theory developed for our first- and second-order modulators in the last section and generalize it for an M^{th} -order modulator (a modulator having M integrators and M feedback loops). Rewriting Eqs. (32.11) and (32.64) for the general M^{th} -order modulator results in

$$|NTF(f)| \cdot |V_{Qe}(f)| = \frac{V_{LSB}}{\sqrt{12f_s}} \cdot \left[2 \sin \pi \frac{f}{f_s} \right]^M \quad (32.87)$$

The RMS noise in a bandwidth, B , can be written, see Eqs. (32.25) and (32.65), as

$$V_{Qe,RMS} = \frac{V_{LSB}}{\sqrt{12}} \cdot \frac{\pi^M}{\sqrt{2M+1}} \cdot \frac{1}{K^{M+1/2}} \quad (32.88)$$

The ideal increase in the SNR can be written as

$$\text{SNR}_{\text{ideal}} = 6.02N + 1.76 - 20 \log \left[\frac{\pi^M}{\sqrt{2M+1}} \right] + [20M+10] \cdot \log K \quad (32.89)$$

or

$$\text{SNR}_{\text{ideal}} = 6.02(N+N_{\text{inc}}) + 1.76 \quad (32.90)$$

The increase in resolution, N_{inc} , is given by

$$N_{\text{inc}} = \frac{1}{6.02} \left[(20M+10) \cdot \log K - 20 \log \left(\frac{\pi^M}{\sqrt{2M+1}} \right) \right] \quad (32.91)$$

This equation shows that for every doubling in the oversampling ratio, K , the resolution increases by $M + 0.5$ bits. In practice, as we have seen already, this equation results in an overestimate for the increase in resolution because the quantization noise is not truly white (because of the 1-bit ADC) and the modulation noise contains unwanted spectral tones (as seen in Fig. 32.70).

M^{th} -Order Modulator Topology

Reviewing the general NS modulator topology of Fig. 32.50 we want to determine the forward transfer function, $A(z)$, and the feedback transfer function, $B(z)$, for an M^{th} -order NS modulator. The transfer function of a general M^{th} -order modulator is

$$Y(z) = X(z) \cdot (z^{-1}) + E(z) \cdot (1 - z^{-1})^M \quad (32.92)$$

Using this equation together with Eq. (32.67) results in a forward modulator transfer function of

$$A(z) = \frac{z^{-1}}{(1 - z^{-1})^M} \quad (32.93)$$

and a feedback filter transfer function of

$$B(z) = \frac{1 - (1 - z^{-1})^M}{z^{-1}} \quad (32.94)$$

The block diagram of an M^{th} -order NS modulator is shown in Fig. 32.71.

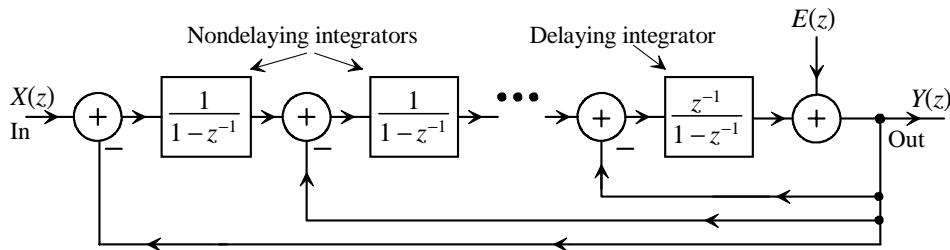


Figure 32.71 Generic block diagram of an M^{th} -order NS modulator.

Decimating the Output of an M^{th} -Order NS Modulator

Let's revisit the derivation of Eq. (32.20). This equation states that the number of sinc stages, L , used in cascade, for near optimum removal of the modulation noise, is one more than the order of the modulator ($L = M + 1$). Rewriting Eq. (32.22)

$$V_{Qe,RMS}^2 = 2 \int_0^{f_s/2} |NTF(f)|^2 \cdot |V_{Qe}(f)|^2 \cdot |H(f)|^2 \cdot df \quad (32.95)$$

where the decimation filter's transfer function is given by

$$|H(f)|^2 = \left[\frac{1}{K} \cdot \frac{\sin\left(K\pi\frac{f}{f_s}\right)}{\sin\left(\pi\frac{f}{f_s}\right)} \right]^{2(M+1)} \quad (32.96)$$

The mean-squared quantization noise is calculated by evaluating

$$V_{Qe,RMS}^2 = 2 \cdot \overbrace{\frac{V_{LSB}^2}{12f_s}}^{|V_{Qe}(f)|^2} \cdot \int_0^{f_s/2} \overbrace{\left[2 \sin \pi \frac{f}{f_s} \right]^{2M}}^{|NTF(f)|^2} \cdot \left[\frac{1}{K} \cdot \frac{\sin\left(K\pi\frac{f}{f_s}\right)}{\sin\left(\pi\frac{f}{f_s}\right)} \right]^{2(M+1)} \cdot df \quad (32.97)$$

or

$$V_{Qe,RMS}^2 = 2 \cdot \frac{V_{LSB}^2}{12f_s} \cdot 2^{2M} \cdot \left[\frac{1}{K} \right]^{2(M+1)} \cdot \int_0^{f_s/2} \frac{\sin^{2(M+1)}\left(K\pi\frac{f}{f_s}\right)}{\sin^{2M}\left(\pi\frac{f}{f_s}\right)} \cdot df \quad (32.98)$$

If we let $\theta = \pi\frac{f}{f_s}$, then we get

$$V_{Qe,RMS}^2 = \frac{V_{LSB}^2}{12f_s} \cdot \left[\frac{2}{K} \right]^{2(M+1)} \cdot \frac{f_s}{\pi} \cdot \overbrace{\int_0^{\frac{\pi}{2}} \frac{\sin^{2(M+1)}(K\theta)}{\sin^{2M}\theta} \cdot d\theta}^{= \frac{K}{2}\pi \prod_{m=1}^M \frac{2m-1}{2m}} \quad (32.99)$$

Finally, the RMS quantization noise associated with an M^{th} -order modulator followed by an $M + 1$ ($= L$) sinc averaging filter is

$$V_{Qe,RMS} = \frac{V_{LSB}}{\sqrt{12}} \cdot \left[\frac{2}{K} \right]^{M+1/2} \cdot \prod_{m=1}^M \frac{2m-1}{2m} \quad (32.100)$$

The change in SNR, when using the sinc averaging filter decimator instead of the ideal filter with bandwidth, B , is given by looking at the ratio of Eq. (32.88) to Eq. (32.100)

$$\text{Increase in SNR} = -20 \log \left[2^{M+1/2} \cdot \prod_{m=1}^M \frac{2m-1}{2m} \cdot \frac{\sqrt{2M+1}}{\pi^M} \right] \quad (32.101)$$

For first-, second-, and third-order modulators, the difference in the SNRs is 2.16, 6.35, and 10.39 dB, respectively. This shows that using a sinc averager, theoretically, increases the SNR if we neglect the decrease in the desired signal amplitude because of the droop, Figs. 31.43 or 31.46. To avoid the droop, as discussed earlier, the desired signal content is often limited to frequencies well below $f/2K$ ($= B$). When the droop (reduction in the desired signal amplitude) is taken under consideration, the SNR, when using the sinc averaging filter, is worse than the ideal filter with bandwidth B .

Implementing Higher Order, Single-Stage Modulators

The single-stage, higher order modulator of Fig. 32.71 can be difficult to implement directly. It is impossible to implement a higher order modulator, when using DAIs, where all but the last integrator are nondelaying. However, as we saw with the second-order modulator using two delaying integrators in Fig. 32.65 and Eqs. (32.82) and (32.83), the stability criteria of a modulator using only delaying integrators is poorer than the criteria of the topology shown in Fig. 32.71 (where only the last integrator is delaying). While we can help the situation by staggering delaying and nondelaying integrators in a modulator, the point is that implementing a higher order modulator without modifying our basic NS topology will result in an unstable circuit. Intuitively, we can understand this by noting that if the modulator's forward gain is too high and the delay through the forward path is too long (because of the large number of integrators), the signal fed back may add to the input signal instead of subtracting from it.

To help with the stability of a higher order modulator a topology that feeds the input signal forward into additional points in the modulator (thereby reducing the forward gain and delay) and feeds the output signal back as discussed earlier (allowing scaling of amplitudes) is needed. Towards this goal, consider the modified NS topology for higher order modulators shown in Fig. 32.72. The forward and feedback transfer functions can be written as

$$A(z) = \frac{a_1 \cdot z^{-M}}{(1-z^{-1})^M} + \frac{a_2 \cdot z^{-(M-1)}}{(1-z^{-1})^{M-1}} + \frac{a_3 \cdot z^{-(M-2)}}{(1-z^{-1})^{M-2}} + \dots + \frac{a_M \cdot z^{-1}}{1-z^{-1}} = \sum_{i=1}^M a_i \cdot \left(\frac{z^{-1}}{1-z^{-1}} \right)^{M-i+1} \quad (32.102)$$

or

$$A(z) = (z-1)^{-M} \cdot [a_1 + a_2(z-1)^1 + a_3(z-1)^2 + \dots + a_M(z-1)^{M-1}] \quad (32.103)$$

and

$$-A(z)B(z) = \frac{b_1 \cdot z^{-M}}{(1-z^{-1})^M} + \frac{b_2 \cdot z^{-(M-1)}}{(1-z^{-1})^{M-1}} + \frac{b_3 \cdot z^{-(M-2)}}{(1-z^{-1})^{M-2}} + \dots + \frac{b_M \cdot z^{-1}}{1-z^{-1}} = \sum_{i=1}^M b_i \cdot \left[\frac{1}{z-1} \right]^{M-i+1} \quad (32.104)$$

or

$$-A(z)B(z) = (z-1)^{-M} \cdot [b_1 + b_2(z-1)^1 + b_3(z-1)^2 + \dots + b_M(z-1)^{M-1}] \quad (32.105)$$

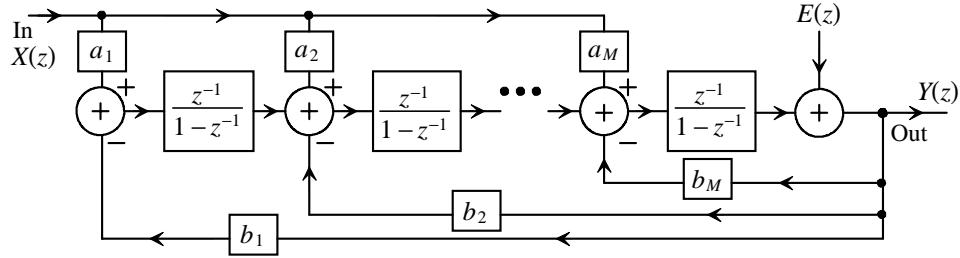


Figure 32.72 Block diagram of a modified M^{th} -order NS modulator.

Before going any further, let's explain what we are trying to do with the modified, higher-order, NS topology of Fig. 32.72. We know that the $NTF(z)$, for a general modulator, is of the form $(1 - z^{-1})^M$ with a shape seen in Fig. 32.73. At high frequencies the modulation noise will get very large. At $f_s/4$, for example, the magnitude of the noise transfer function, $|NTF(f)|$, is $(\sqrt{2})^M$ (see Fig. 31.51). For the modified NS modulator we will try to reduce the modulation noise at higher frequencies by changing the shape of the $NTF(z)$. Our modified $NTF(z)$ will be of the form

$$NTF(z) = LPF(z) \cdot (1 - z^{-1})^M = LPF(z) \cdot \left(\frac{z-1}{z}\right)^M = HPF(z) \quad (32.106)$$

where $LPF(z)$ [$HPF(z)$] is a lowpass [highpass] filter implemented with the feedback coefficients b_x . The goal is to flatten out the higher frequency modulation noise (keep the noise from getting too large) thereby reducing the $NTF(f)$ at high frequencies and keeping the modulator stable. One drawback of using this technique is that the signal no longer sees just a delay in its transfer function but rather it sees the lowpass response. The modified STF will be of the form

$$STF(z) = NTF(z) \cdot A(z) = LPF(z) \cdot \sum_{i=1}^M a_i \cdot (z-1)^{i-1} \quad (32.107)$$

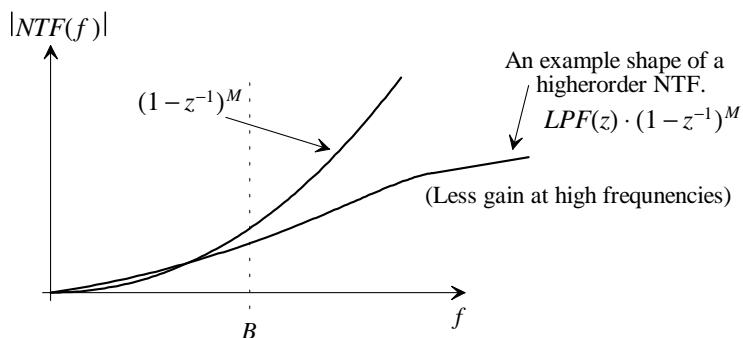


Figure 32.73 Showing the change in the NTF in a higher order modulator.

so that the feed forward coefficients, a_x , can be used to help make the $STF(f)$ constant over the region of interest (the STF can be made to have an overall lowpass response). The NTF is given by

$$NTF(z) = \frac{1}{1 + A(z)B(z)} \quad (32.108)$$

or

$$NTF(z) = \frac{1}{1 - (z-1)^{-M} \cdot \sum_{i=1}^M b_i \cdot (z-1)^{i-1}} = HPF(z) \quad (32.109)$$

The coefficients, b_x , are selected for a highpass response. Note also that our coefficients are positive since the feedback paths, as seen in Fig. 32.72, are subtracting. The design of the modulator at this point is to determine the feed-forward and feedback coefficients using basic digital-signal processing filter design (and, to keep the algebra simple, a computer program of some sort), then to simulate the design to see if it exceeds specifications. One challenge, among others, is to meet a given SNR without causing harmful integrator saturation.

Other topologies have been developed to implement higher-order NS modulators. The reader is referred to Chs. 4 and 5 of [2] for further information.

32.2.2 Multi-Bit Modulators

Throughout this chapter we have assumed $N = 1$; that is, we have used a comparator for our quantizer in the forward path of our NS modulator. The main advantage of single-bit modulators, as discussed earlier, is the inherent linearity of the 1-bit feedback DAC. Feedback DAC linearity is important because the output of the DAC is directly subtracted from the input signal. Any distortion or nonlinearity (or noise) in the output of the DAC will directly affect the modulator's performance and, ultimately, limit the modulator's SNR . The benefits of using a multibit ($N > 1$) quantizer in a NS modulator are increased SNR (see Eq. 32.89), better stability (the modulator behaves closer to the linearized theory developed in this chapter), fewer spectral tones, and simpler digital-decimation filter. The drawbacks of using multibit topologies, are the increase in ADC complexity (the ADC must be a flash converter) and the need for the DAC to be accurate to the final accuracy of the modulator. The ADC errors, like gain errors in the integrators, are less important since they are in the forward, high-gain path of the modulator.

Simulating a Multibit NS Modulator Using SPICE

Figure 32.74 shows a circuit-level implementation of a first-order, multibit, NS modulator using a 4-bit ADC and DAC. Figure 32.75 shows the SPICE simulation outputs of this modulator in the time and frequency domains with the same input sinewave used in generating Fig. 32.68. Comparing Fig. 32.75 to Fig. 32.68 the decrease in modulation noise is obvious. Note that (a) of the figure shows both the input to the modulator and the output of the ideal DAC while (b) is the DAC's output spectrum. Looking at the output of the DAC avoids the need for a Fourier transform on the modulator's output digital data.

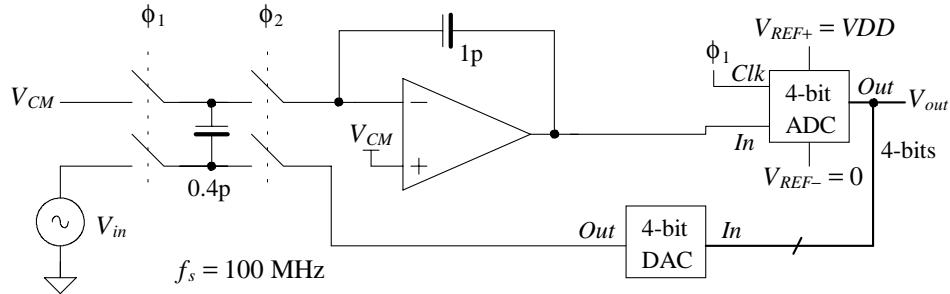


Figure 32.74 Circuit implementation of a first-order multi-bit NS modulator.

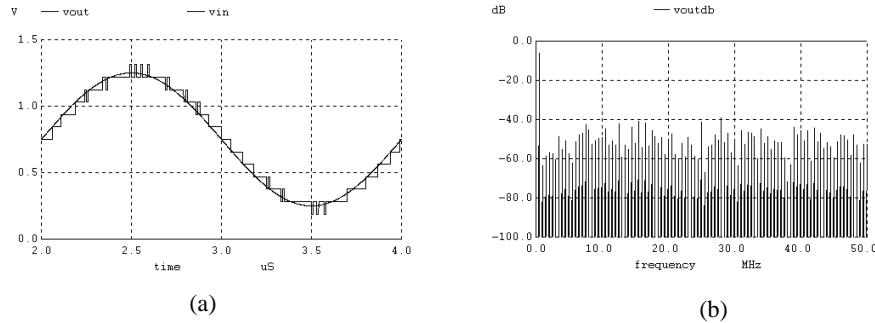


Figure 32.75 Output of the (a) multibit modulator and (b) its spectrum.

Most of the design effort, when developing multi-bit modulators, goes into the design of the feedback DAC. Because it is nearly impossible to design highly accurate DACs without trimming, or some sort of error correction, methods have been developed that attempt to randomize DAC errors. If the errors appear as a random variable, they may appear as white noise in the output spectrum and not affect the SNR of the data converter.

Figure 32.76 shows one possible implementation of a DAC that utilizes resistive unit elements. While this figure is busy, let's attempt to explain how the DAC functions. The DAC is based on unit-element (equal value) resistors. In one case we connect VDD to one corner of the resistor cube and ground to the opposite corner (assuming $V_{REF+} = VDD$ and $V_{REF-} = 0$). There exist two voltage dividers along each of the sides of the resistor square. The output of the DAC can change from zero, to $(1/8)VDD$, to $(2/8)VDD$, ... up to $(8/8)VDD$. Depending on the output of the decoder, one tap from each side is fed to the analog output. Because there are two sides, the outputs from each side are combined and effectively averaged.

The purpose of the counter is to vary the connections of VDD and ground around the outside of the resistive divider to randomize variations in the output voltage due to resistor mismatch. To understand this in more detail consider a constant DAC output

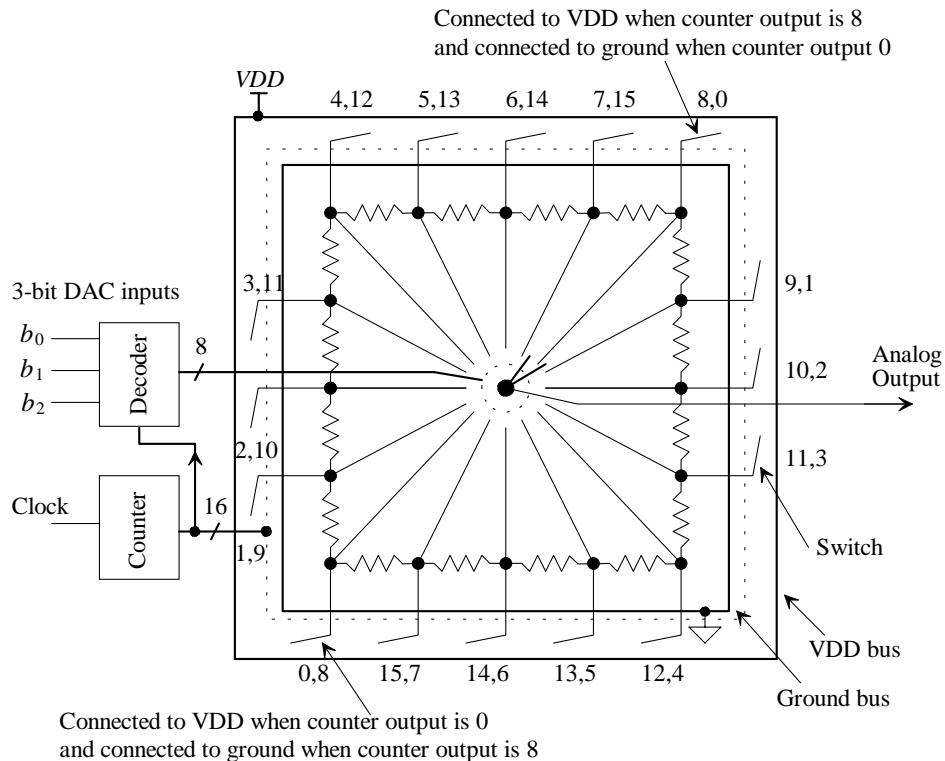


Figure 32.76 Implementation of a DAC for use in a multibit NS modulator.

voltage of $VDD/2$. As the counter changes output values, so do the connections to VDD and ground in the resistor string. To keep a constant output voltage of $VDD/2$ the switches in the center of the DAC move accordingly based on the output of the counter and the input to the decoder. In this way variations in the resistors, hopefully, average out to a constant value.

Multibit Demodulator (Used in a NS DAC) Implementation (Error Feedback)

The NS topologies we've discussed so far are sometimes called *interpolative modulators* since the signal fed back is the average of the input signal interpolated between known values of the modulator output (the average of the modulator outputs should be the input signal). However, NS modulators were first introduced (see C. C. Cutler, "Transmission systems employing quantization," 1960, U.S. Patent No. 2,927,962 [filed 1954]) using the error feedback topology shown in Fig. 32.77. Error feedback topologies are not used in analog input modulators because errors in the analog subtraction directly add to the input signal. We can use this topology, however, in the implementation of a digital input demodulator (sometimes also called a modulator), as the subtraction is digital.

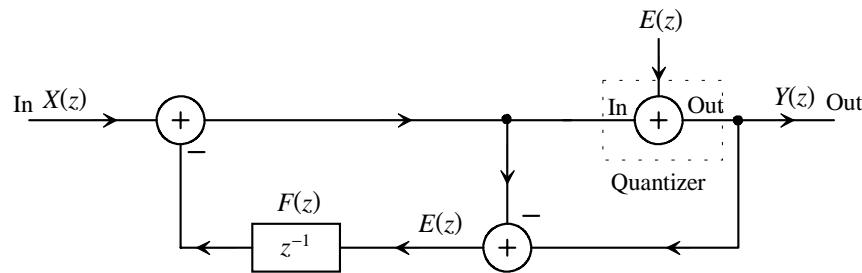


Figure 32.77 Block diagram of an error feedback modulator.

Looking at Fig. 32.77 we note that by definition the difference between the input and the output of the quantizer is the quantization noise, $E(z)$. This noise is subtracted from the input after a delay (for a first-order modulator) resulting in

$$Y(z) = X(z) - F(z) \cdot E(z) + E(z) = X(z) + E(z)[1 - F(z)] \quad (32.110)$$

Note that the signal transfer function for an error feedback-based modulator is simply one; that is, $STF(f) = 1$. For a first-order NS modulator we set $F(z) = z^{-1}$ (a register), which results in

$$Y(z) = X(z) + E(z) \cdot (1 - z^{-1}) \quad (32.111)$$

A second-order modulator with a $NTF(f) = (1 - z^{-1})^2$ would use a feedback filter, noticing from Eq. (32.110) that $NTF(f) = 1 - F(z)$, of

$$F(z) = 1 - (1 - z^{-1})^2 = z^{-1} \cdot (2 - z^{-1}) \quad (32.112)$$

Implementation of a second-order NS modulator is shown in Fig. 32.78. Note that when trying to implement higher-order modulators using error feedback we run into the same problem we encountered when using an interpolative modulator, namely, instability resulting from a NTF that is too large at higher-frequencies. As with interpolative modulators, we can design the NTF to be a highpass response.

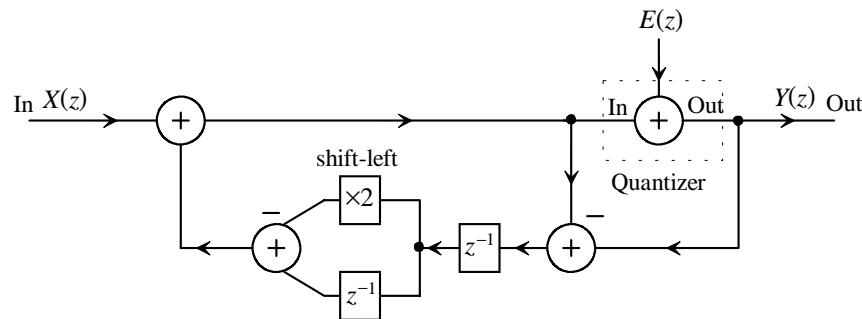


Figure 32.78 Block diagram of a second-order error feedback modulator.

We've introduced the error feedback topology with the idea that it can be used in a modulator (demodulator) that performs digital-to-analog conversion. We first introduced a modulator for use in a DAC back in Fig. 32.12. At this point we need to answer the question, "Why is the NS topology of Fig. 32.77 a better choice for DAC implementation, in general, than the topologies of Figs. 32.12 and 32.71?" The answer to this comes from the realization that the quantizer and difference block in Fig. 32.77 can be implemented by simply removing lower bits from the digital input words. This is illustrated in Fig. 32.79. The resulting error feedback modulator will be simpler to implement than the modulators based on interpolative topologies. Figure 32.80 shows Fig. 32.77 redrawn to show the simpler implementation.

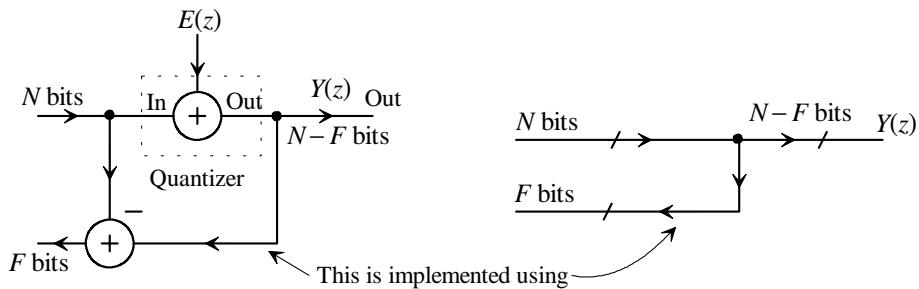


Figure 32.79 Showing how quantizer and difference block are implemented.

The number of bits used in the modulator, N , is selected to avoid overflow when the maximum input signal and fed-back signal are subtracted. When using two's complement numbers, the words input to the adder must be the same length. The smaller word's MSB is used to increase the smaller word's size until the word lengths match. We'll comment more on this important concern in a moment.

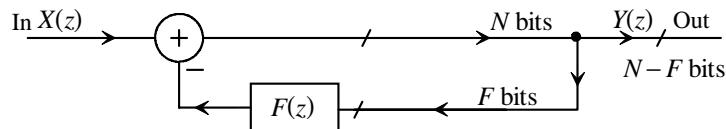


Figure 32.80 Block diagram of an error feedback modulator.

Figure 32.81 shows the block diagram of an NS-based DAC. As we saw in Fig. 32.11, if a 1-bit output is used, the modulator can be connected directly to the reconstruction filter (RCF). The 1-bit DAC is perfectly linear so distortion concerns are reduced. Using a multibit modulator and DAC gives a better SNR, for a given oversampling ratio and modulator order, as well as easing the requirements placed on the RCF. The drawback, as discussed earlier, is that the DAC must be accurate to the final desired output resolution since it is in series with the output signal path.

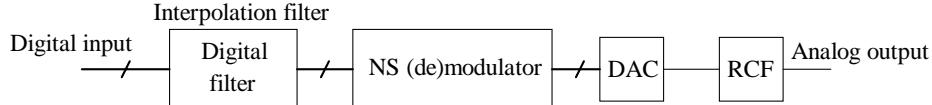


Figure 32.81 DAC using a NS modulator and digital filter.

Let's comment on how to estimate the quantization noise added to the signal from the error feedback quantization process. Assuming we are using two's complement numbers we know

$$\overbrace{011111...}^{N \text{ bits}} = V_{REF+} - \frac{\overbrace{V_{REF+} - V_{REF-}}^{1 \text{ LSB}}}{2^N} \quad (32.113)$$

and

$$1000000... = V_{REF-} \quad (32.114)$$

For the DAC to function properly we must change the numbers back to binary offset (complement the left-most or most-significant bit) unless the DAC input uses two's complement format. This is easy to see if the output of the modulator is a single bit ($N-F=1$) since an MSB of $1 = V_{REF+}$ and a $0 = V_{REF-}$ where $V_{LSB} = V_{REF+} - V_{REF-}$. By dropping F bits the voltage weighting of an LSB in the modulator output can be written as

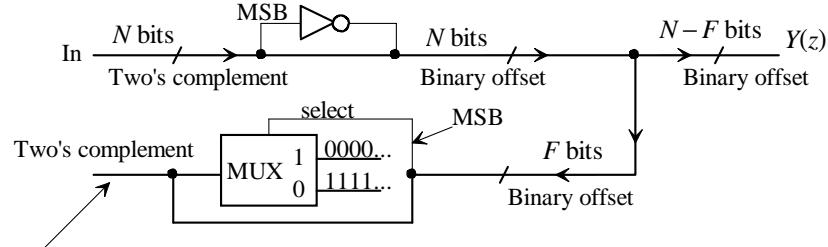
$$V_{LSB} = \frac{V_{REF+} - V_{REF-}}{2^{N-F}} \text{ if } N-F > 1 \quad (32.115)$$

(See question 30.14 or Ex. 35.21 for further discussions.) This result is used in Eq. (32.8) to estimate the quantization noise spectrum in a NS modulator.

Implementation Concerns

We know from our discussions in the last chapter that most digital additions and subtractions utilize two's complement numbers because of the simplicity (see Fig. 31.55 and the associated discussion) in implementing the hardware. However, consider the two's complement N -bit input in Fig. 32.79. If we drop the lower F bits, the resulting number fed back to $F(z)$ (the quantization noise) is not in two's complement format.

To circumvent these types of problems, the topology shown in Fig. 32.82 can be used. The input to the quantizer/subtractor is changed from two's complement format into binary offset format. (See Figs. 31.36 and 31.37 for a comparison of the formats.) Quantization is then performed; the lower bits are dropped from the output and fed back. The fed-back word (the quantization error) is then changed from a binary offset number back into a two's complement number. The size of the word fed back is adjusted to match the size of the modulator's input (knowing that the words used in two's complement arithmetic must be the same size so that the sign bit is in the same location in each word, see also Fig. 31.55).



Adjust, using the MUX, the output word size so that it matches the modulator's input word size. The adder, on the input of the modulator, should see the same size words.

Figure 32.82 Implementation of the quantizer and difference blocks.

32.2.3 Cascaded Modulators

The NS modulators discussed up to this point, in this chapter, have been single feedback loop topologies with the general form seen in Fig. 31.81. This includes the higher order topologies discussed in Sec. 32.2.1 and the error-feedback topologies of the last section. In this section we discuss cascaded or multistage NS modulators. The cascaded modulators discussed here are sometimes called *MultistAge noise Shaping or MASH modulators*. While our focus in this section is on modulators for ADCs, it is easy to extend the theory developed to modulators used in DACs.

We indicated, in the last section, that feeding back the quantization noise, $E(z)$, to the input isn't practical in analog implementations of NS modulators. The output of the analog subtraction ($E[z]$) would be added directly to the input signal. Instead of feeding $E(z)$ back to the input, cascaded modulators feed it forward to the input of another modulator. The second modulator's output is then a delayed version of $E(z)$ as well as its own unwanted modulation noise. If this output is subtracted from the output of the first modulator, we can effectively reduce the resulting overall quantization noise.

The major benefit of a cascaded topology is stability. Unconditionally stable first- and second-order loops can be cascaded to implement higher order modulators. In addition, as we'll briefly discuss, modulators consisting of a first-stage modulator using a 1-bit ADC and DAC followed by a multibit modulator can provide reasonable resolutions with low oversampling ratio K .

Second-Order (1-1) Modulators

A second-order NS modulator can be implemented using a cascade of two first-order modulators (called a 1-1 modulator), as seen in Fig. 32.83. The output of the first modulator is given by

$$Y_1(z) = z^{-1}X(z) + (1 - z^{-1})E_1(z) \quad (32.116)$$

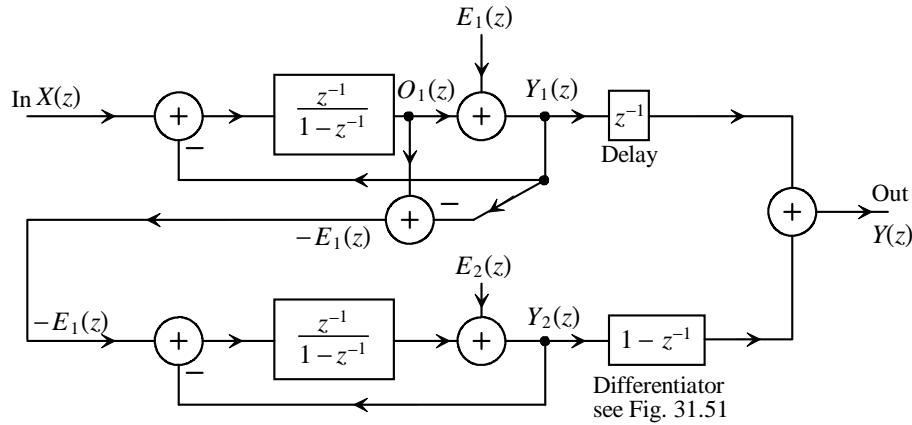


Figure 32.83 Second-order (1-1) cascaded modulator.

while the output of the second modulator is

$$Y_2(z) = -z^{-1}E_1(z) + (1 - z^{-1})E_2(z) \quad (32.117)$$

The overall modulator output is given by

$$\begin{aligned} Y(z) &= z^{-1}Y_1(z) + (1 - z^{-1})Y_2(z) \\ &= z^{-2}X(z) + z^{-1}(1 - z^{-1})E_1(z) - z^{-1}(1 - z^{-1})E_1(z) + (1 - z^{-1})^2E_2(z) \\ &= z^{-2}X(z) + (1 - z^{-1})^2E_2(z) \end{aligned} \quad (32.118)$$

Note how the second modulator is used to subtract the first's quantization noise, $E_1(z)$, from the final output. If all of the components are ideal, the resulting modulator has second-order noise shaping. In practice, however, the coefficients of $E_1(z)$ in Eq. (32.118) will not exactly cancel. When this occurs, $E_1(z)$ is said to *leak to the output of the modulator*. Differences in the coefficients are caused by gain errors in the first modulator's analog integrator when compared to the output of the digital differentiator.

Let's attempt to characterize the performance of the 1-1 modulator if the integrators have gain coefficients, G_I , other than one as seen in Fig. 32.36. We can write the output of the first modulator's integrator in Fig. 32.83 as

$$O_1(z) = \frac{G_{I1} \cdot z^{-1}}{1 + (G_F - 1)z^{-1}} \cdot X(z) - \frac{G_{I1} \cdot z^{-1}}{1 + (G_F - 1)z^{-1}} \cdot E_1(z) \quad (32.119)$$

Using Eq. (32.39) with this equation we can write

$$E_{1out}(z) = Y_1(z) - O_1(z) = \frac{(G_F - G_{I1}) \cdot z^{-1}}{1 + (G_F - 1)z^{-1}} \cdot X(z) + \frac{1 - (1 - G_{I1}) \cdot z^{-1}}{1 + (G_F - 1)z^{-1}} \cdot E_1(z) \quad (32.120)$$

where, ideally, the output quantization noise of the first modulator, $E_{1out}(z)$, is $E_1(z)$. If the modulator is functioning properly, then $G_F = 1$ independent of G_I as discussed earlier. Equation (32.120) can then be written as

$$E_{1out}(z) = (1 - G_{I1}) \cdot z^{-1} \cdot X(z) + [1 - (1 - G_{I1}) \cdot z^{-1}] \cdot E_1(z) \quad (32.121)$$

Using this equation in Eq. (32.117) while assuming the second modulator uses an integrator scaling factor, G_{I2} , and G_{F2} is one results in (rewriting Eq. [32.118])

$$\begin{aligned} Y(z) &= z^{-2}X(z) + z^{-1}(1 - z^{-1})E_1(z) - z^{-1}(1 - z^{-1})E_{1out}(z) + (1 - z^{-1})^2E_2(z) \\ &= \underbrace{z^{-2}X(z) + (1 - z^{-1})^2E_2(z)}_{\text{Desired output}} + \underbrace{[E_1(z) - X(z)] \cdot z^{-2}(1 - z^{-1}) \cdot (1 - G_{I1})}_{\text{Unwanted term}} \end{aligned} \quad (32.122)$$

While we can set the second modulator's integrator gain coefficient, G_{I2} , to 0.4 to avoid integrator saturation, as discussed earlier, we must set G_{I1} as close to unity as possible. Using a unity gain coefficient results in a reduction in the modulator's overall dynamic range (see Fig. 32.35 and the associated discussion). Note how the input signal appears in the unwanted term in Eq. (32.122). It should be obvious at this point that we can add scaling parameters at various points in the modulator to attempt to maximize the modulator's dynamic range. Also note how the number of bits in the 1-1 modulator's output will be more than one bit (two bits if comparators are used in each first-order modulator).

Third-Order (1-1-1) Modulators

By adding a third first-order modulator to our 1-1 modulator of Fig. 32.83 we get a 1-1-1 or third order modulator, Fig. 32.84. The output of the added third modulator can be written as

$$Y_3(z) = -z^{-1}E_2(z) + (1 - z^{-1})E_3(z) \quad (32.123)$$

while the ideal output of the 1-1-1 cascade is given by

$$Y(z) = Y_1(z) + Y_2(z) + Y_3(z) = z^{-3}X(z) + (1 - z^{-1})^3E_3(z) \quad (32.124)$$

Again, as we saw in Eq. (32.122), noise from the first modulator can leak through to the output and spoil the overall cascade's SNR. Indeed, if the leakage from the first modulator is large enough, we get no benefit from adding the third modulator. Notice, in Eq. (32.122), that the unwanted term exhibits first-order differentiation, $(1 - z^{-1})$. We might expect better overall performance, that is, less leakage if the first modulator is second order. The unwanted term would then exhibit second-order differentiation.

Third-Order (2-1) Modulators

A third-order modulator formed by using a second-order modulator followed by a first-order modulator is shown in Fig. 32.85. The output of the first modulator is given by

$$Y_1(z) = z^{-1}X(z) + (1 - z^{-1})^2E_1(z) \quad (32.125)$$

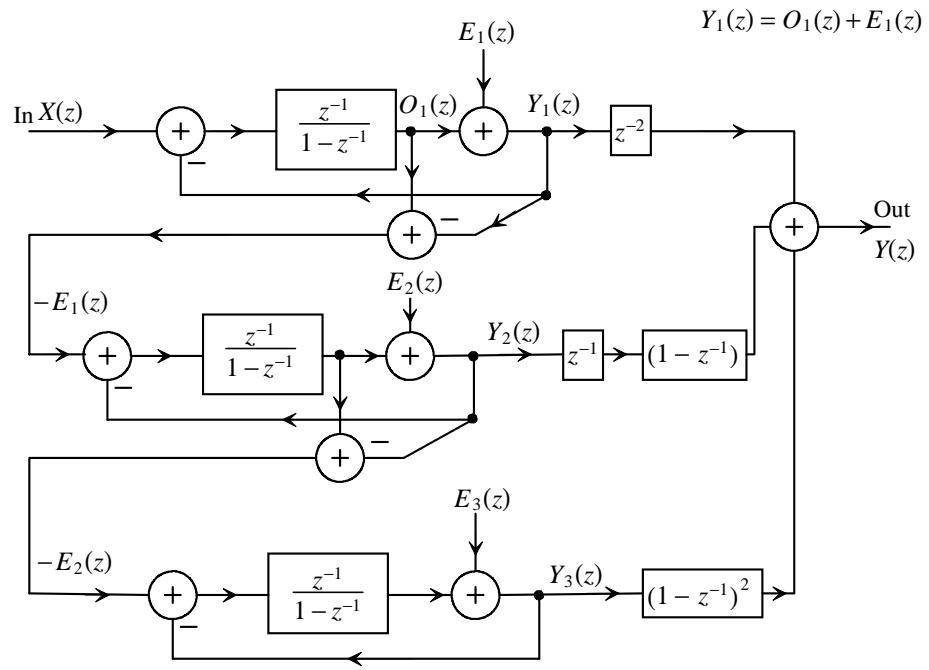


Figure 32.84 Third-order (1-1-1) cascaded modulator.

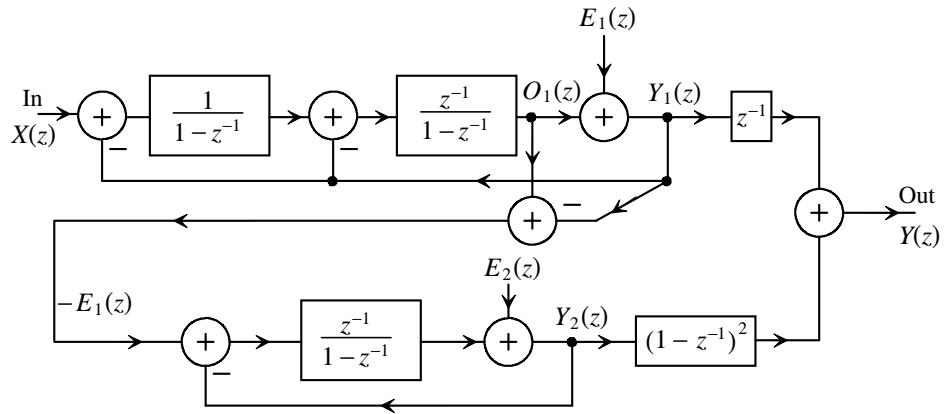


Figure 32.85 Third-order (2-1) cascaded modulator.

while the output of the second modulator is

$$Y_2(z) = -z^{-1}E_1(z) + (1 - z^{-1})E_2(z) \quad (32.126)$$

The output of the 2-1 modulator is then, ideally,

$$Y(z) = Y_1(z)z^{-1} + (1 - z^{-1})^2Y_2(z) = z^{-2}X(z) + (1 - z^{-1})^3E_2(z) \quad (32.127)$$

Let's attempt to characterize the leakage to the output by first determining the output of the second integrator $O_1(z)$ (the input to the comparator). We'll use the topology shown in Fig. 32.58, with $G_3 = 1$, to define our gains. The output, $O_1(z)$, is (assuming that $G_F = G_1G_2G_c = 1$)

$$O_1(z) = \frac{G_1G_2 \cdot z^{-1}X(z) - [(G_1G_2 + G_2) - G_2z^{-1}] \cdot z^{-1}E_1(z)}{1 + z^{-1} \cdot (G_2G_c - 1) + z^{-2}(1 - G_2G_c)} \quad (32.128)$$

Again writing the input to the second modulator as (using Eq. [32.75])

$$\begin{aligned} E_{1out}(z) &= Y_1(z) - O_1(z) \\ &= \frac{(1 - G_1G_2) \cdot z^{-1}X(z) + [(1 - z^{-1})^2 + (G_1G_2 + G_2) \cdot z^{-1} - G_2z^{-2}]E_1(z)}{1 + z^{-1} \cdot (G_2G_c - 1) + z^{-2}(1 - G_2G_c)} \end{aligned} \quad (32.129)$$

noting that if $G_1 = G_2 = G_c = 1$ then $E_{1out}(z) = E_1(z)$. If we write the output of the cascade as

$$Y(z) = z^{-2}X(z) + z^{-1}(1 - z^{-1})^2E_1(z) - z^{-1}(1 - z^{-1})^2E_{1out}(z) + (1 - z^{-1})^3E_2(z) \quad (32.130)$$

then

$$\begin{aligned} Y(z) &= \overbrace{z^{-2}X(z) + (1 - z^{-1})^3E_2(z)}^{\text{Desired output}} + \\ &\quad z^{-1}(1 - z^{-1})^2 \left[\frac{(1 - G_1G_2) \cdot z^{-1}X(z) + [(1 - G_2) - (G_2G_c - G_2)z^{-1}] \cdot z^{-1}E_1(z)}{1 + z^{-1} \cdot (G_2G_c - 1) + z^{-2}(1 - G_2G_c)} \right] \end{aligned} \quad (32.131)$$

When this equation is compared to Eq. (32.122), we see that the undesired term is second-order differentiated. Also, we have more control over the integrator gains. Third-order modulators using the 2-1 topology are much more robust than the 1-1-1-based topology and can provide output signals free of unwanted tones. Again, if integrator saturation (and thus dynamic range) isn't a concern, then we can set $G_1 = G_2 = 1$.

One of the interesting uses of the 2-1 modulator is the configuration where the first (second-order) modulator utilizes a 1-bit ADC and DAC, while the second (first-order) modulator utilizes a multibit ADC and DAC. The overall linearity of this topology is dominated by the second-order modulator, while the multibit modulator provides an enhancement in dynamic range for a given oversampling ratio. These very interesting, and potentially ubiquitous, data converters are discussed in greater detail in Ch. 7 of [2].

Implementing the Additional Summing Input

Before leaving our introduction to cascaded converters, let's discuss the implementation of the extra summing block used to generate the quantization noise, $E(z)$. Figure 32.86 shows the topology of the two summing blocks and how they can be combined.

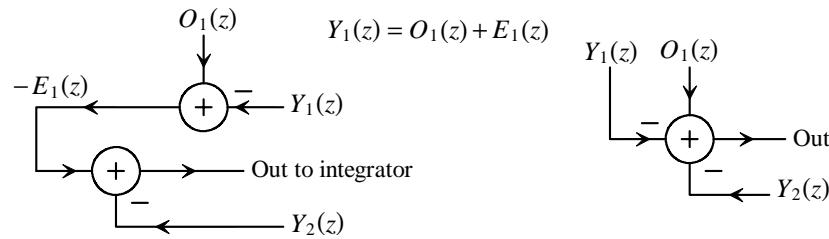


Figure 32.86 Showing implementation of the dual summing block as a single block.

One way to implement the extra subtracting input and the integrator is shown in Fig. 32.87. This DAI is a modification of the DAI shown in Fig. 32.59. The output of this integrator is related to the inputs by

$$V_{out}(z) = O_1(z) \cdot \frac{C_{I2}}{C_{F2}} \cdot \frac{z^{-1/2}}{1-z^{-1}} - Y_2(z) \cdot \frac{C_{I2}}{C_{F2}} \cdot \frac{1}{1-z^{-1}} - Y_1(z) \cdot \frac{C_{I22}}{C_{F2}} \cdot \frac{1}{1-z^{-1}} \quad (32.132)$$

If we set $C_{I2} = C_{I22} = C_{F2}$ and we realize that the comparator in the second modulator, assuming it is clocked with the rising edge of ϕ_1 (or the falling edge of ϕ_2), adds a half-clock cycle delay in series with the $Y_1(z)$ input and a full clock cycle delay in series with $O_1(z)$ and $Y_2(z)$, then we can write

$$V_{out}(z) = [O_1(z) - Y_1(z) - Y_2(z)] \cdot \frac{z^{-1}}{1-z^{-1}} \quad (32.133)$$

Figure 32.88 shows the implementation of a 2-1 modulator.

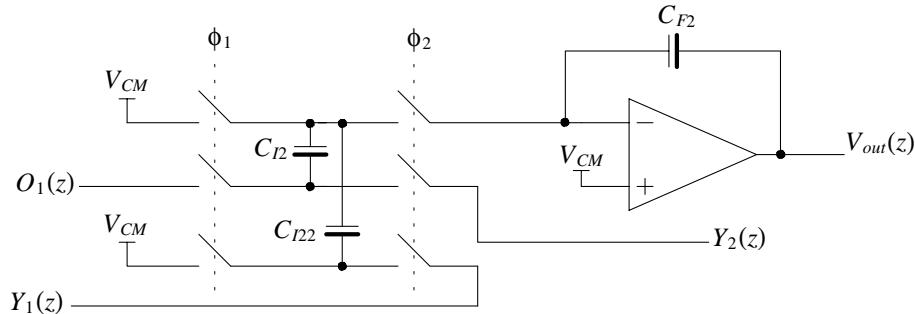


Figure 32.87 Implementing the dual summing block for a cascaded modulator.

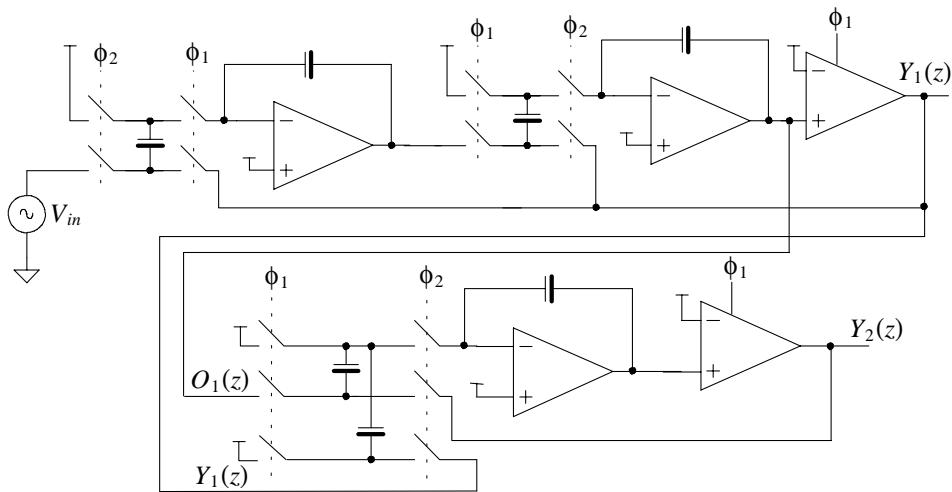


Figure 32.88 Implementation of a 2-1 NS modulator.

We could also use the topology shown in Fig. 32.89 to implement the summing block of Fig. 32.86. This topology has the benefit of using a single capacitor for a simpler circuit and no matching differences between C_{12} and C_{22} . Unfortunately, as discussed in Ch. 27, the topology is no longer insensitive to the parasitic capacitance on the top plate of the switched capacitor. In the parasitic insensitive topologies, Fig. 32.87 for example, the top plate of the capacitor is always held at the common-mode voltage, V_{CM} . In the topology of Fig. 32.89 the top plate is charged to $y_1(t)$ when the ϕ_1 switches are closed and discharged to V_{CM} when the ϕ_2 switches are closed. The difference between these voltages combined with the value of the unwanted parasitic capacitance to ground on the top plate causes unwanted charge to transfer to the feedback capacitor and a gain error. This by itself isn't too bad. However, the unwanted capacitance can have a large depletion capacitance component, resulting in a voltage-dependent capacitance and thus nonlinear gain. Nevertheless, in some applications this topology may still prove useful.

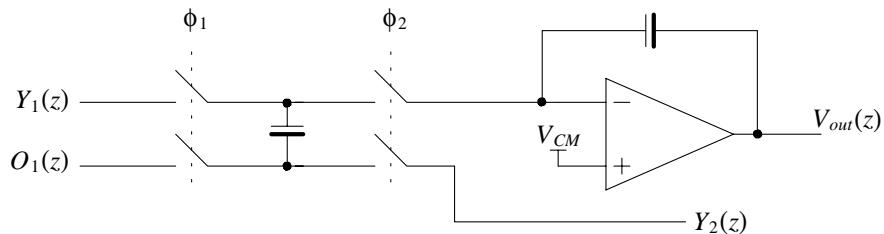


Figure 32.89 Implementing the dual summing block with a single capacitor results in sensitivity to the top plate parasitic capacitance.

32.2.4 Bandpass Modulators

The modulators we've discussed so far in this chapter have been lowpass topologies; that is, the desired signal resides from DC to B . In this section we briefly discuss the idea that we can perform data conversion on a range of frequencies that doesn't include DC. A bandpass modulator is one that pushes the modulation noise away from some desired bandwidth of interest.

In Sec. 31.2.4 we discussed the idea that we can implement a sinc-shaped averaging bandpass filter centered around $f_s/4$ (or $f_s/6$) having a transfer function of

$$H(z) = \frac{1 - z^{-K}}{1 + z^{-2}} \quad (32.134)$$

Comparing this equation to the equation for the equivalent lowpass averaging filter, Eq. (31.99), we can see that transforming our low pass modulator topologies into bandpass modulator topologies with bandpass responses centered at $f_s/4$ can be accomplished by

$$\text{Substituting } z^{-2} \text{ for } -z^{-1} \quad (32.135)$$

Figure 32.90 shows the result of a lowpass second-order modulator transformed into a fourth-order bandpass modulator. It's now called a fourth-order modulator because the number of poles in the NTF is now four. The transfer function for this, $f_s/4$, bandpass modulator is

$$Y(z) = X(z) \cdot \overbrace{(-z^{-2})}^{\text{STF}} + E(z) \cdot \overbrace{(1 + z^{-2})^2}^{\text{NTF}} \quad (32.136)$$

Writing the modulation noise for a bandpass modulator results in

$$|NTF(f)|^2 \cdot |V_{Qe}(f)|^2 = \frac{V_{LSB}^2}{12f_s} \cdot \left[2 \cos 2\pi \frac{f}{f_s} \right]^4 \quad (32.137)$$

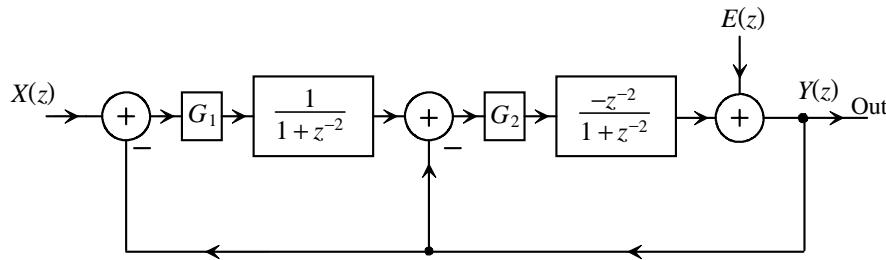


Figure 32.90 Block diagram of a fourth-order bandpass modulator.

Figure 32.91 shows the modulation noise spectrum of a fourth-order modulator, Eq. (32.127). Notice how, at $f_s/4$, the modulation noise goes to zero. The oversampling ratio is, once again, defined as

$$K = \frac{f_s}{2B} \quad (32.138)$$

If $f_s/4 = 25$ MHz and the desired bandwidth is 50 kHz, then $K = 1,000$.

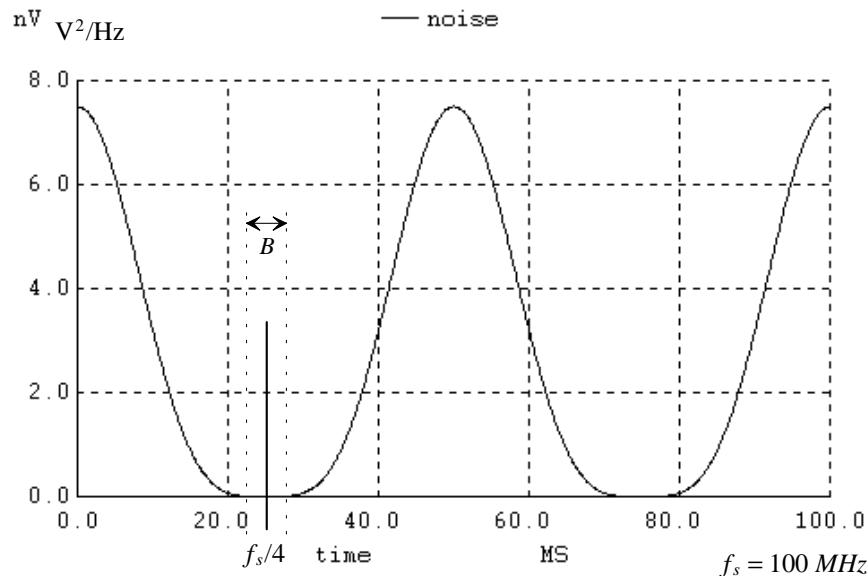


Figure 32.91 Modulation noise in a bandpass modulator.

Implementing a Bandpass Modulator

Toward implementing a bandpass modulator, consider the circuit shown in Fig. 32.92. The top portion of the circuit is simply the DAI discussed in Sec. 31.3.1. The bottom portion provides the positive feedback needed for the addition (instead of subtraction) of the delayed output. We've adjusted the values of the capacitors so that the resulting transfer function of the DAI is (see question 31.51 in the last chapter)

$$V_{out}(z) = \frac{V_1(z) \cdot z^{-1/2} - V_2(z)}{1 + z^{-1}} \cdot \frac{C_I}{C_F} \quad (32.139)$$

To change the denominator in this equation to $1 + z^{-2}$, we can use a clocking frequency of $f_{s,\text{new}} = 2f_s$, effectively changing the z^{-1} to z^{-2} where z is still $e^{j2\pi f_s t}$. In most practical implementations fully-differential integrators are used [6]. The inverting block in Fig. 32.92 is implemented with the inverting output of the fully-differential op-amp.

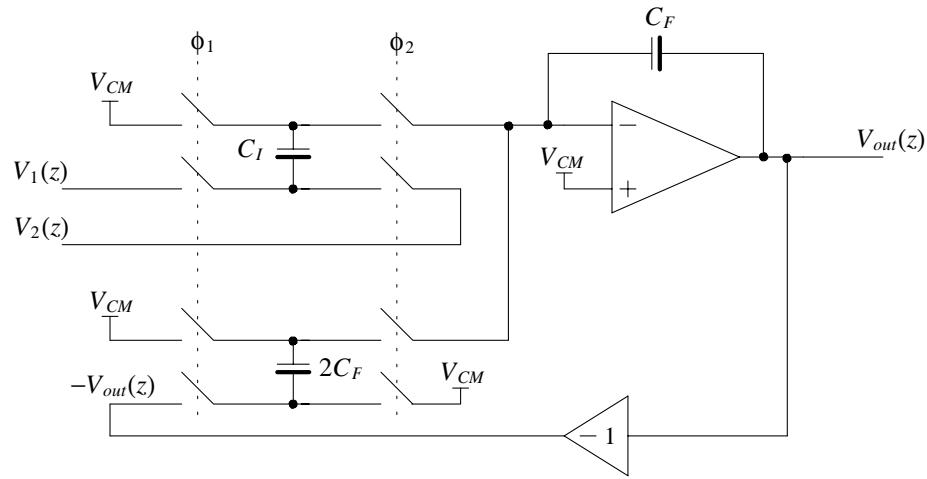


Figure 32.92 Implementing a DAI for use in a bandpass modulator.

In a bandpass modulators $1/f$ noise isn't a concern since it is filtered out with the digital filter. If the modulator is used in a wireless application where the intermediate frequency (IF) is $f_s/4$, extracting the real (I, or in-phase component) and imaginary (Q, or quadrature component) becomes trivial (see Fig. 32.93). Since a single modulator is used, and the I/Q extraction is digital, the channel mismatch encountered in typical baseband demodulators is absent.

We might wonder, after looking at this figure, how we multiply the modulator output, a 1-bit word, by 1, 0, and -1 . We know that a 1 coming out of the modulator corresponds to V_{REF+} and a 0 corresponds to V_{REF-} . After reviewing Fig. 31.37 in the last

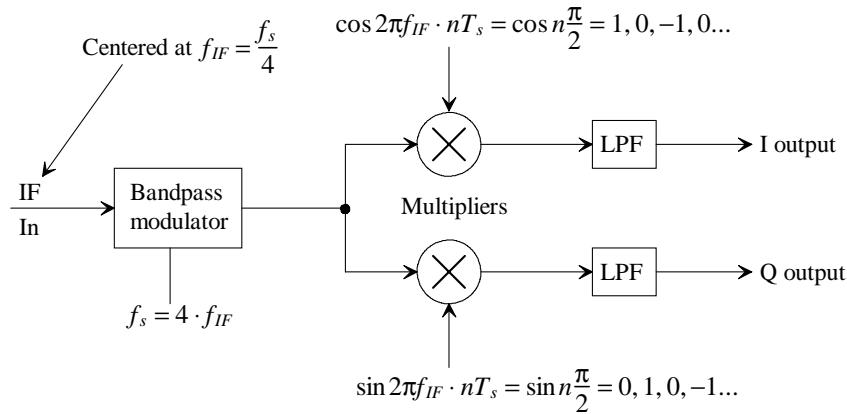


Figure 32.93 Digital I/Q demodulation.

chapter, we can convert these 1-bit outputs to 2-bit words in two's complement format. A 1 output is changed to 01 (+1) and a modulator output of 0 becomes 11 (-1) in two's complement prior to multiplication. Multiplication results in $01 \times 1 = 01$, $01 \times 0 = 00$, $01 \times -1 = 11$, $11 \times 1 = 11$, $11 \times 0 = 00$, or $11 \times -1 = 01$. A simple MUX with some logic for output selection can be used to implement the multiplier.

REFERENCES

- [1] J. C. Candy and G. C. Temes (eds.), *Oversampling Delta-Sigma Data Converters*, IEEE Press, 1992. ISBN 0-87942-285-8
- [2] S. R. Norsworthy, R. Schreier, and G. C. Temes (eds.), *Delta-Sigma Data Converters: Theory, Design, and Simulation*, IEEE Press, 1996. ISBN 0-7803-1045-4
- [3] R. K. Hester, *Introduction to Oversampled Data Conversion*, Notes from a tutorial at the 1995 International Solid-State Circuits Conference (ISSCC-95).
- [4] P. A. Lynn and W. Fuerst, *Introductory Digital Signal Processing*, Second Edition, John Wiley and Sons, 1998. ISBN 0-471-97631-8
- [5] E. P. Cunningham, *Digital Filtering: An Introduction*, John Wiley and Sons, 1995. ISBN 0-471-12475-3
- [6] A. K. Ong, and B. A. Wooley, "A Two-Path Bandpass SD Modulator for Digital IF Extraction at 20 MHz," *IEEE Journal of Solid-State Circuits*, Vol. 32, No. 12, pp. 1920-1934, December 1997.

QUESTIONS

- 32.1** Show the details and assumptions leading to Eq. (32.1).
- 32.2** Would it be possible to operate the DAI of Fig. 32.3 without a 0.75 V supply? Give an example. Show simulation results with the output initially at 0.75 V and the same input used to generate Fig. 32.4. Are the DAI outputs the same?
- 32.3** Show the derivation of Eq. (32.4) from the block diagram shown in Fig. 32.6.
- 32.4** In the basic NS modulator shown in Fig. 32.7, what component serves as the ADC? What component serves as the DAC?
- 32.5** Show, using timing diagrams, how Eq. (32.5) is correct.
- 32.6** Show, using SPICE simulations, how increasing the RC circuit's time constant in Fig. 32.10 removes additional modulation noise making the output smoother. What happens to the amplitude of the desired signal?
- 32.7** Show the spectrums (modulator input, output, and output after filtering) of the signals in question 32.6. Discuss what the spectrums indicate.
- 32.8** Explain how the quantizer in Fig. 32.12 functions.

- 32.9** What are we assuming about an input signal if the modulation noise follows Eq. (32.7)?
- 32.10** What is the magnitude of Eq. (32.7)?
- 32.11** What is the difference between quantization noise and modulation noise?
- 32.12** Show the steps and assumptions leading to Eq. (32.15)?
- 32.13** Is the statement that on page 163 that "every doubling in the oversampling ratio results in 1.5 bits increase in resolution" really true if K is small (say 8 or 16)? Explain.
- 32.14** Does noise-shaping work for DC input signals? If so, how?
- 32.15** Show the steps leading up to Eq. (32.25).
- 32.16** What is the impulse response of the following z-domain transfer function?

$$H(z) = \left[\frac{1 - z^{-16}}{1 - z^{-1}} \right]^2$$

- 32.17** Regenerate Fig. 32.23 if $L = 3$. What is the droop at B ?
- 32.18** Is it possible to eliminate the op-amp in Fig. 32.24 and use the following topology? Comment on the problems associated with this topology.

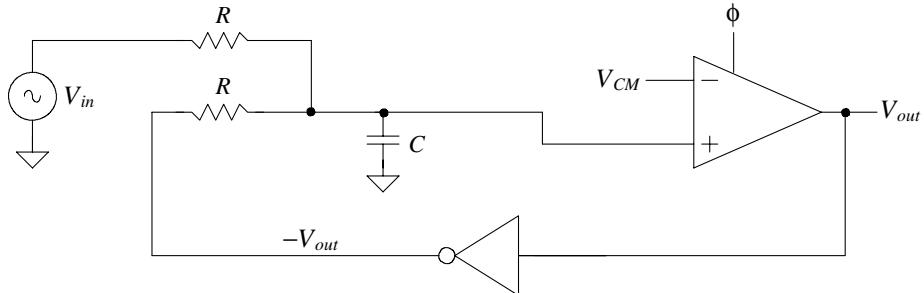


Figure 32.94 First-order NS modulator for question 32.18.

- 32.19** Simulate the operation of the circuit shown in Fig. 32.94.
- 32.20** Sketch a modulator output, similar to Fig. 32.27, if the input is 0.2 V.
- 32.21** In your own words, describe ripple in the output of a digital filter connected to an NS modulator.
- 32.22** Does adding a dither signal to the input of a NS modulator help reduce the peak-to-peak ripple in the digital filter output? Does it help to break up tones in the filter's output?
- 32.23** Derive Eq. (32.39).

- 32.24** Repeat Ex. 32.11 if the integrator's gain is set to 0.5.
- 32.25** Verify that Eq. 32.43 is correct. Use pictures if needed.
- 32.26** Would large parasitic op-amp input capacitance affect the settling time of a DAI?
- 32.27** Determine the transfer function of the DAI shown in Fig. 32.43.
- 32.28** Derive Eq. (32.65).
- 32.29** Sketch the implementation of the full-differential second-order NS modulator.
- 32.30** Derive Eq. (32.75)
- 32.31** Sketch the fully-differential equivalent of Fig. 32.59.
- 32.32** Resimulate the modulator in Ex. 32.13 if the gains are set to one. Comment on the stability of the resulting circuit.
- 32.33** Resimulate the modulator in Ex. 32.13 if the input is only 50 mV. Comment on the stability of the resulting circuit.
- 32.34** Regenerate Fig. 32.67 by selecting integrator gains so that the maximum output swing of any op-amp is 1.3 V.
- 32.35** Comment, in your own words, on why the actual SNR of a NS-based data converter can be worse than the ideal values calculated in the chapter.
- 32.36** Derive Eq. (32.91). Make sure each step of the derivation includes comments.
- 32.37** Resimulate Fig. 32.74 using two-bit ADC and DAC.
- 32.38** Sketch a possible implementation of a quantizer for the error feedback modulator shown in Fig. 32.78.
- 32.39** What transfer function does the following block diagram implement?

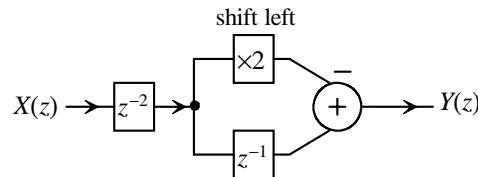


Figure 32.95 Circuit for Question 32.39.

- 32.40** In Fig. 32.84 sketch the block diagram implementation of the circuit in series with the $Y_2(z)$ output.
- 32.41** Sketch the block diagram implementation of the transfer function $(1 - z^{-2})^2$. What kind of filter does this transfer function implement?

- 32.42** Sketch the implementation of the multipliers in Fig. 32.93.
- 32.43** Would clock jitter be a concern in a bandpass modulator? (Hint: Review Fig. 31.14 in the last chapter and the associated discussion.)

Chapter

33

Submicron CMOS Circuit Design

In this chapter we turn our attention toward transistor-level circuit design using a submicron CMOS process, that is, a CMOS process with a minimum channel length, L_{min} , less than 1 μm . We divide the chapter up into three sections. The first section covers basic submicron CMOS processes and device models. The second and third sections provide digital and analog circuit design examples and discussions, respectively. We assume throughout the chapter, that the reader is well grounded in the material presented in *CMOS: Circuit Design, Layout, and Simulation* [1] (the first CMOS book).

Before getting too far into this chapter, let's discuss how we distinguish the material presented here from the material presented in the first CMOS book. If we recall, in the first CMOS book, an older CMOS process was used to illustrate the fundamental design ideas, methods, and procedures and to provide practical models for comparing hand-calculations and simulation results. The older CMOS devices followed the "square-law" MOSFET model. While one can argue that the older CMOS processes will never be obsolete because of their inherent higher voltage handling capability (a modern CMOS process generally limits VDD , with $VSS = 0$, to between 1 and 3.3 V), this isn't the main reason for using them to illustrate design fundamentals. The main reason comes from the fact that hand-calculation parameters (transconductance and output resistance, for example) are derived from the SPICE level 1 square-law model. An older MOSFET that can then be modeled relatively well using the level 1 model yields simulation results that match hand-calculations and provides immediate feedback to the designer that "I know what I'm doing." The level 1 model can accurately model devices with an $L_{min} > 5 \mu\text{m}$. Reasonable hand-calculation accuracy, say within 20%, can be achieved in a process with L_{min} between 1 and 5 μm (in any case, though, the most accurate SPICE MOSFET model available should be used). However, if hand-calculations, based on the level 1 square-law model, are used in a submicron CMOS process, the error is generally well above 100%! As an example, Fig. 33.1a shows an IV plot of a typical 0.5 μm ($= L_{min}$) NMOS transistor, while Fig. 33.1b shows the level 1 SPICE results.

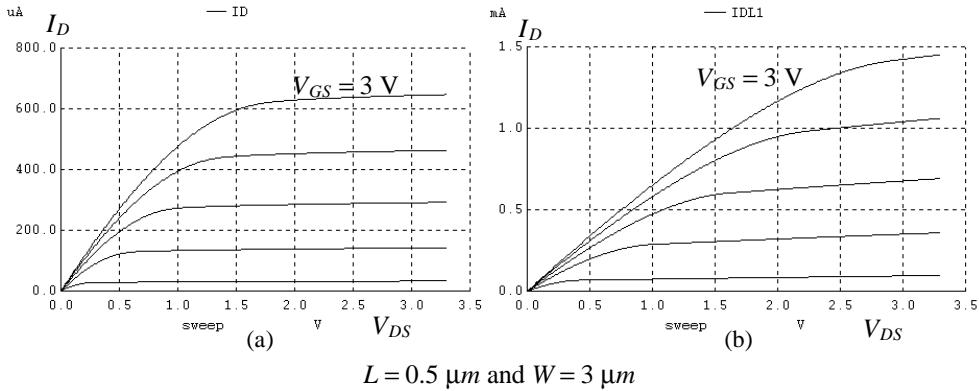


Figure 33.1 (a) IV characteristics of a submicron MOSFET, and (b) its level 1 SPICE representation.

Looking at Fig. 33.1, we should make several key observations. To begin, notice how the drain current, I_D , varies as the square of the gate-source voltage, V_{GS} , in the level 1 model (b), while the variation in the actual curves (a) is more or less linear with V_{GS} . Next, notice how the point the MOSFET enters the saturation region, $V_{DS,sat}$, is defined by $V_{GS} - V_{THN}$ (the gate-source voltage minus the NMOS threshold voltage) in the level 1 model (b). This, as was discussed in Chs. 5 and 6, results in an overestimate of $V_{DS,sat}$ and is one reason body-effect is neglected so often in DC hand-calculations. Finally, notice how the slope of the curves, when the MOSFET is operating in the saturation region, is larger in (b) than in (a). We know that the small-signal output resistance of the MOSFET is the reciprocal of this slope. Further then, we would expect the level 1 model to provide lower values of gain in simulations than a model that more exactly matches part (a).

33.1 Submicron CMOS: Overview and Models

In this section we discuss the CMOS process flow [2] for devices with L_{min} less than $0.35 \mu\text{m}$, implementation of capacitors and resistors, and lastly the EKV MOSFET model [3-5] and why we'll use this model to illustrate circuit design techniques in this book.

33.1.1 CMOS Process Flow

Figure 33.2 shows the basic CMOS process flow for a sub-0.35 μm process. This process flow illustrates the differences between a submicron process and the older process flows discussed in Chs. 2–4. In particular, (1) shallow trench isolation (STI) is used instead of the local oxidation of silicon (LOCOS) for device isolation, (2) n+ poly is used for NMOS formation while p+ poly is used in PMOS formation, (3) lightly doped drains are used to reduce short channel effects, (4) silicided source/drains/gates are used to reduce parasitic resistances, and (5) both devices are surface devices (the PMOS is no longer a buried channel device).

We can summarize the process steps, shown in Fig. 33.2, as follows:

- Part (a) starts with a p-type wafer or a p+ wafer with a p- epitaxial layer as discussed in Ch. 2. The active areas are patterned by first depositing a thin oxide and a nitride. A photoresist is deposited and patterned on the top of the nitride. The exposed areas of nitride are removed
- Part (b) shows the result of etching down into the silicon areas that are exposed, that is, those not covered with the photoresist (or nitride after the nitride is etched off).
- Part (c) shows the cross-section of the wafer after a thick oxide has been deposited over the entire wafer and the top of the wafer has undergone polishing using chemical-mechanical polishing (CMP). Notice how the top of the wafer is flat. Also notice the shallow trenches are filled with a chemical vapor deposited (CVD) oxide (called STI). STI is used in place of LOCOS because of the ability to define smaller openings in the top of the wafer (smaller active area windows). The effective encroachment on the devices' width is reduced and the MOSFETs can be placed closer together.
- Part (d) shows the implant used for making the body of the PMOS transistors (the n-well) and the implants used to adjust the threshold voltage.
- Part (e), at the top of the next page, shows the result of patterning the polysilicon gates on the top of the wafer.

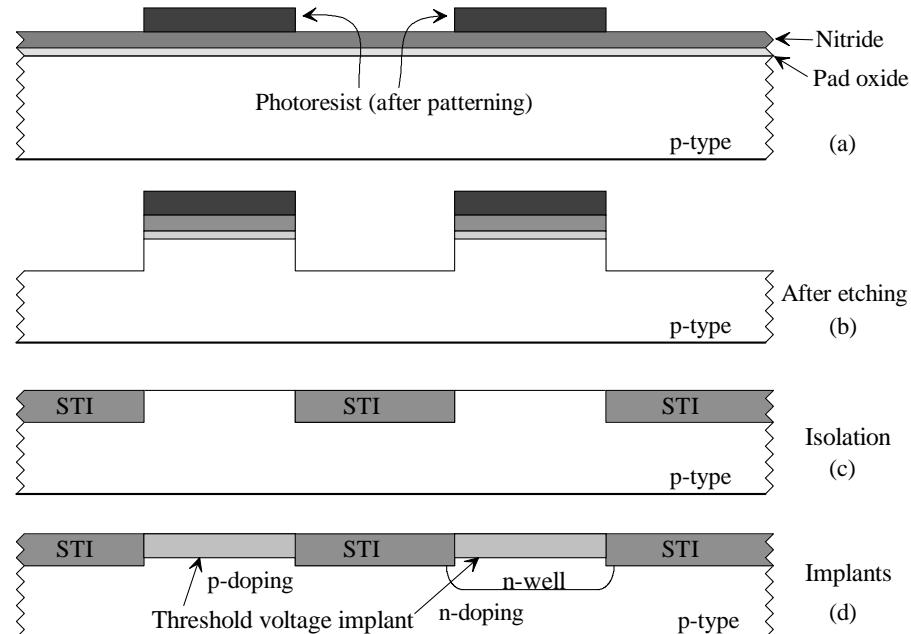


Figure 33.2 CMOS process flow.

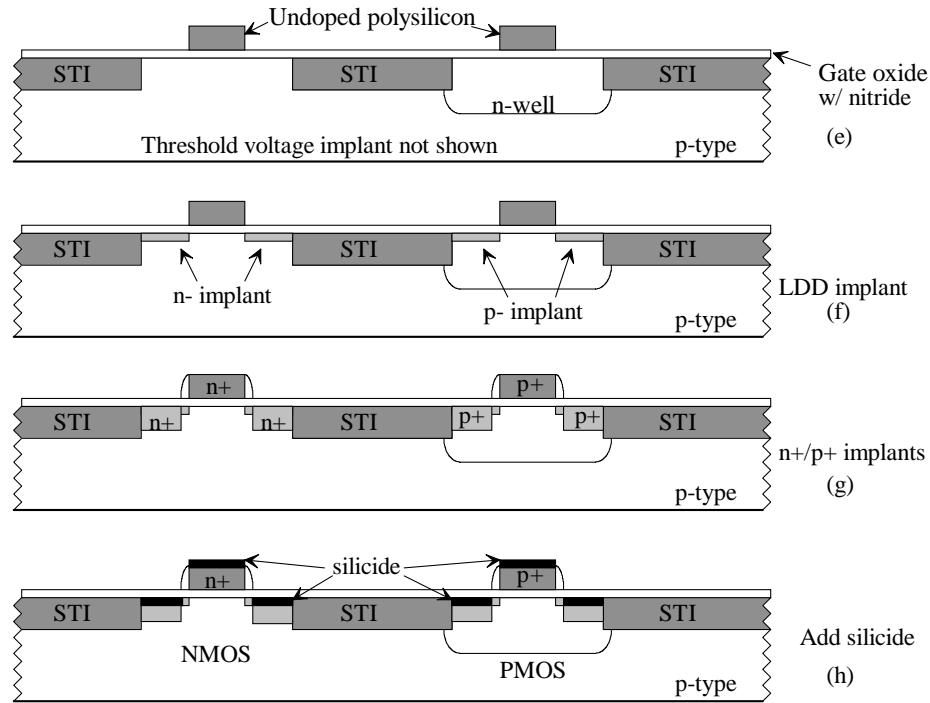


Figure 33.2 (cont'd) CMOS process flow.

- In part (f) light, and shallow, implants are shown which are used in lightly doped drain, or LDD, MOSFET formation.
- Part (g) shows formation of the lateral oxide spacer adjacent to the gate poly (used for LDD MOSFET formation) and the results of using implants to heavily dope the gates, sources and drains. Implanting the gate polysilicon is important. The p+ poly used in the PMOS formation here results in a surface device (conduction between the drain and source occurs along the oxide/semiconductor interface) rather than a buried device (conduction occurs through a buried channel). The threshold voltage of the PMOS is easier to set precisely (because we don't have to counter dope the channel) and the short channel effects become less severe. The drawbacks of switching from a buried-channel PMOS to a surface-channel PMOS are the reduction in mobility and the increase in flicker ($1/f$) noise.
- Finally, part (h) shows the cross-sectional view of the resulting NMOS and PMOS devices after a silicide (combination of silicon and a refractory metal such as tungsten) has been deposited. The addition of the silicide complicates the process but produces devices that have significantly less parasitic series gate and source/drain resistance.

33.1.2 Capacitors and Resistors

As we saw in the last chapter, capacitors are an important component when implementing a mixed-signal integrated circuit (IC). In this section we briefly discuss implementing capacitors and resistors in a submicron CMOS process. While in some of these processes (for example, a flash memory process) two layers of poly are available for capacitor formation, here we assume that poly-poly capacitors (discussed in Ch. 7) are not available.

Using a MOSFET as a Capacitor

Figure 33.3 shows the CV curve for an NMOS transistor with source, drain, and body connected to ground. We might try to use this device in one of the discrete-analog-integrators (DAIs) of the last chapter. However, the capacitors used in the DAI are "bipolar" (the voltage across them can be positive or negative). We see in the Fig. 33.3 that V_{GS} should be much greater than 400 mV (the threshold voltage) for the device to behave as a capacitor. If V_{GS} falls close to the knee slightly above V_{THN} , then the capacitance can become nonlinear and distortion can appear in a circuit's output. Note that this curve shifts to the right with a nonzero V_{SB} .

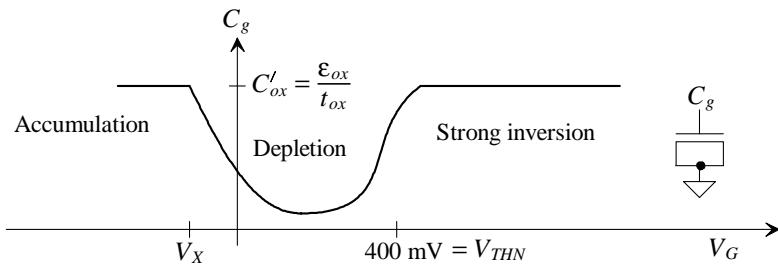


Figure 33.3 Using a MOSFET as a capacitor.

Using a Native or Natural MOSFET Capacitor

Figure 33.4 shows the layout, CV curves, and schematic symbol of the native, or also sometimes called the natural, MOSFET capacitor. The native MOSFET capacitor is formed by laying out poly over n+ active in an n-well. The result is a shift in the MOSFET's threshold voltage, V_{THN} . While the native MOSFET still cannot be used as a bipolar capacitor, it does find uses in many applications where low voltages are a concern.

Note that we might be tempted to use this MOSFET in an amplifier (not as a capacitor) as a near-depletion mode device. However, since the source and drain are shorted together through the resistive n-well, the resulting device may not be too useful.

The Floating MOS Capacitor

A novel method of implementing a capacitor using regular PMOS transistors is shown in Fig. 33.5 [6]. Two PMOS devices are laid out either adjacent or interdigitated in the *same* n-well. For the moment let's not concern ourselves with the exact voltage of the n-well,

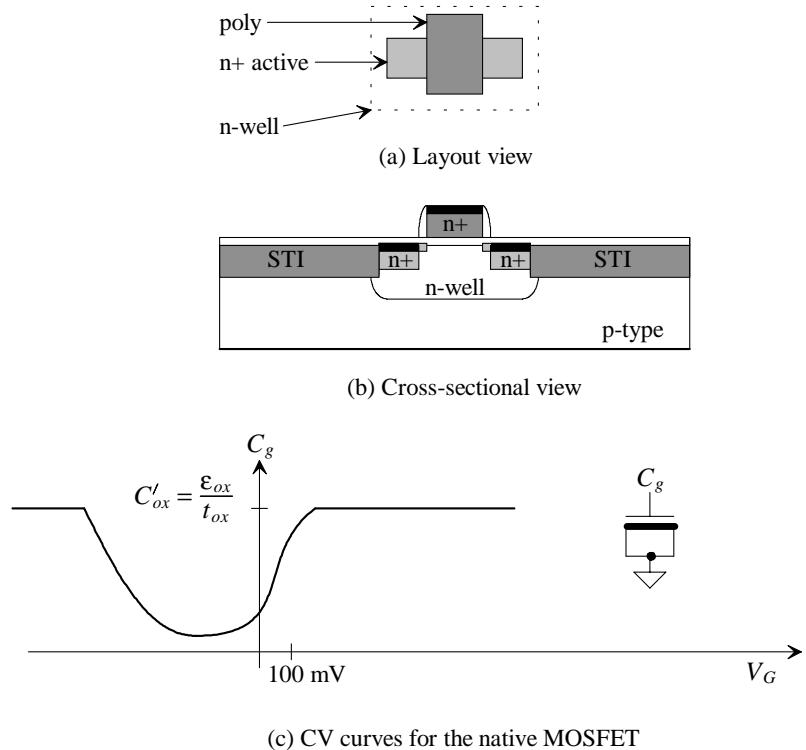


Figure 33.4 The native or natural MOSFET.

but rather assume that the capacitor and resistor time constant is so large compared to the signal frequencies of interest that little charge, ideally zero on average, flows in the big resistor. An increase in voltage on A, in the figure, will cause an accumulation of charge under the left MOSFET's gate oxide. At the same time an equal and opposite charge will appear under the right MOSFET's gate oxide. The result is that the charge accumulated on B will be equal and opposite to the charge accumulated on A. Since the capacitors are in series the overall capacitance seen between A and B, C_{AB} , is the series connection of each MOSFET's own gate-oxide capacitance as seen in the figure.

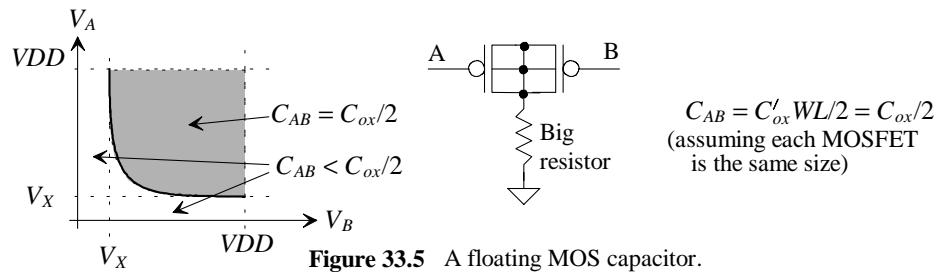


Figure 33.5 A floating MOS capacitor.

To implement the big resistor in Fig. 33.5, a topology like the one seen in Fig. 33.6 can be used [6]. The long L PMOS device is used to generate a very small current. This current is mirrored across and used to hold the n-well and p+ active areas at ground (on average). Because the current is so small, the actual voltages seen in the n-well and active areas can be considerably different than ground over short periods of time allowing the capacitor action discussed above to occur. Finally, notice that the MOSFETs used in the capacitor operate in accumulation as long as the voltage on either side of the capacitor is greater than V_x . For the PMOS equivalent of Fig. 33.3, V_x is positive and V_{THP} is negative with the source/drain and body grounded.

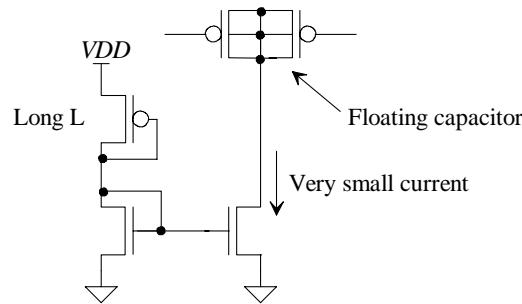


Figure 33.6 Implementing a large resistor for the floating MOS capacitor.

Metal Capacitors

Probably the most common method of making capacitors in a submicron CMOS process is using the metal layers. Consider the cross-sectional view of a parallel plate capacitor shown in Fig. 33.7. If the plate capacitance between the metal1 and metal2 dominates because the metals have a large layout area (that is, the fringe capacitance contribution is small), then the capacitance can be estimated using

$$C_{12} = \text{Area} \cdot (\text{capacitance per area}) \quad (33.1)$$

If the capacitance per area is $50 \text{ aF}/\mu\text{m}^2$, then it would take an area of $100 \mu\text{m}$ by $200 \mu\text{m}$ to implement a 1 pF capacitor. While large area is a problem, it isn't the main problem with a metal parallel-plate capacitor. The main problem occurs from the extremely large bottom

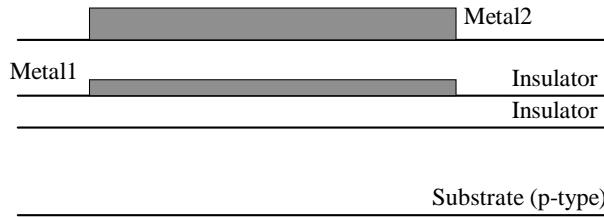


Figure 33.7 Parallel-plate capacitor using metal1 and metal2.

plate parasitic capacitance, that is, the capacitance from metal1 to substrate. This parasitic capacitance can be anywhere from 80 to 100% of the desired capacitance. Further it usually slows the circuit response and results in a waste of power.

To help decrease the bottom plate's percentage of the desired capacitor value, consider the cross-sectional view shown in Fig. 33.8, where four layers of metal implement a capacitor. The capacitance of this structure can be estimated using

$$C = C_{12} + C_{23} + C_{34} \quad (33.2)$$

If plate capacitance between each metal layer is, again, $50 \text{ aF}/\mu\text{m}^2$, then the area required to implement a 1 pF capacitor is $100 \mu\text{m}$ by $66 \mu\text{m}$. The area needed is reduced by one-third of the area used in the metal1/metal2-only capacitor. While we used the same plate capacitance value in between each level, we know that this will vary because of the differing thickness in between the metals. The absolute value of the capacitors, in most situations, isn't important but rather the ratio of capacitors is the important parameter, as seen in the last chapter. Also notice how, in a modern CMOS process, the thickness of the metals (made most often now with copper) increases as we move away from the substrate.

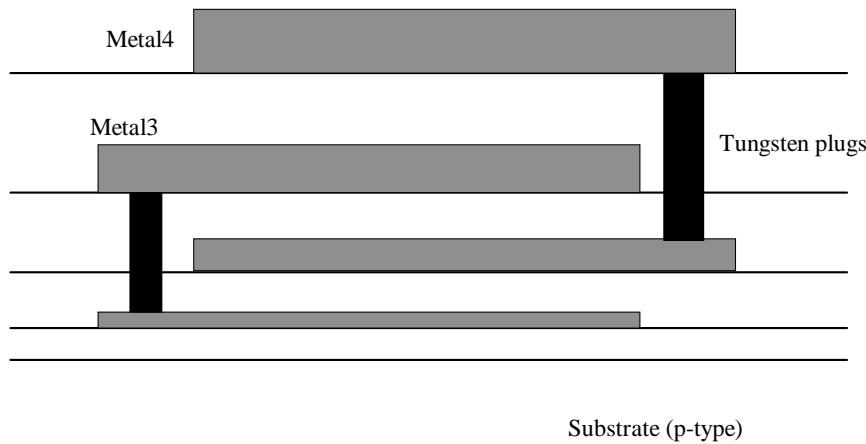


Figure 33.8 Cross-sectional view of a parallel plate capacitor using metal1-metall4.

The value of the capacitors in Figs. 33.7 and 33.8 was set by the areas of the metals and the corresponding plate capacitance. We assumed the perimeter of the metals and the resulting fringe capacitance was a small contributor. Figure 33.9 shows typical minimum sizes and distances between pieces of metal1 where the fringe capacitance dominates. We can make a capacitor using the two pieces of metal1 shown in this figure. A typical value of capacitance per length is $25 \text{ aF}/\mu\text{m}$. The parasitic bottom plate capacitance is half of this value or $12.5 \text{ aF}/\mu\text{m}$. Since, as seen in Fig. 33.9, the electric fields can terminate on the close adjacent metal, the bottom component is a smaller percentage than it was when the plate capacitance dominated.

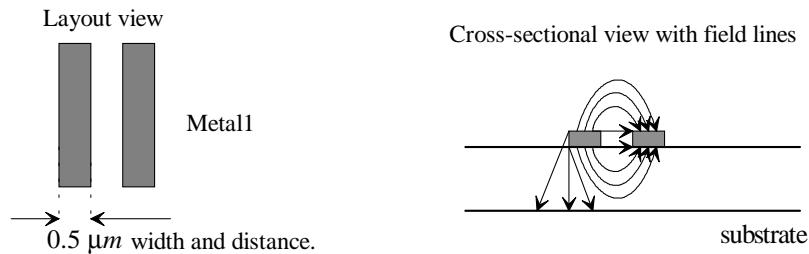


Figure 33.9 Typical size when fringing capacitance dominates.

Example 33.1

Estimate the size of a metal1 only 1 pF capacitor. Also estimate the bottom parasitic capacitance.

If we look at the layout of the capacitor shown in Fig. 33.10, we can estimate the capacitance of a 1 μm by 2 μm section as 25 aF/ μm^2 . This would mean that we need an area of metal1 that measures 200 μm by 200 μm to implement a 1 pF capacitor. The bottom plate capacitance can be estimated as 0.5 pF.

Note that while this capacitor results in twice the area of the metal1/metal2 capacitor of Fig. 33.7 and the associated discussion, it only uses one layer of metal and the bottom parasitic is smaller. One might wonder if further benefits can be achieved by using the fringe capacitance and multilevels of metal. ■

Consider the use of metal2 and via1 in the implementation of a capacitor shown in Fig. 33.11. While the fringe capacitance is still a major component in this capacitor because of the addition of the via between the metals, it is sometimes called a lateral capacitor (there exists a "plate" capacitance between the vias). A typical value of

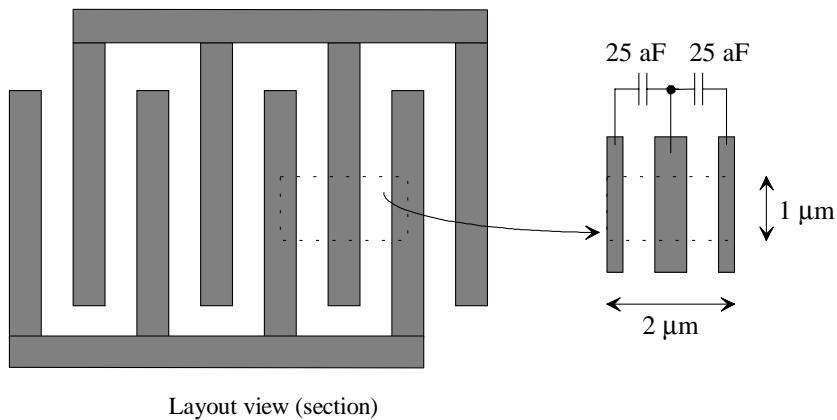


Figure 33.10 Layout of a 1 pF capacitor using only metal1.

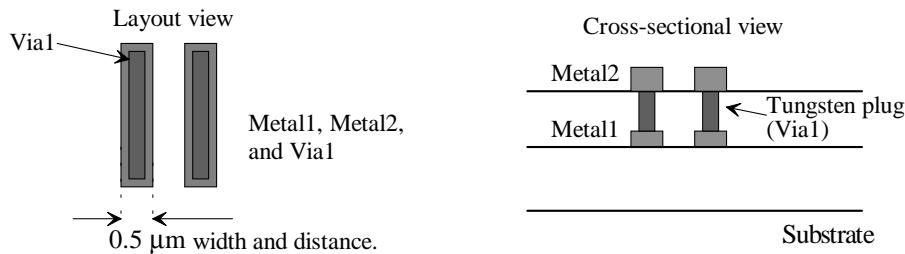


Figure 33.11 Using two layers of metal and the via to implement a lateral capacitor.

capacitance for this structure is $200 \text{ aF}/\mu\text{m}$. The bottom plate capacitance remains approximately $15 \text{ aF}/\mu\text{m}$. Using additional vias and metal layers will increase the capacitance but generally not linearly. The higher levels of metal, e.g. metal4 or metal5, generally have larger spacing and width design rules than do the lower levels of metal. *Nonetheless, using the lateral capacitor with several layers of metal and vias is the most common way to implement a capacitor in a mixed-signal circuit.*

Example 33.2

Repeat Ex. 33.1 using the lateral capacitor of Fig. 33.11.

Since the capacitance per length is now $200 \text{ aF}/\mu\text{m}$, the area required for a 1 pF capacitor, keeping in mind that both metal1 and metal2 are used, is $50 \mu\text{m}$ by $100 \mu\text{m}$. The bottom plate capacitance becomes 0.15 pF .

It's interesting to note that if we used four layers of metal with three vias and a lateral capacitance value of $500 \text{ aF}/\mu\text{m}$, the area drops to $50 \mu\text{m}$ by $40 \mu\text{m}$. ■

An Important Note

While we've concentrated on the bottom plate parasitic, it is also possible to have a top plate parasitic. Often, to avoid coupling noise into the relatively large area occupied by the capacitor, a ground plate is placed above the capacitor. This would allow noisy digital signals to be routed above the capacitor, as seen in Fig. 33.12.

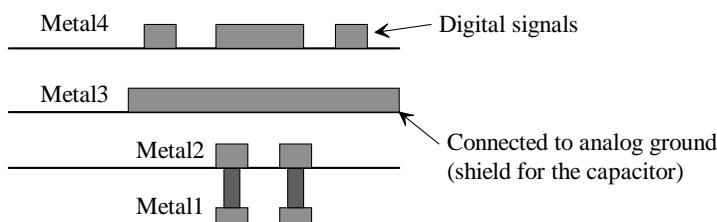


Figure 33.12 Using a metal3 shield to isolate the lateral capacitor.

Resistors

Using a large number of capacitors in a circuit can result in large layout area, as just discussed. Because of this, resistors are used whenever and wherever possible. For example, a DAC may have been implemented using a charge redistribution topology in the past. Now, however, it can be implemented using an R - $2R$ topology and in considerably less space. Table 33.1 shows the various characteristics of resistors available in a submicron CMOS process. In the following discussion we assume that we are concerned with implementing an R - $2R$ DAC (see Chs. 29 and 34).

Silicide	Resistor Type	R_s (ohms/sq) AVG.	TCR1 (ppm/C) AVG.	TCR2 (ppm/C ²) AVG.	VCR1 (ppm/V) AVG.	VCR2 (ppm/V ²) AVG.	Mis-match % $\Delta R/R$
	n-well	500 ± 10	2400 ± 50	7 ± 0.5	$8000 \pm$	500 ± 50	< 0.1
	n+	120 ± 1	21 ± 10	0.6 ± 0.03	700 ± 50	150 ± 15	< 0.5
	p+	300 ± 5	160 ± 10	0.8 ± 0.03	600 ± 50	150 ± 15	< 0.2
	n+ diff	100 ± 2	1500 ± 10	0.04 ± 0.1	2500 ± 50	350 ± 20	< 0.4
	p+ diff	125 ± 3	1400 ± 20	0.4 ± 0.1	80 ± 80	100 ± 25	< 0.6
*	n+	3 ± 0.3	3300 ± 90	1.0 ± 0.2	2500 ± 125	3800 ± 400	< 0.4
*	p+	2 ± 0.1	3600 ± 50	1.0 ± 0.2	2500 ± 400	5500 ± 250	< 0.7
*	n+ diff	3 ± 0.1	3700 ± 50	1.0 ± 0.2	350 ± 150	600 ± 60	< 1.0
*	p+ diff	2.5 ± 0.1	3800 ± 40	1.0 ± 0.2	150 ± 50	800 ± 40	< 1.0

Table 33.1 Properties of resistors in a submicron CMOS process.

At first glance after reviewing Table 33.1, it may appear as though the n-well offers the best choice for a resistor in a data converter since it has < 0.1% matching characteristic. However, after reviewing the voltage coefficient specification for the n-well, i.e., $0.008/V$, we see a problem. If the voltage across one resistor in the R - $2R$ string is 2 V, while the voltage across another resistor is 0 V, we will see a mismatch between the two resistors of 1.6%. Such a large voltage-varying mismatch can cause severe linearity problems. It should be mentioned in passing that the origin of the n-well voltage coefficient comes from the extension of the depletion region into the n-type material used in the n-well (a problem not found in polysilicon resistors).

The polysilicon resistors available depend on the process steps. In one scenario the poly is doped *in situ* while it's being deposited. Since the poly is silicided (called a polycide) the possible pn-junction between the p+ and n+ poly isn't a concern. If the poly is doped with an implant after it has been deposited, then a nitride layer above the gate oxide is required to keep the implant from penetrating into the MOSFET's channel. In either case, a silicide blocking mask is generally available to block out the siliciding of poly (or active.)

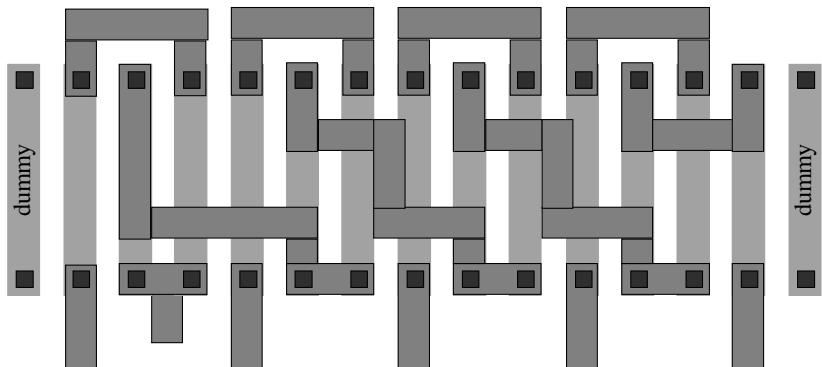
The p+ and n+ diffusions (the source and drain areas) can also be used for resistor implementation. Again, the silicide block can be used to keep from siliciding the diffusions (a silicided diffusion is called a salicide). Since the matching characteristics, temperature behavior, and voltage coefficient are, overall, better for the polysilicon resistors, they are generally preferred in the implementation of precision circuits such as data converters.

In general, the resistor's width and length should be at least 10 and 100 times the minimum feature size of the process, respectively. For example, if L_{min} is 0.15 μm , then the minimum width of the resistor should be 1.5 μm . Requiring minimum widths and lengths for the resistors is important both for matching and to ensure that the self-heating, which occurs because of the different current densities flowing in the R - $2R$ resistors, doesn't cause any noticeable differences in DAC linearity. In simple terms, the larger resistor area dissipates heat better than the same valued resistor in a smaller area.

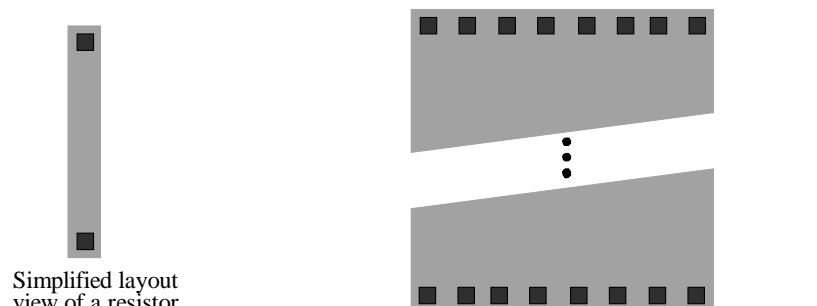
Figure 33.13a shows the conceptual layout of an R - $2R$ resistor string in a minimum area. Figure 33.13b shows the actual layout of the resistors having large width and length along with a large number of contacts to reduce metal/resistive material contact resistance. Figure 33.13c shows the problem of laying out metal over the resistive material, that is, resistor *conductivity modulation*. The figure shows what happens when a metal, having a potential greater than the potential of the underlying resistor is laid out directly over a resistor. Electrons are attracted towards the surface of the resistor causing spots of lower resistivity. The solutions to avoiding or reducing conductivity modulation are (1) avoiding running metal over the resistors, (2) using higher levels of metal to route the resistive signals so as to increase the distance between the resistor and the overlaying metal (remembering vias and contacts must be plentiful to avoid adding unwanted series resistance), or (3) inserting a conducting "shield" connected to analog ground and made with metal1 between the resistors and the routing wires above the R - $2R$ resistor array.

Finally, to conclude this subsection, we ask, "What is the best method of laying out the resistors in an R - $2R$ string to avoid process gradients and achieve good matching?" While there are no absolute answers, we will discuss a possibility where layout area is a concern. In other words, we won't discuss methods that use a large amount of layout area to average out process variations but will limit our averaging to at most twice the layout area of the R - $2R$ string shown in Fig. 33.13.

Figure 33.14 shows one possibility for averaging process gradients using a common-centroid configuration (see Ch. 7) with two R - $2R$ strings connected in series. In this figure we are assuming that the process variations change linearly with position. For example, the first resistor in the string may have an effective value of 1k, while the second's value may be 1.01k, and the third's value is 1.02k, etc. The normalized change in the resistance value is shown in the figure using numbers. However, we could show that the process gradients average out no matter what numbers are used, when using this layout topology, as long as the sheet resistance varies linearly with position. For example, the MSB $2R$ in the top string of Fig. 33.14 (on the left) has a value of 14 (6 + 8). The MSB $2R$ in the bottom string (on the right) has a value of 24. Adding the values of the two resistors, by connecting them in series, results in a resistor value of 38 ($2R = 38$ while $R = 19$). The middle resistor value in the top string has a value of 12, while the bottom

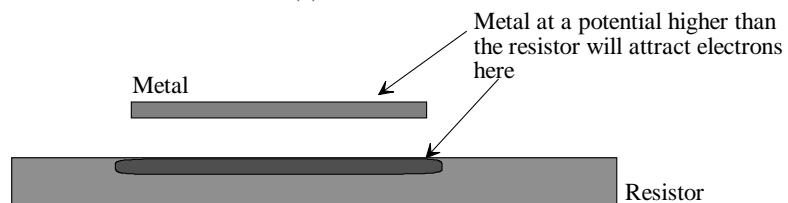


(a)



Layout of actual resistor with large width and length for better matching and power dissipation.

(b)



Cross sectional view of metal over resistor.

(c)

Figure 33.13 (a) Minimal layout of R-2R string, (b) actual layout of resistor, and (c) conductivity modulation of the resistor value.

resistor has a value of 7. Again, adding the two resistors results in a value of 19. Fundamentally, the limiting factor in matching then becomes the voltage and temperature (because of the different current densities through the resistors) coefficients of the resistors and the finite resistance of the MOSFET switches used in the DAC (discussed in more detail in the following chapter).

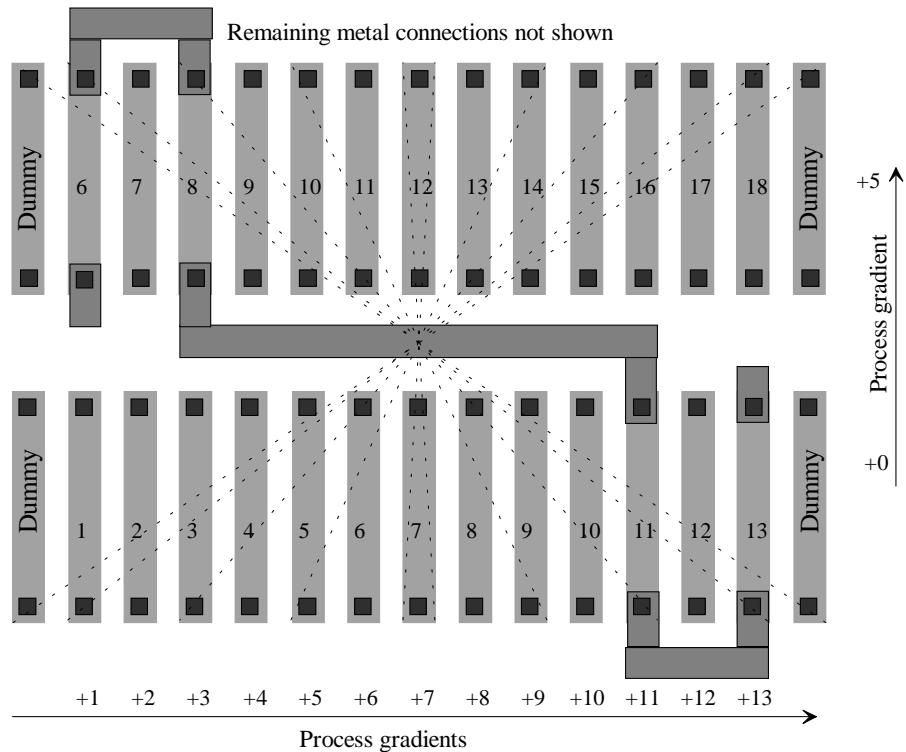


Figure 33.14 Two-string layout for improving matching of R-2Rs. Assumes the resistors are connected together on higher levels of metal to avoid conductivity modulation.

33.1.3 SPICE MOSFET Modeling

In this section we discuss using the EKV MOSFET model [4] in SPICE mixed-signal simulations. We present a basic overview of the model and why we have selected it for our example simulations.

Model Selection

In general, selecting a SPICE MOSFET model for submicron CMOS circuit design begins with comparing simulated DC curves against measured curves, for example, I_D vs. V_{DS} or

I_D vs V_{GS} . While this is an important concern, and any SPICE model used for simulating submicron circuits should show good agreement, it won't be what we focus on here. Here we focus on the ability of the model to transition continuously from weak to strong inversion. While we might think that looking at the DC curves of a device (e.g., I_D vs. V_{GS}) would show the discontinuities (kinks) between weak and strong inversion, a much better indication is to look at several devices operating under similar, related conditions.

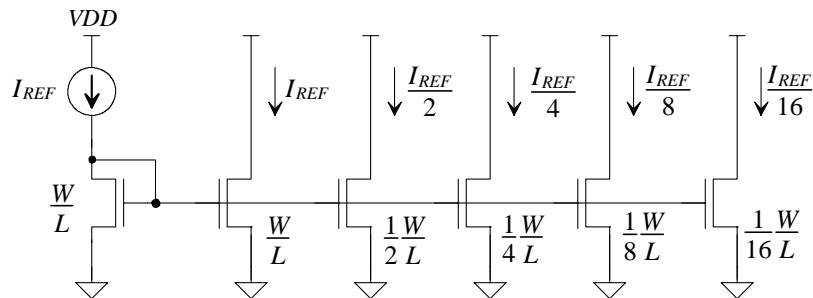


Figure 33.15 Binary weighted current mirror.

Consider the binary weighted current mirror shown in Fig. 33.15. In the following discussion we assume the lengths and widths of the devices are so large that oxide encroachment and lateral diffusion are not an issue. Clearly, in Fig. 33.15, the MOSFETs will all be operating in the same region, for example, strong inversion. What we are going to do, with the binary weighted current mirror, is utilize the fact that MOSFETs in series and parallel can be combined, as seen in Fig. 33.16, to implement a test circuit to evaluate the performance of our submicron SPICE model. Note that our test circuit will have nothing to do, directly, with short-channel behavior, but rather it will evaluate the fundamental implementation of the model.

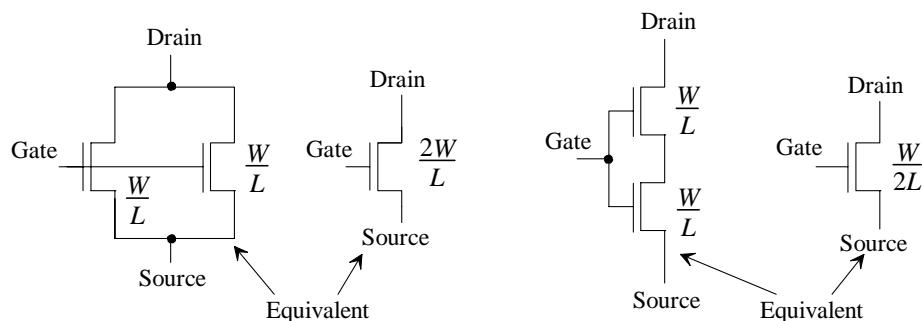


Figure 33.16 Combining parallel and series MOSFETs.

Figure 33.17 shows the implementation of the binary-weighted current mirror using a *W-2W* topology (based on the *R-2R* divider discussed in the last section). To understand the operation of this circuit first note that currents in M1 and M2 have the same value. Next, notice that the currents flowing in the remaining MOSFETs should sum to I_{REF} . In fact, using the information shown in Fig. 33.16, the remaining MOSFETs can be combined into a MOSFET with the same size as M1 or M2 (and, again, having a drain current of I_{REF}). Each stage we add to this topology essentially divides the reference current by two keeping the overall sum of the currents at I_{REF} .

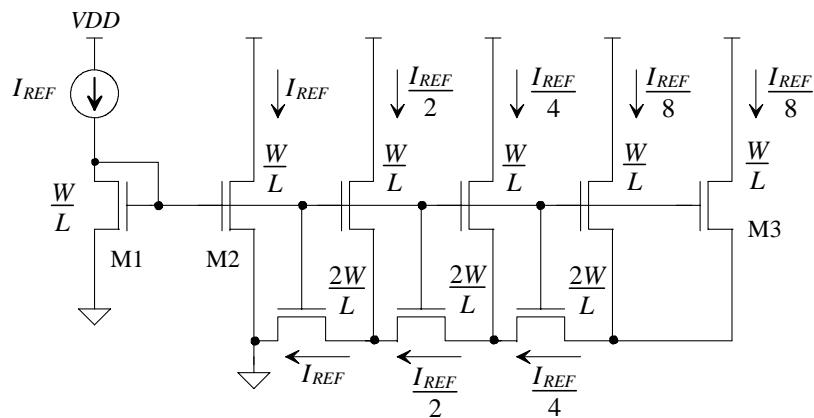


Figure 33.17 W-2W current mirror.

This circuit is useful because it now relates the current flowing in a strongly inverted device, say M1 or M2 in Fig. 33.17, to a weakly inverted device, say M3. If the MOSFET model is operating with a truly continuous change from one region to another the currents will be binary-related and sum to I_{REF} . Figure 33.18 shows the simulation results for an eight-stage *W-2W* current mirror modeled with the EKV model. The 1 to 2% error in the binary currents is related to the differing drain-to-source voltages, V_{DS} , of the devices. The minimum length of the device modeled by the EKV model in this simulation is 0.15 μm while the actual length used in the simulation is 5 μm (33 times minimum). The widths of the devices used in the simulation are 20 μm and 40 μm .

Figure 33.19 shows the simulation results using a MOSFET model that doesn't model these transitions well. If one were to use this model where the *W-2W* section is used in a DAC, the designer might think the DAC performance is circuit-limited and not matching-limited (keeping in mind that all MOSFETs are perfectly matched in a SPICE simulation). This also points out an important point: *Simulations don't always tell the truth! The good design engineer knows the limitations of the models and the simulator.*

While there are other reasons for using the EKV model (simulation speed, scaling, well-behaved, etc.), the topic of MOSFET modeling is outside the scope of this book [5].

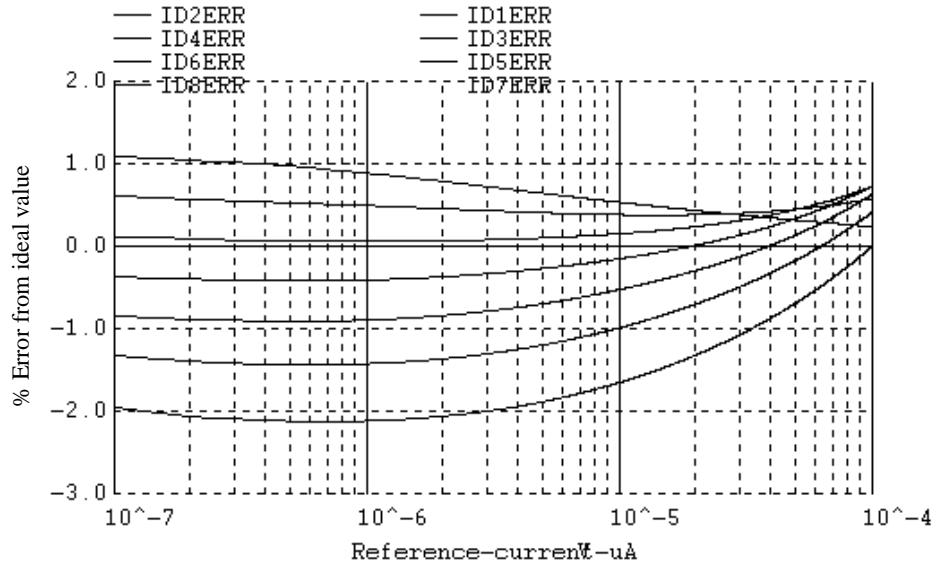


Figure 33.18 Simulated W-2W current mirror using EKV model.

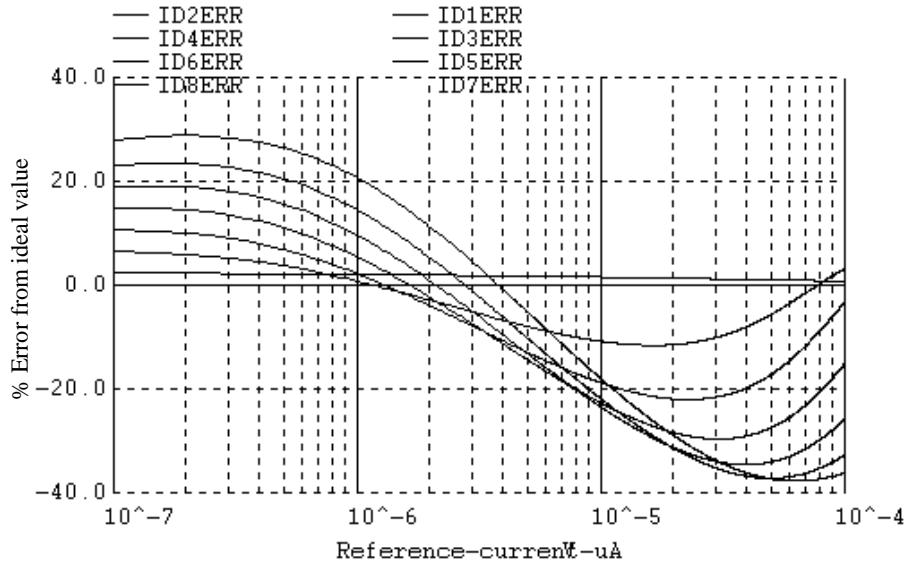


Figure 33.19 Simulated W-2W current mirror showing model problems.

Model Parameters

The EKV model parameters are listed below for both NMOS and PMOS devices. Notice that the minimum length is 0.15 μm and the minimum width is 1.05 μm . Also note that the level used depends on the simulator. For the simulations in this book, again, we will utilize WinSPICE. While some of the model names are discussed in Ch. 5, information concerning the remaining names can be found in [4]. Also note that these models are located in the file models.txt in the zip file chap33_spice.zip located at cmosedu.com.

```
*** SPICE Models

*** Models created by Daniel Foty.
*** (c) 2001, Gilgamesh Associates and EPFL - All rights reserved.
*** These models are provided without warranty or support.
*** These models represent a completely fictitious 0.15um process, and do
*** NOT correspond to any real silicon process. They are provided expressly for
*** use in the examples provided in this text, and should not be used for any
*** real silicon product design.

*** NMOS EKV MOSFET Model ****
*** Level=44 in WinSPICE and ELDO, Level=55 in ADM/HSPICE, Level=5 in PSPICE,
*** Level=EKV in Spectre
*** Lmin=0.15u Wmin=1.05u (If Scale=0.15u then Lmin=1 and Wmin=7)
*-----
.MODEL nmos nmos
+ LEVEL=44

*** Setup Parameters
+ UPDATE=2.6

*** Process Related Model Parameters
+ COX=9.083E-3 XJ=0.15E-6

*** Intrinsic Model Parameters
+ VTO=0.4 GAMMA=0.71 PHI=0.97 KP=453E-6
+ E0=88.0E6 UCRIT=4.0E6
+ DL=-0.05E-6 DW=-0.02E-6
+ LAMBDA = 0.30 LETA=0.28 WETA=0
+ Q0=280E-6 LK=0.5E-6

*** Substrate Current Parameters
+ IBN=1.0 IBA=200E6 IBB=350E6

*** Intrinsic Model Temperature Parameters
+ TNOM=27.0 TCV=1.5E-3 BEX=-1.5 UCEX=1.7 IBBT=0

*** 1/f Noise Model Parameters
+ KF=1E-27 AF=1

*** Series Resistance and Area Calculation Parameters
+ HDIF=0.24e-6 ACM=3 RSH=5.0 RS=1250.526
+ RD=1250.526 LDIF=0.07e-6
```

```
*** Junction Current Parameters
+ JS=1.0E-6 JSW=5.0E-11 XTI=0 N=1.5

*** Junction Capacitances Parameters
+ CJ=1.0E-3 CJSW=2.0E-10 CJGATE=5.0E-10
+ MJ=0.5 MJSW=0.3 PB=0.9 PBSW=0.9 FC=0.5

*** Gate Overlap Capacitances
+ CGSO=3.0E-10 CGDO=3.0E-10 CGBO=3.0E-11

*** PMOS EKV MOSFET Model ****
*** Level=44 in WinSPICE and ELD0, Level=55 in ADM/HSPICE, Level=5 in PSPICE,
*** Level=EKV in Spectre
*** Lmin=0.15u Wmin=1.05u (If Scale=0.15u then Lmin=1 and Wmin=7)
*-----
.MODEL pmos pmos
+ LEVEL = 44

*** Setup Parameters
+ UPDATE = 2.6

*** Process Related Model Parameters
+ COX=9.083E-3 XJ=0.15E-6

*** Intrinsic Model Parameters
+ VTO=-0.4 GAMMA=0.69 PHI=0.87 KP=92.15E-6
+ E0=51.0E6 UCRIT=18.0E6
+ DL=-0.05E-6 DW=-0.03E-6
+ LAMBDA=1.1 LETA=0.45 WETA=0
+ Q0=200E-6 LK=0.6E-6

*** Substrate Current Parameters
+ IBN=1.0 IBA=0.0 IBB=300E6

*** Intrinsic Model Temperature Parameters
+ TNOM=25.0 TCV=-1.4E-3 BEX=-1.4 UCEX=2.0 IBBT=0.0

*** 1/f Noise Model Parameters
+ KF=1.0E-28 AF=1

*** Series Resistance and Area Calculation Parameters
+ HDIF=0.24E-6 ACM=3 RSH=5.0 RS=3145.263
+ RD=3145.263 LDIF=0.07e-6

*** Junction Current Parameters
+ JS=1.0E-7 JSW=5.0E-12 XTI=0 N=1.8

*** Junction Capacitances Parameters
+ CJ=1.3E-3 CJSW=2.5E-10 CJGATE=5.5E-10
+ MJ=0.5 MJSW=0.35 PB=0.9 PBSW=0.9 FC=0.5

*** Gate Overlap Capacitances
+ CGSO=3.2E-10 CGDO=3.2E-10 CGBO=3.0E-11
```

An Important Note

We will be using the "scale" option available in WinSPICE with the EKV model. This option is added to a netlist as follows

```
.option scale=0.15u
```

A MOSFET specified by

```
M1 1 2 3 4 NMOS L=1 W=10
```

would indicate that the MOSFET has a length of 0.15 μm and a width of 1.5 μm . A possible mistake, when writing a netlist manually, would be to forget to add the scale parameter.

Figures 33.20 and 33.21 show example IV plots for both NMOS and PMOS devices. The NMOS device is sized 10/1 (actual width 1.5 μm and length 0.15 μm). The PMOS device is sized twice as large as the NMOS, that is, 20/1 so that its current levels are similar to the NMOS device. In our example process used in the book VDD is 1.5 V.

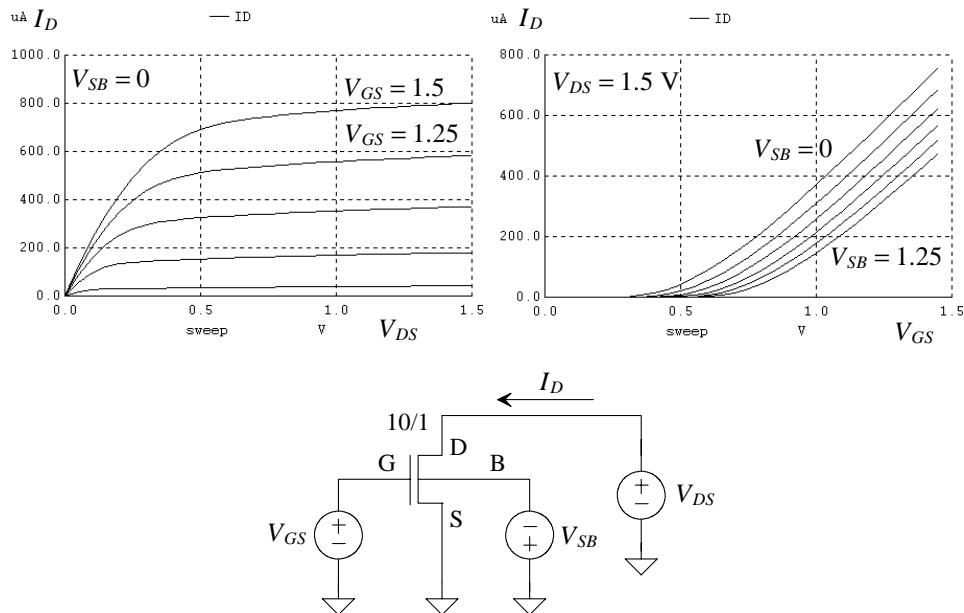


Figure 33.20 NMOS curves for $L = 1$ and $W=10$.

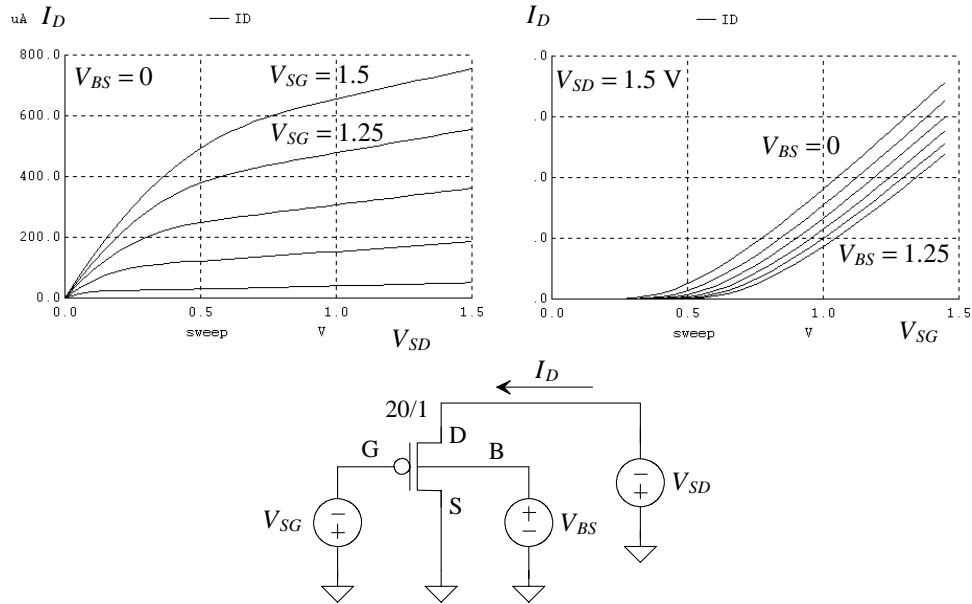


Figure 33.21 PMOS curves for $L = 1$ and $W=20$.

A Note Concerning "Long L MOSFETs"

We might think that with the terminology "long L" simply by making the MOSFET's length very long we can avoid short-channel effects. Further, we might believe that using a long length would result in a device that follows the square-law model. Unfortunately, though, as the process dimensions shrink the devices are designed to drop large voltages over small distances. A 5 μm length device in a 0.15 μm process will have significantly different characteristics than a 5 μm device in a 2 μm process. Increasing the channel length will not significantly affect the depletion width between the channel and the drain (assuming the MOSFET is in saturation) for a fixed V_{DS} .

33.2 Digital Circuit Design

In this section we discuss digital circuit design using a CMOS submicron process. Our focus will be on the digital building blocks used in the last two chapters, that is, switches, delay elements, counters, and adders. We assume the MOSFET models discussed in the last section accurately model the fictitious submicron process used in the design examples presented in this chapter.

33.2.1 The MOSFET Switch

In this section we assume the reader is familiar with the material concerning the switches discussed in Chs. 10 and 27. Figure 33.22 shows a basic NMOS switch in a test configuration for determining its effective digital switching resistance. From the simulation results shown in Fig. 33.23 we can relate the length and width of a MOSFET to the effective digital resistance in our submicron process using

$$R_n = 10 \text{ k}\Omega \cdot \frac{L}{W} \approx \frac{1 + K_n \cdot (VDD - V_{THN})}{KP_n} \cdot \frac{L}{W} \quad (33.3)$$

where the term $1 + K_n \cdot (VDD - V_{THN})$ (K_n is a constant dependent on the process) is used to model the reduction in the mobility because of the thin oxide used in the MOSFET formation. Notice that this is an average estimate for the resistance in all cases.

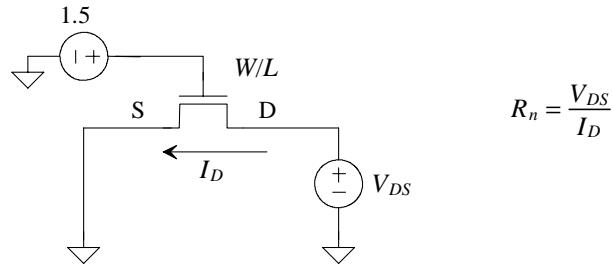


Figure 33.22 Determining resistance of an NMOS switch.

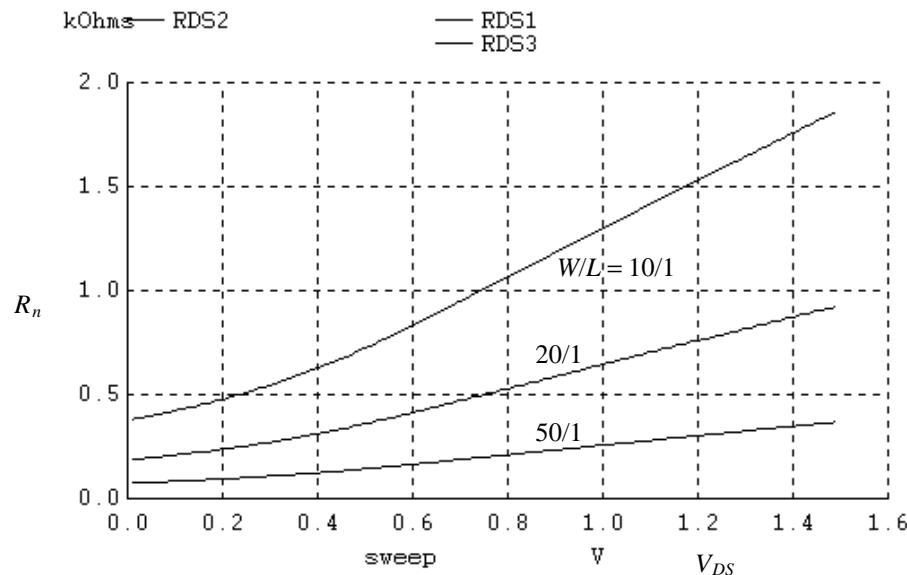


Figure 33.23 NMOS effective resistance from Fig. 33.22.

Example 33.3

Estimate, and verify with a SPICE simulation, the delay time in the following circuit (Fig. 33.24).

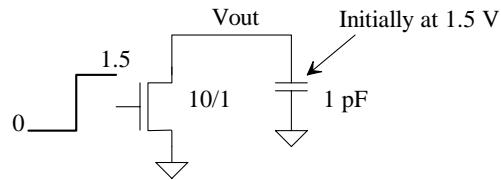


Figure 33.24 Circuit used in Ex. 33.3.

Using Eq. (33.3) the effective digital resistance of the MOSFET is 1k. The propagation delay (input going high and output going low) can then be estimated, knowing the load capacitance is much larger than the MOSFET capacitances, using

$$t_{PHL} = R_n \cdot C_L = 1k \cdot 1 \text{ pF} = 1 \text{ ns}$$

The SPICE simulation results are shown in Fig. 33.25. ■

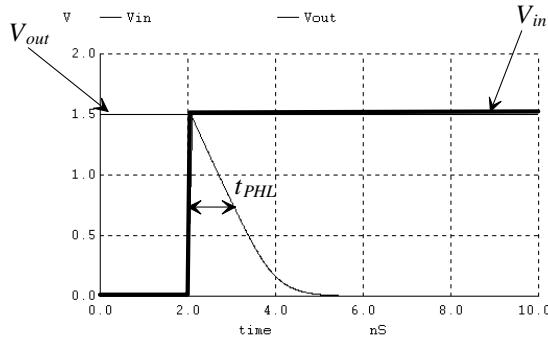


Figure 33.25 Simulation results for Ex. 33.3.

Figure 33.26 shows the test circuit used to determine the PMOS effective digital switching resistance, R_p . From the simulation results shown in Fig. 33.27 we can write

$$R_p = 20 \text{ k}\Omega \cdot \frac{L}{W} \approx \frac{1 + K_p \cdot (VDD - V_{THP})}{K P_p} \cdot \frac{L}{W} \quad (33.4)$$

Again this is an average estimate for the resistance for all source-to-drain voltages. Also note how it doesn't matter if we use actual device sizes or scaled sizes in this equation because of the ratio.

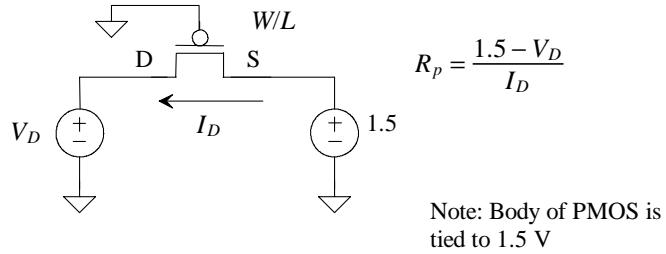


Figure 33.26 Determining resistance of an PMOS switch.

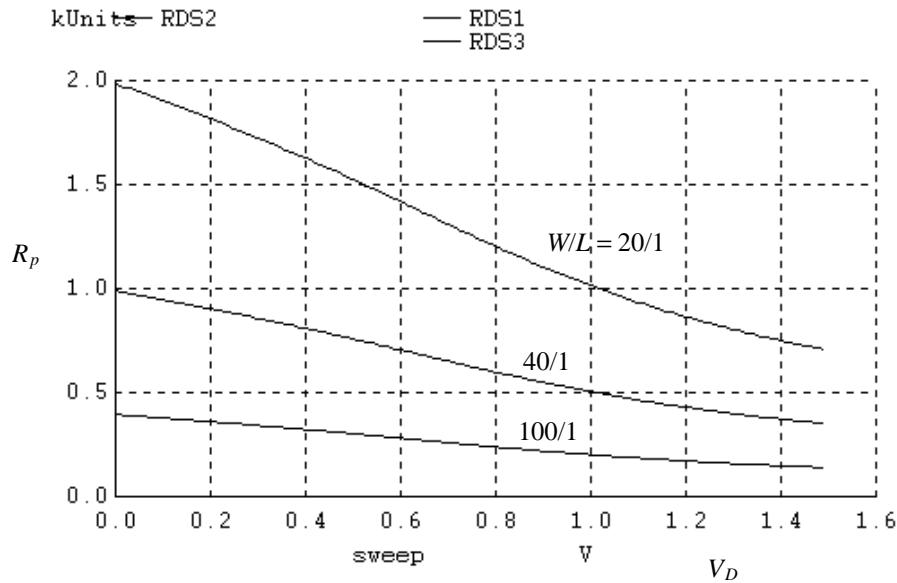


Figure 33.27 PMOS effective resistance from Fig. 33.26.

Bidirectional Switches

The NMOS and PMOS switches shown in Figs. 33.22 and 33.26 make use of the fact that the NMOS source is connected to ground and the PMOS source is connected to VDD . For complementary static CMOS logic design we can rely on Eqs. (33.3) and (33.4) to estimate the MOSFET sizes for a particular drive strength. However, if the MOSFET is used as a pass gate where current can flow bidirectionally, the effective switching resistance can become very large. This is related to the fact that the PMOS switch can't pass a logic low (0 V) well and an NMOS switch can't pass a logic high well (VDD). Of course, combining the NMOS and PMOS into a transmission gate (TG) eliminates this concern at the price of larger layout area and the need for two complementary clocks.

Figure 33.28 shows the NMOS device used as a switch with the input at VDD (= 1.5 V here). Figure 33.29 shows how the effective switching resistance of this device changes with size under various output voltages. Notice that, as we would expect, R_n gets very large as the output approaches $VDD - V_{THN}$ (with body effect).

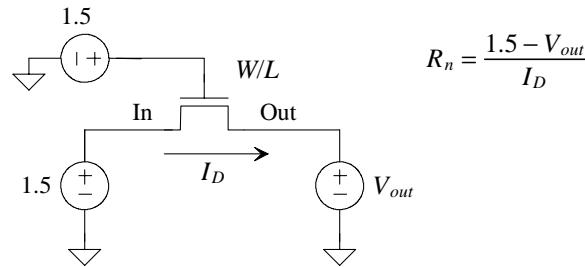


Figure 33.28 Determining resistance of a bidirectional NMOS switch.

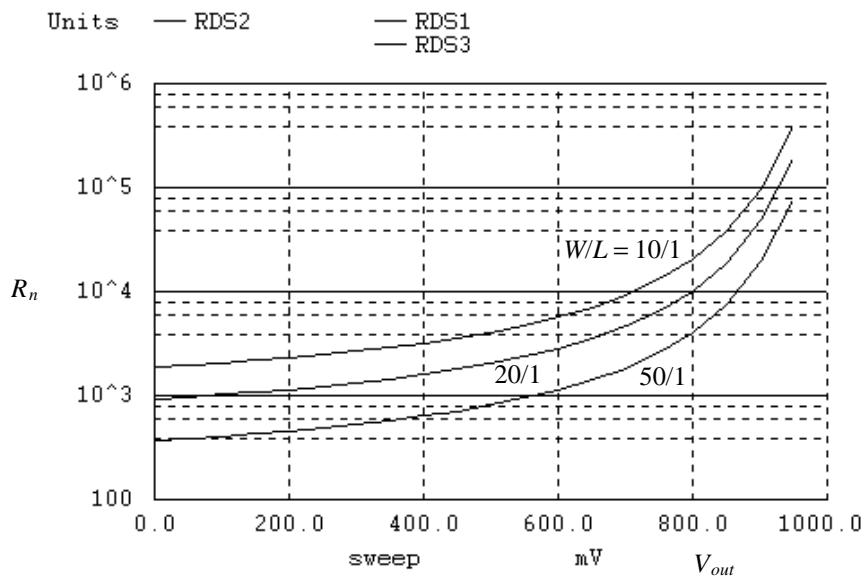


Figure 33.29 NMOS effective resistance from Fig. 33.28.

Another application where a switch can be used bidirectionally was in the DAI discussed in the last chapter. If the switches used in the DAI can be replaced with NMOS devices the implementation is simpler. If we increase the voltage of the gate signal, it is possible to turn the MOSFET all the way on and pass VDD from the switch input to its output. Figure 33.30 shows the results if the gate signal in Fig. 33.28 is increased to a value of 2.3 V.

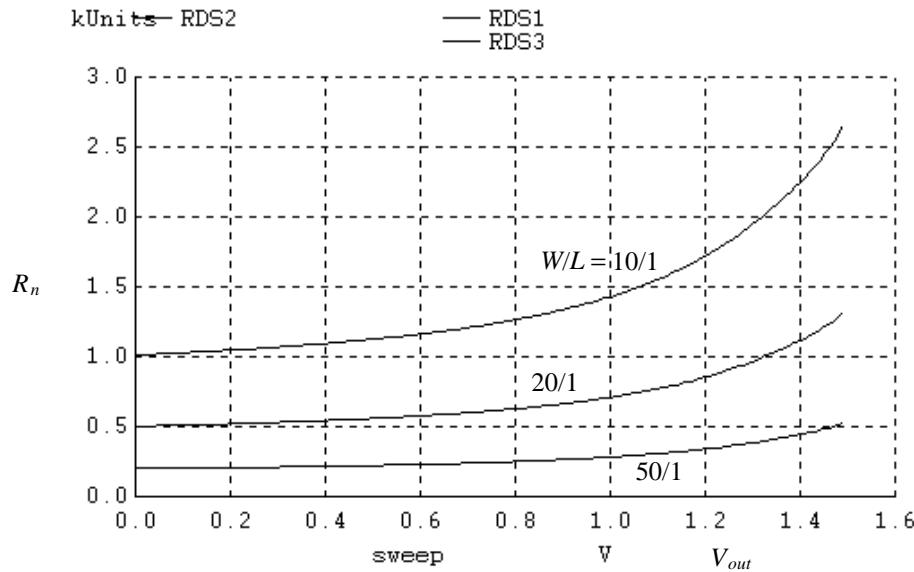


Figure 33.30 NMOS effective resistance from Fig. 33.28 with gate at 2.3 V.

Toward the goal of increasing the amplitude of the clock signal controlling the switches consider the bootstrapped clock driver circuit shown in Fig. 33.31. This circuit is a simple implementation of a voltage (charge) pump discussed in Ch. 18. The output amplitude of the circuit approaches $2VDD$ so concern for oxide damage or long-term failure is warranted as discussed in Ch. 6. The inverters are sized with 100/1 PMOS and

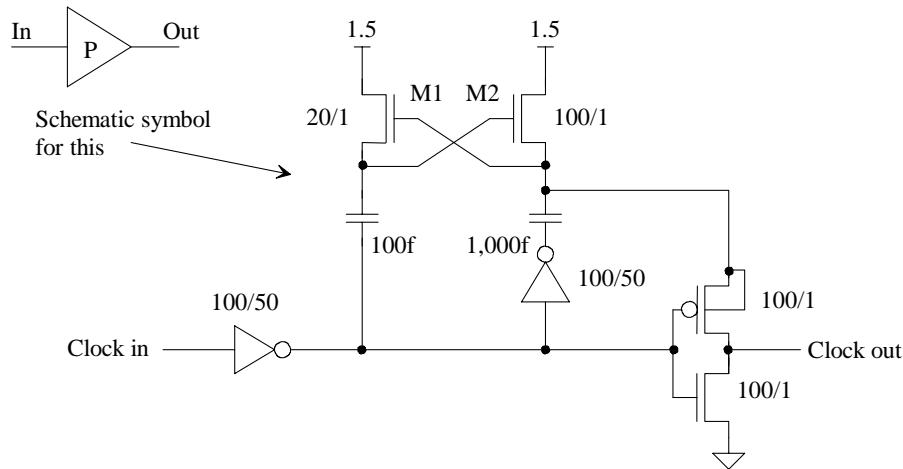


Figure 33.31 Charge-pump clock driver. Peak output is around 2.8 V.

50/1 NMOS devices. The other devices are sized to minimize power while supplying a reasonable level of output drive. For example, the 100 fF capacitor only has to supply charge to the gate of M2, but the 1,000 fF capacitor supplies charge to both the gate of M1 and the load. Further scaling is possible to further reduce power and enhance output drive. Note this circuit is noninverting and can be used in a nonoverlapping clock generator, as shown in Fig. 33.32.

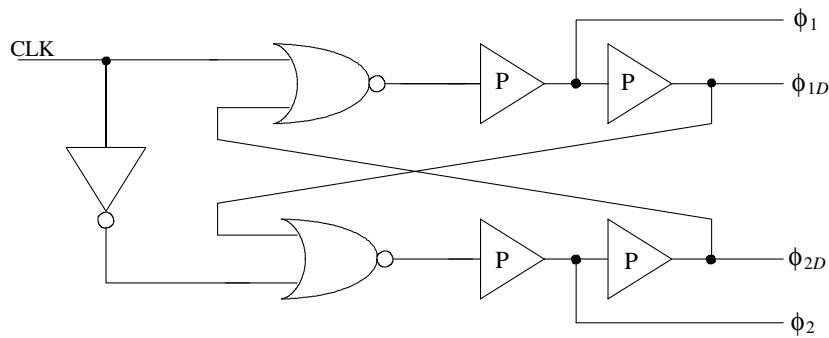


Figure 33.32 Nonoverlapping clock generation circuit with pumped outputs.

The clock generator of Fig. 33.32 provides the two phases of a clock signal as well as slightly delayed clocks for use in a sample-and-hold, see Ch. 27. As the simulation results show in Fig. 33.33, the output amplitude is approximately 2.5 V when a 100 fF load is connected to each phase of the output. The time that all four clock signals are low, the nonoverlap time (dead time), can be increased by increasing the delay in series with the output of the NOR gates. Adding inverter pairs to the outputs of each NOR gate is a common method of increasing the dead-time. Also note that the capacitors in Fig. 33.31 can be implemented using NMOS devices since they will always be in strong inversion.

A Clocked Comparator

One of the circuits that we used often in Ch. 32, in NS data converters, was a clocked comparator. Let's develop a clocked comparator using inverters and switches. Examine the evolution of circuits shown in Fig. 33.34. In parts (a) and (b) the basic inverter-based latch is shown. In order to reduce power and make the comparator clocked, we add the NMOS switch shown in part (c). Before each comparison, we want to ensure the comparator is equilibrated. To erase the memory of the previous comparison, the PMOS switch in part (d) is added to our circuit. When the clock, ϕ , goes high, the PMOS device turns off and the NMOS device turns on allowing the inverters to latch in a stable condition. The only thing we need to add to this circuit is circuitry to somehow create an imbalance across the inverters when ϕ goes high. We can do this several ways. One possibility is shown in Fig. 33.34e.

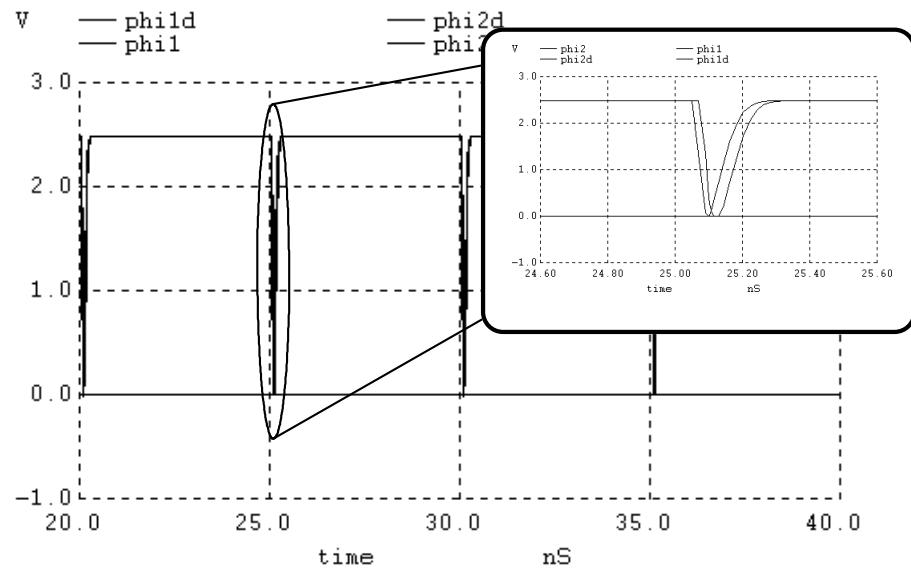


Figure 33.33 Typical output of Fig. 33.32.

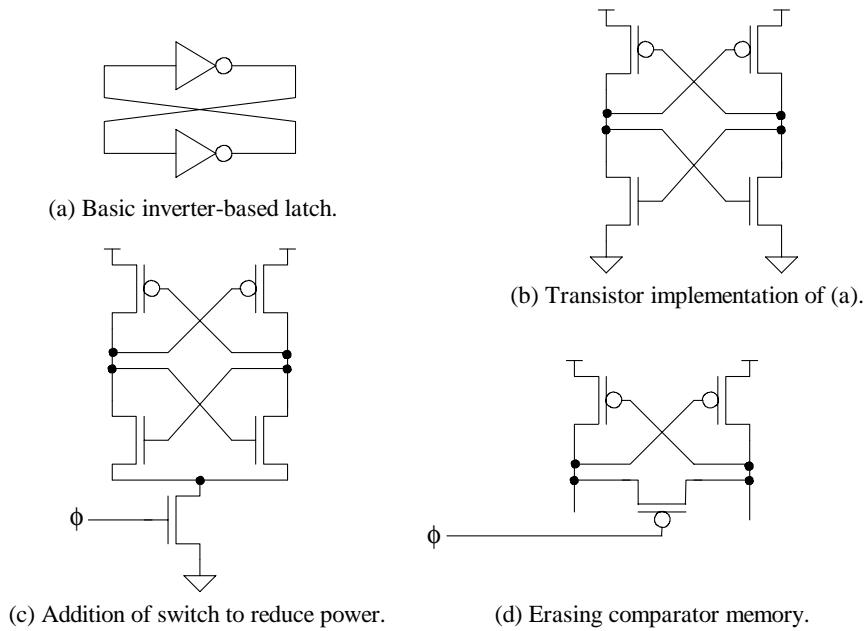
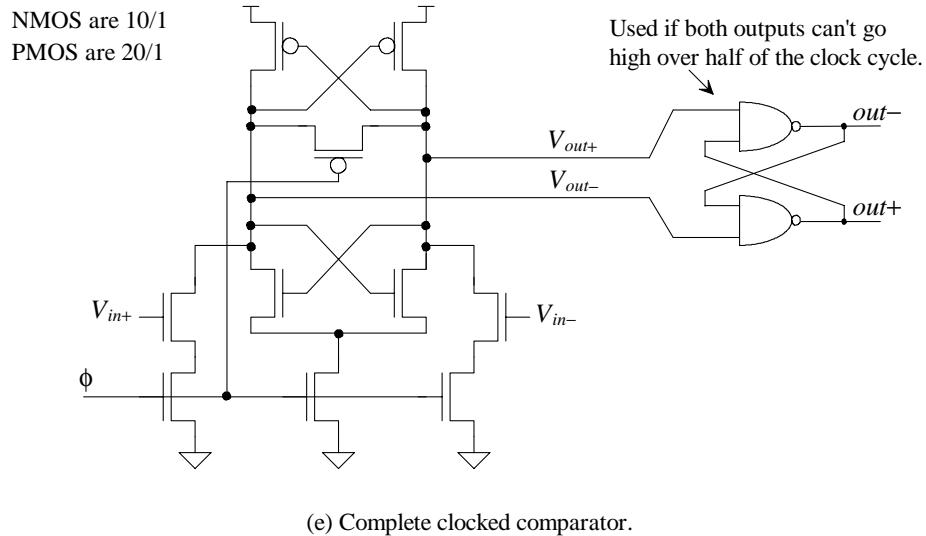


Figure 33.34 Implementation of a clocked comparator.



(e) Complete clocked comparator.

Figure 33.34 (cont'd) Implementation of a clocked comparator.

In Fig. 33.34e we connect our inputs to two common-source amplifiers which have two purposes: (1) to amplify the input signals and create an imbalance in the latch, and (2) to isolate the inputs from the switching noise resulting from the latch positive feedback. (Kickback noise was discussed in Ch. 27.) The two switches we add in series with our common-source amplifiers are not, in all cases, necessary. We add them here to reduce power dissipation when ϕ is low. The clocked comparator doesn't draw any current when ϕ is low. It does, however, draw current when ϕ is high. The amount is dependent on the voltage applied to the comparator's inputs. Note also that when ϕ is low, both V_{out+} and V_{out-} are equilibrated to a voltage of $VDD - V_{THP}$. This means that the output of the comparator is essentially a logic high whenever ϕ is low. In order to make the circuit appear as a truly rising edge comparator, where the comparison from a rising edge is valid until the next rising edge of an SR flip-flop (see Ch. 13) can be added to the output of the clocked circuit.

Figure 33.35 shows simulation results with a typical input waveform. Note how, when ϕ is low, the outputs of the latch go to $VDD - V_{THP}$. Also note how the final outputs remain valid until the next rising edge of the clock. One thing that this simulation doesn't reveal is the comparator's random offsets. Since the comparator is simulated using perfectly matched devices, we will only see the comparator's systematic offset (see Ch. 25). While comparator offset wasn't a concern in a noise-shaping data converter, it is a major concern in a Nyquist-rate data converter. We will revisit comparator and op-amp offsets in Ch. 34.

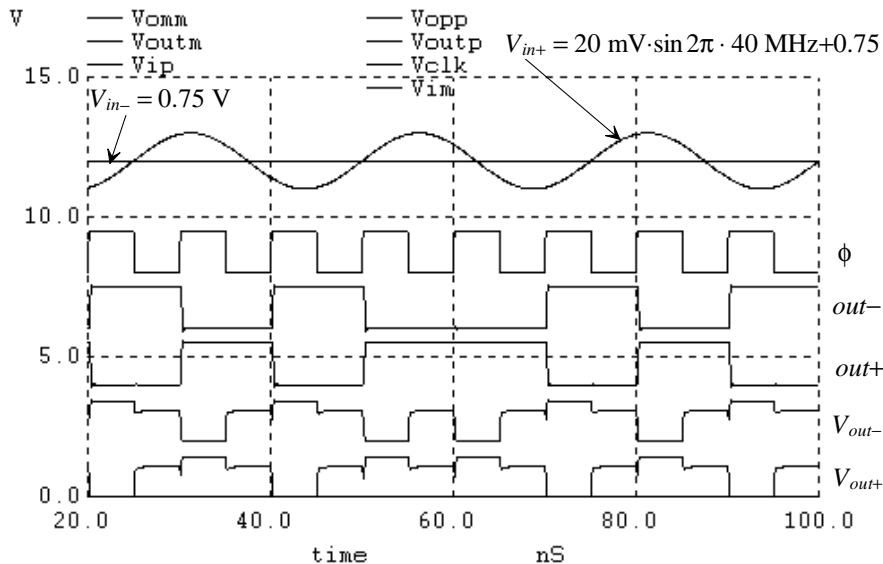


Figure 33.35 Simulation results for the comparator of Fig. 33.34.

One last comment before leaving this topic: Notice, after reviewing the simulation netlist, how we used voltage sources for our comparator inputs. In any practical simulation it is a better idea to add some series resistance between the comparator inputs and the voltage sources to simulate finite source impedance. Resistors with 10k values are typical for simulations of this nature. Using these resistors shows kickback noise on the inputs to the comparator. This noise is *often the fundamental limitation* of the comparator's performance in an actual circuit. Methods to reduce this kickback, such as cascading the input common-source amplifiers or using an additional amplification stage such as a diff-amp, should be used in a general application. Note that kickback noise shouldn't be a problem in the comparator used in a noise-shaping data converter due to the large capacitance connected to the input of the comparator (the DAI's integrating capacitance).

Common-Mode Noise Elimination

High-speed digital signals can often appear more like a sinewave than a square wave. When these sinewave-like signals are applied to the inputs of a digital gate, the timing delay between the gate's inputs and output can vary. This delay variation can be the result of noise coupled onto the wires used to connect the circuits together or because of power-supply fluctuations. To avoid these problems in analog circuits, as discussed on page 191 and in Chs. 25 and 27, fully-differential outputs are used. Using fully differential outputs in a digital system, however, can result in large layout area. The gates must have both differential inputs and outputs. Because of this and the large noise margin of digital signals, generally only signals that propagate over long distances will experience

detrimental noise corruption. However, any digital signal can be corrupted because of power-supply fluctuations. In this section we discuss the idea of common-mode noise elimination, CMNE. The CMNE circuit will, ideally, eliminate common-mode noise on a wire pair while, at the same time, not affect the differential component. The technique can be used to "square-up" digital signals. We apply this technique to the design of a high-speed digital buffer.

Toward the design of a CMNE circuit, examine Fig. 33.36a. This circuit was the heart, or decision circuit, of our comparator in Ch. 26. Assuming for the moment that all transistors are sized equally we know that when i_{o+} is greater than i_{o-} , the output v_{o+} is greater than v_{o-} . Any common signal to both i_{o+} and i_{o-} will not be a factor in which signal, v_{o+} or v_{o-} , is larger. This circuit can only function if both i_{o+} and i_{o-} are positive. Figure 33.36b shows the decision circuit of part (b) where we've added a PMOS decision circuit so that the currents i_{o+} and i_{o-} can be positive or negative. The schematic representation of this circuit using inverters is shown in Fig. 33.36c.

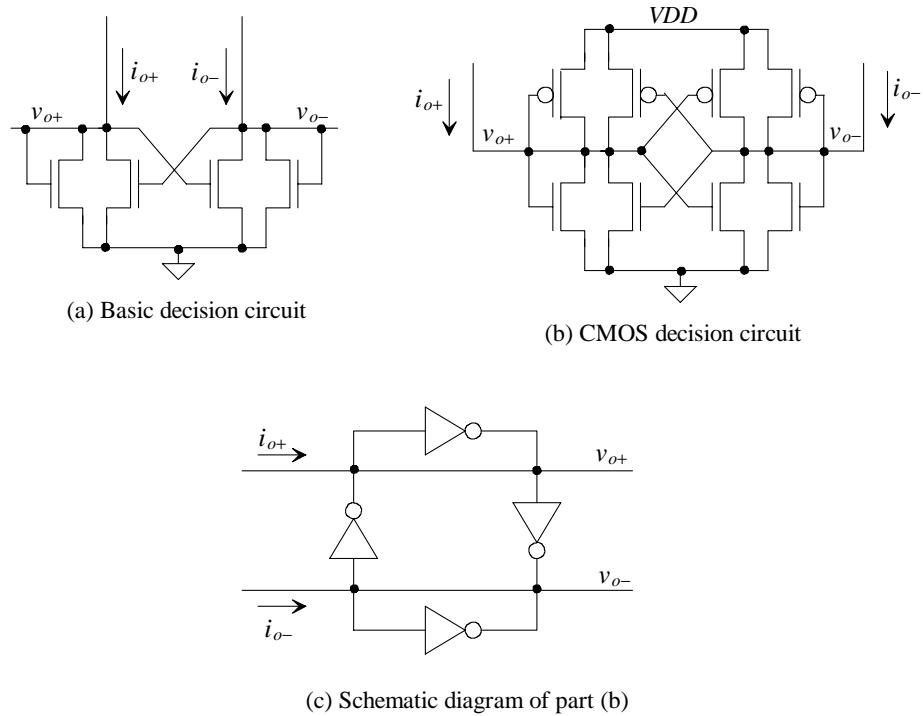


Figure 33.36 Common-mode noise elimination circuit.

To determine quantitatively how this circuit functions, consider the inverter output vs. input plot shown in Fig. 33.37. The switching point, V_{SP} , is defined as the point where the inverter's input and the output voltages are equal. This is also the DC operating point of the CMNE circuit in Fig. 33.36c with zero input current. The inverter's model is also shown in this figure. When v_{IN} increases, v_{OUT} ($= g_m \cdot v_{IN} \cdot r_o$) decreases. The actual values of g_m and r_o are not needed to understand the operation of the CMNE circuit.

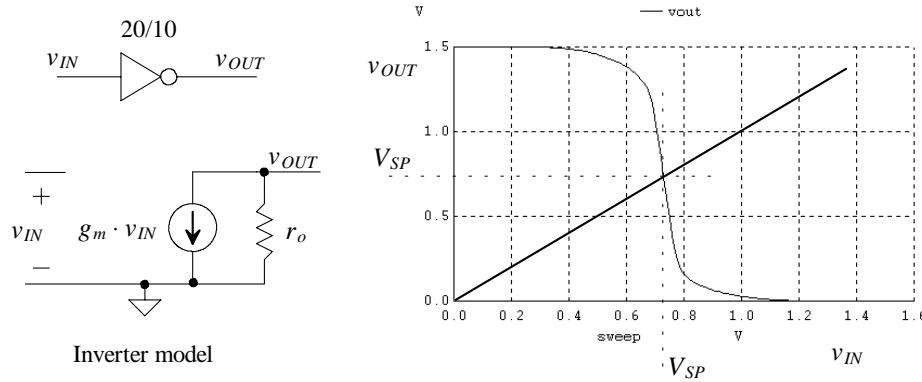


Figure 33.37 Modeling an inverter.

Consider the CMNE circuit shown with the inverter currents in Fig. 33.38. In this circuit the input current is made up of a common-mode component, I_{CM} , and a difference-mode component, i_{diff} , so that

$$i_{o+} = i_{CM} + i_{diff} \quad (33.5)$$

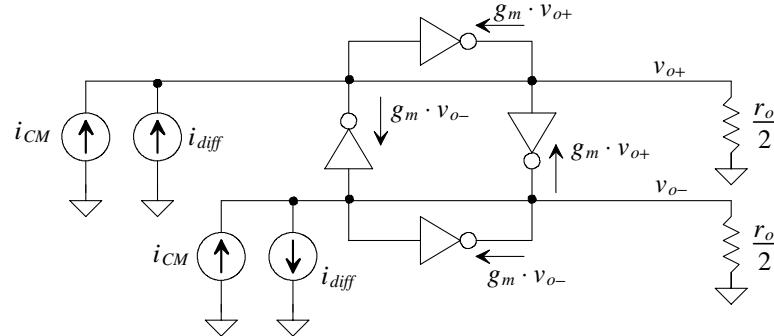


Figure 33.38 Analysis of a CMNE circuit.

and

$$i_{o-} = i_{CM} - i_{diff} \quad (33.6)$$

The resistors with values $r_o/2$ connected to the two signal nodes model the parallel combination of the two inverters' output resistance at each node. The fact that there are only two signal nodes in this circuit is an important point since we don't want the CMNE circuit to affect the desired, fully-differential signals. We want the desired signal to pass through the CMNE circuit without any delay. If we sum the currents at each node, we get

$$i_{diff} = g_m v_{o-} + g_m v_{o+} + \frac{v_{o+}}{r_o/2} + i_{CM} \quad (33.7)$$

and

$$-i_{diff} = g_m v_{o-} + g_m v_{o+} + \frac{v_{o-}}{r_o/2} + i_{CM} \quad (33.8)$$

Taking the difference in these two equations results in

$$i_{diff} = \frac{v_{o+} - v_{o-}}{r_o} \quad (33.9)$$

If we write the outputs as

$$v_{o+} = v_{cm} + v_{diff} \quad (33.10)$$

and

$$v_{o-} = v_{cm} - v_{diff} \quad (33.11)$$

we notice the common-mode component subtracts leaving

$$v_{diff} = i_{diff} \cdot \frac{r_o}{2} \quad (33.12)$$

This equation shows that the input signal, i_{diff} , generates an output voltage, v_{diff} , and that any common-mode noise, v_{cm} , is removed from the output signal.

A simple, yet useful, application of CMNE is shown in Fig. 33.39. Here the CMNE circuit is placed in between two inverters. This circuit essentially behaves like a diff-amp, or a comparator, amplifying the difference in the input signals and rejecting the common-mode component. To be more correct, the common-mode component is shorted to an inverter switching-point voltage, V_{SP} , so that the difference-mode signals swing around V_{SP} . The inverters added at the end of the circuit ensure that good, solid CMOS logic levels are output from the circuit.

Figure 33.40 shows a plot of the buffer's output against its input for reference voltages, V_{REF} , of 0.7 and 0.9 V. As we would expect, the buffer switches states only after the input exceeds the reference voltage. This buffer can also be used if the input signals are complementary logic signals. The delay through the buffer is only two inverter delays.

Because of the two inverters with their inputs shorted to their outputs the power dissipation in this basic example can be excessive (approximately 1 mA for the example given here). To reduce the current draw the CMNE circuit can be scaled.

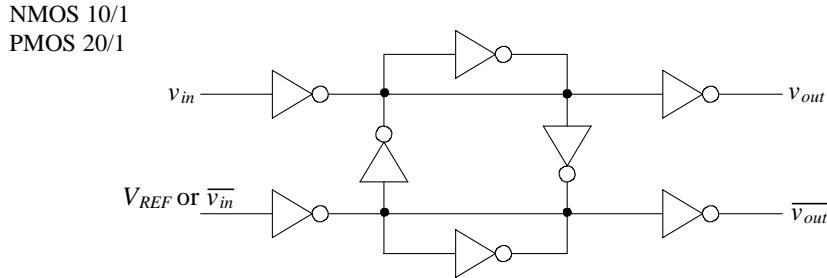


Figure 33.39 A digital buffer (comparator) using CMNE.

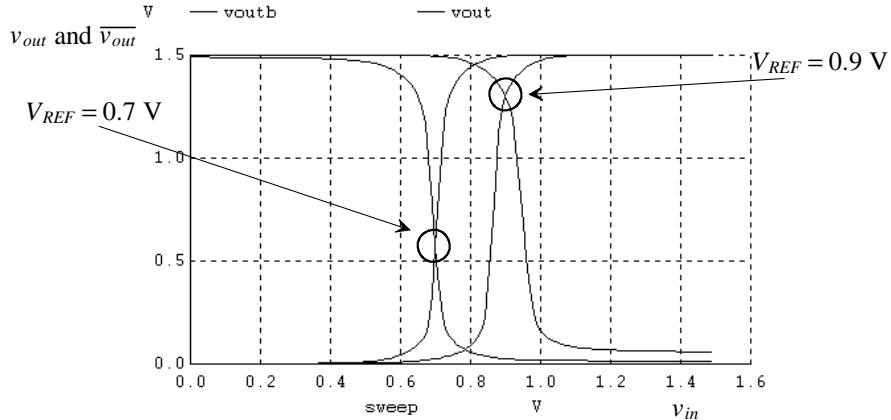


Figure 33.40 Simulating the circuit in Fig. 33.39 showing switching points for references of 0.7 and 0.9 V.

One might wonder, after reviewing Fig. 33.40, if it's possible to increase the gain of the buffer by adjusting the sizes of the inverters. We see that it takes over 300 mV input signal to cause a full logic transition on the buffer's output. Also, it would be nice if the buffer could be designed to have hysteresis, as discussed in Ch. 26. While we can increase the length of the devices used in the inverters to both increase the gain and decrease the power dissipation, the trade-offs are decreases in speed (the delay through the buffer) and frequency response of the CMNE circuit. At very high frequencies the inverters aren't fast enough to eliminate common-mode noise.

After reviewing how we introduced hysteresis into our decision circuit back in Ch. 26, we see that by increasing the length of the MOSFETs used in the shorted input/output inverter we can achieve the same results in the buffer of Fig. 33.39. Consider Fig. 33.41, a redrawn version of Fig. 33.38, with weak inverters. By weak we mean that the W/L ratio

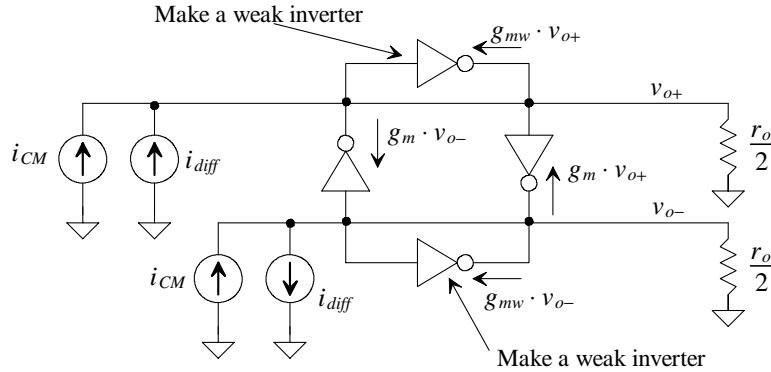


Figure 33.41 Figure 33.38 redrawn with weak inverters for hysteresis.

of the transistors used in this inverter is smaller than the other W/Ls used in the circuit. We indicate that the modified inverter's g_m is now weak, and thus less than the other inverter's g_m , by relabeling it g_{mw} . We can rewrite Eq. (33.9) as

$$i_{diff} = \frac{v_{o+} - v_{o-}}{r_o} + (g_{mw} - g_m) \cdot \frac{v_{o+} - v_{o-}}{2} \quad (33.13)$$

or

$$i_{diff} = \frac{2v_{diff}}{(v_{o+} - v_{o-})} \cdot \left[\frac{1}{r_o} + \frac{1}{2/(g_{mw} - g_m)} \right] \quad (33.14)$$

This equation can be written as

$$v_{diff} = i_{diff} \cdot \left(\frac{r_o}{2} \parallel \frac{1}{g_{mw} - g_m} \right) \quad (33.15)$$

Since g_{mw} is less than g_m , the effective resistor added in parallel with r_o is negative having the effect of increasing the resistive loading of the CMNE circuit. This increase has the effect of boosting the differential gain of the buffer of Fig. 33.39 and introducing hysteresis into the switching point. Note that if g_{mw} is equal to g_m , then Eq. (33.15) reduces to Eq. (33.12).

The allowable range of V_{REF} is set by the threshold voltages and the power supplies. For example, V_{REF} can be no larger than $VDD - V_{THP}$. In reality V_{REF} is limited to voltages less than this because the drive of the inverter connected to V_{REF} decreases. Notice, in Fig. 33.40, how the final output crossover point is creeping toward VDD .

The CMNE technique can be used in a variety of places including delay-locked loops or input and output buffers or simply to balance two differential digital signals. While we used CMOS inverters here the technique can be extended to use inverting amplifiers. We will extend this technique to balancing the outputs of fully-differential op-amps later in the chapter.

Example 33.4

Resimulate the buffer shown in Fig. 33.39 if the length of the MOSFETs is increased to 2. Rerun the simulation if weak inverters, Fig. 33.41, are used with their Ls increased from 2 to 2.1.

Figure 33.42 shows the simulation results. In part (a) the increase in gain with the increase in L from 1 to 2 is evident. The current drawn from VDD decreases from 1 mA to 0.5 mA. Note in this figure we show $V_{REF} = 0.5, 0.7$, and 0.9 V. The references at 0.5 and 0.9 are at the edge of the allowable reference voltages, V_{REF} . In part (b) the small increase in the weak inverter's length (only 5%) shows that the gain is improved further. This small increase introduces little hysteresis into the input buffer. Increasing the weak inverter's length further, say to 3, causes both the gain and hysteresis to further increase. ■

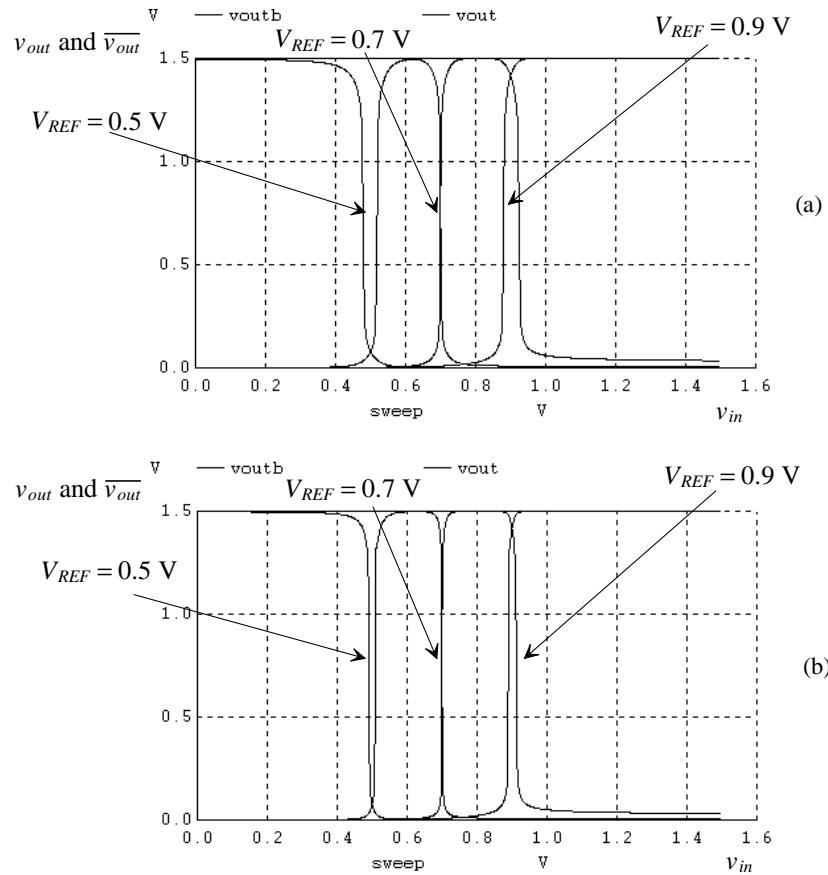


Figure 33.42 Simulation results for the buffers described in Ex. 33.4.

33.2.2 Delay Elements

Delay elements were used to implement our digital averaging or comb filters in Ch. 31. We used these filters on the output of our noise-shaping, modulator-based ADCs or on the input of a NS DAC in Ch. 32. Since the delay elements are continuously clocked, we discuss the use of dynamic circuits in this section. Dynamic elements are used because they result in lower power and smaller-layout area than static CMOS circuits. Low power and small-layout area become extremely important when we realize that a filter may employ hundreds, or even thousands, of delays.

Figure 33.43 shows a basic cascade of pass transistors and inverters first shown and discussed in Ch. 14. When clk goes high, the output of the master is transferred to the slave. When clock goes back low, the input capacitance of the slave inverter remains charged and the output is unchanged. We can think of this circuit as a rising-edge triggered D flip-flop. The pass transistors don't pass a logic high well, so the input voltage to the inverters will be at most $VDD - V_{THN}$ (with body effect). This can result in the inverter having a significant crossover current and large power dissipation. The average current pulled from VDD by the circuit of Fig. 33.43 while clocked at 100 MHz and having equally sized devices (both 10/1) is 50 μ A. To avoid excess current draw, the switching point of the inverter can be decreased toward ground by making the PMOS device weak. The output rise time, with a weak PMOS device, can become very long.

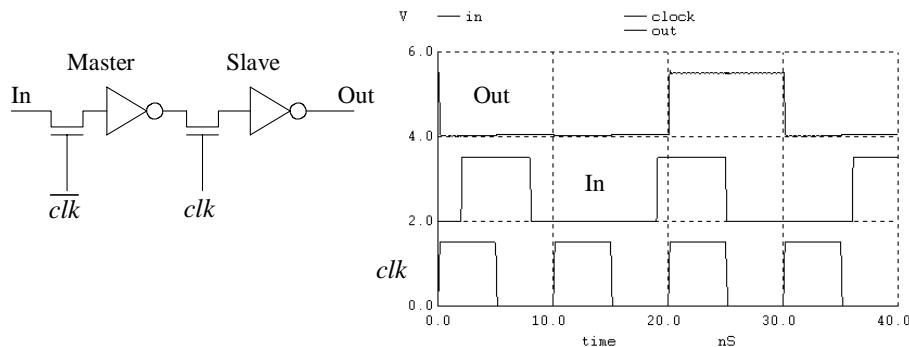


Figure 33.43 Simple delay element using pass transistors and CMOS inverters.

Figure 33.44 shows a positive edge-triggered delay using clocked-CMOS logic. The problem of having excess power-supply current because of marginal logic levels is absent in this configuration. However, the layout area will be larger than the delay in Fig. 33.43. When the delay of Fig. 33.44 is clocked at 100 MHz, the average current pulled from VDD is well below 10 μ A. It is easy to implement a reset using this delay cell because we have a high-impedance node, that is, the node connecting the master and the slave is never connected directly to ground or VDD . We can implement a reset, or set, in the circuit in Fig. 33.43 by adding an additional switch in series with the input and two additional switches on both inverters' inputs.

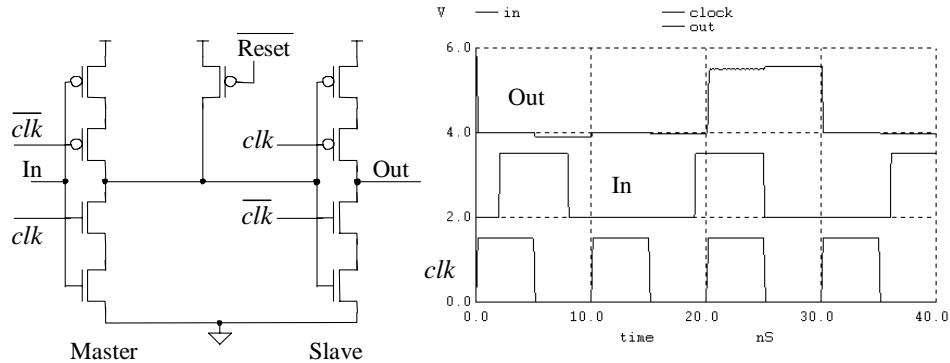


Figure 33.44 Simple delay element using clocked CMOS logic.

Both cells in Figs. 33.43 and 33.44 require the use of two clock signals. Figure 33.45 shows a delay element [7] that functions as an edge-triggered D flip-flop with a single clock signal. The circuit technique used to implement this delay is termed *true single-phase clocking* (TSPC). As seen in Fig. 33.45, both true and complement outputs are available (the true output being buffered). The current pulled by this delay cell, when clocked at 100 MHz, is below 10 μ A. The layout area is comparable to the layout area used by the clocked CMOS delay in Fig. 33.46.

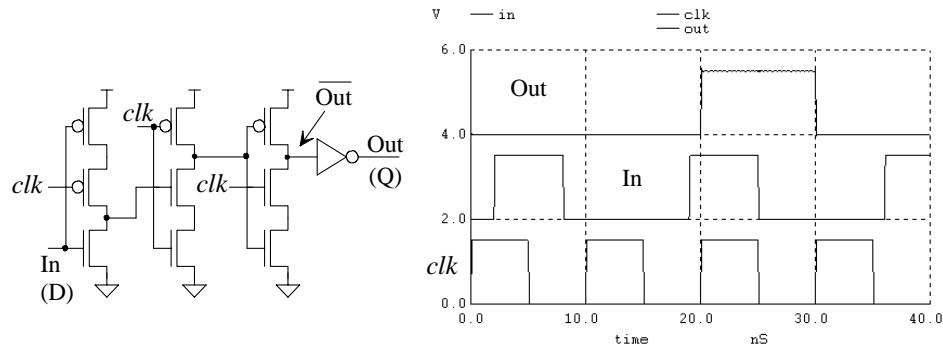


Figure 33.45 Delay element using TSPC.

By having a complementary output available, and requiring only a single-phase clock signal, a divide-by-two circuit can be implemented, Fig. 33.46, using TSPC. We used clock frequency dividers in the implementation of both decimating and interpolating filters (see Fig. 31.63, for example). A divide-by-four is implemented by cascading two divide-by-two circuits. In the general case, we can implement up- or down-ripple counters by cascading TSPC delays as seen in Fig. 33.47. Also shown in this figure is a synchronous up counter.

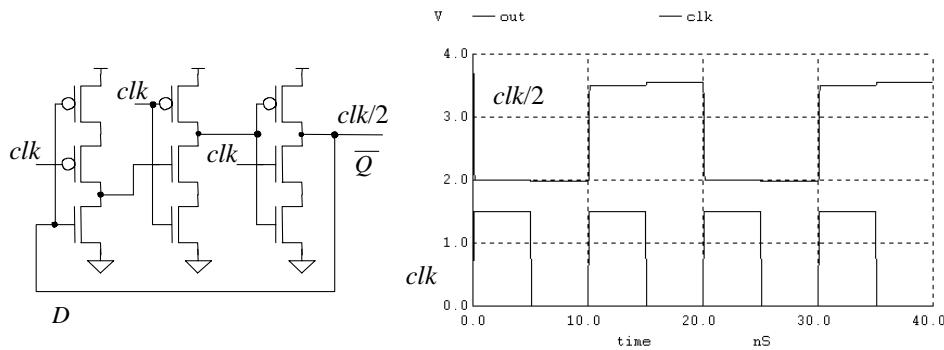


Figure 33.46 Divide-by-two circuit using TSPC.

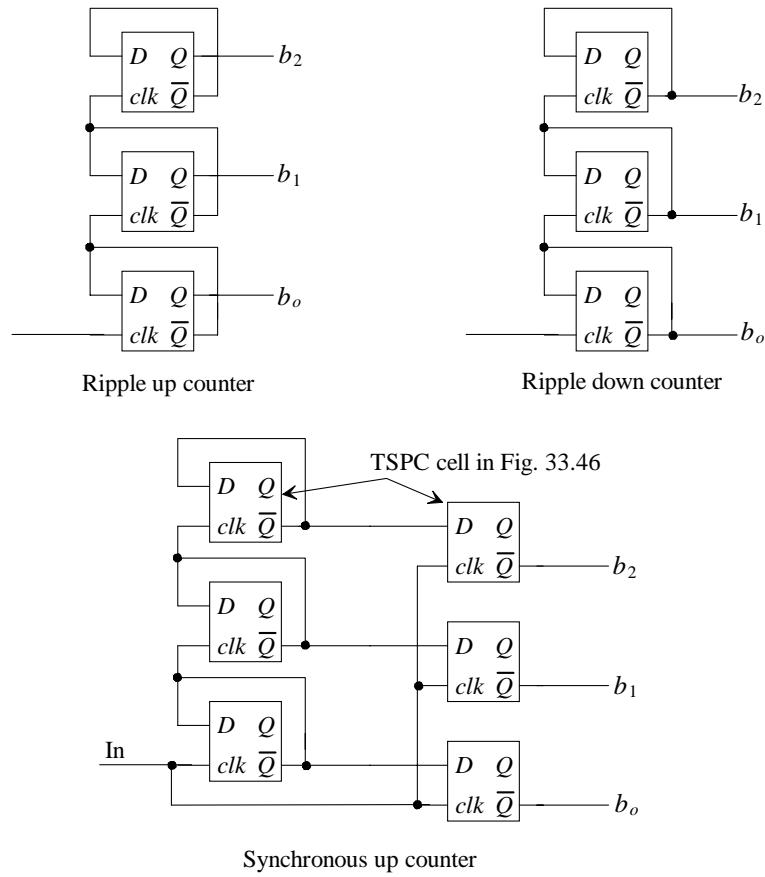


Figure 33.47 Counters using edge-triggered latches.

Example 33.5

Sketch the implementation of a synchronous up/down counter. Discuss its operation.

Figure 33.48 shows the basic block diagram of the counter. An adder is used to either add or subtract one from the contents of the register. Two's complement numbers are used in the adder to avoid overflow problems (see Ex. 31.22). The MUX selects either +1 ($= 000\dots 1$) if UP is high or -1 ($= 111\dots 1$) if UP is low to add to the contents of the register. ■

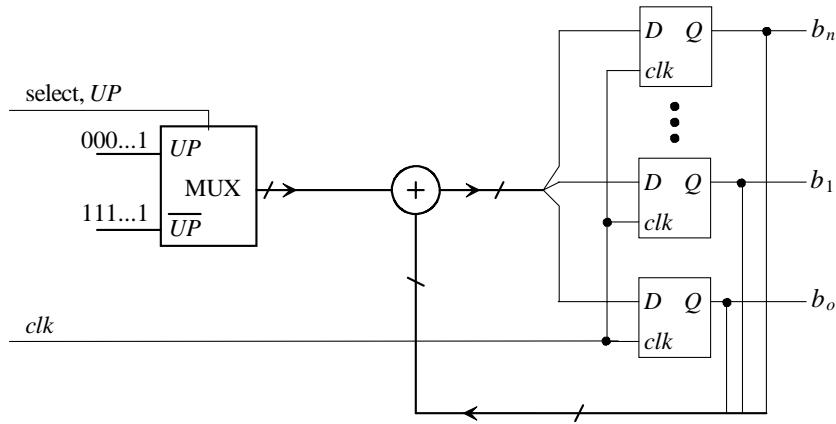


Figure 33.48 Synchronous up/down counter using an adder.

33.2.3 An Adder

The last digital building block we will look at in this chapter is the adder. An adder was used in all of our digital filters discussed in Chs. 31 and 32. While there are many ways to implement adders, here we discuss ripple adders using dynamic logic. Again these designs result in low power and a small layout area. If the delay through the adder is too long, for a specific application, we use pipelining, Fig. 33.49, to segment the adder's internal delays. Note that using pipelining results in a delay in series with the adder. The adder in Fig. 33.49 would have a z^{-3} delay in series with the output signal. In a practical circuit we would segment 4-bit, or more, adders instead of the single bit adders shown in the figure.

The output of a 1-bit adder can be written as

$$s_{out} = a_{in} \cdot b_{in} \cdot c_{in} + (a_{in} + b_{in} + c_{in}) \cdot \overline{c_{out}} \quad (33.16)$$

where a_{in} and b_{in} are the adder's inputs, while c_{in} is the carry input. The carry output can be written as

$$c_{out} = a_{in} \cdot b_{in} + c_{in} \cdot (a_{in} + b_{in}) \quad (33.17)$$

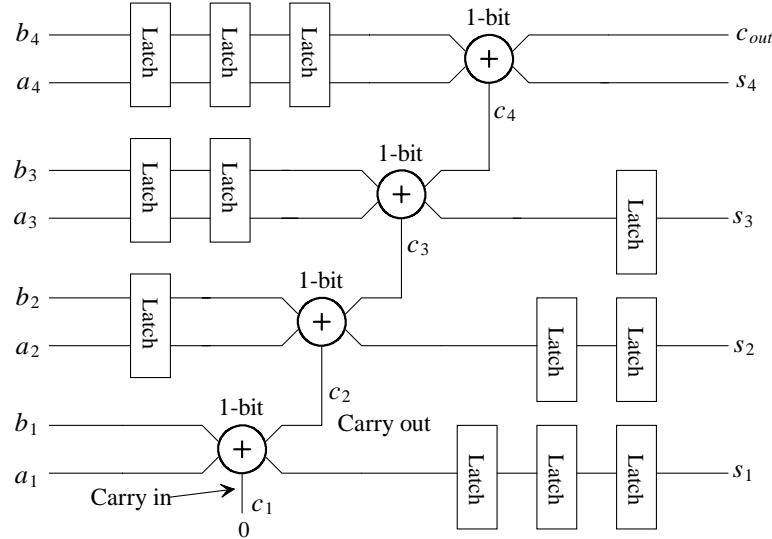


Figure 33.49 A 4-bit pipelined adder. The latches (clocked) behave as delay elements.

Figure 33.50 shows the implementation of a dynamic adder. The first stage generates the carry out and is implemented in NMOS precharge-evaluate (PE) logic. The second stage is implemented using PMOS PE logic. The overall gate can be thought of as a domino gate. The output of the adder is valid when clk is high.

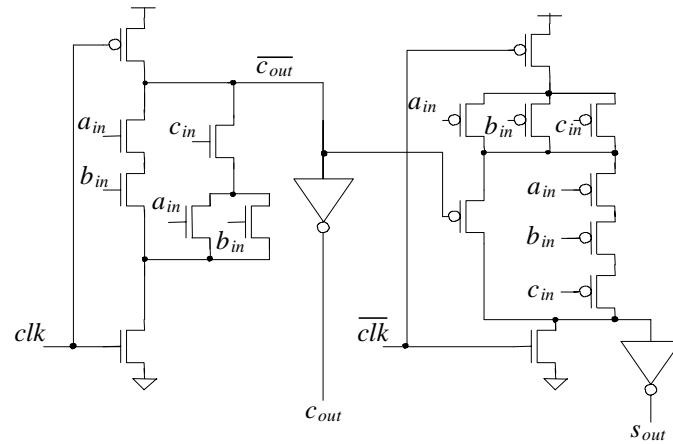


Figure 33.50 Full-adder bit implemented using dynamic logic.

33.3 Analog Circuit Design

In this section we discuss analog circuit design using a submicron CMOS process. We assume the reader is familiar with the fundamentals of analog design presented in Chs. 20-25 of [1]. The first subsection presents a discussion of biasing concerns, the second subsection discusses op-amp design, and the third subsection covers circuit noise.

33.3.1 Biasing

When starting an analog design we begin by selecting the excess gate voltage, ΔV , also known as the overdrive voltage, and the biasing current level. Selecting these two parameters sets the width and length of the MOSFETs. While using large area devices ($L \cdot W$) that are placed close together results in the best device matching [8], our focus here will be on speed and power.

Selecting the Excess Gate Voltage

Figure 33.51 shows the basic source cross-coupled diff-amp first discussed in Ch. 24. For the moment we will assume the minimum voltage across the drain and source of a MOSFET, $V_{DS,sat}$, is ΔV . Further, we know that the gate-to-source voltage of a MOSFET, V_{GS} , can be written as $\Delta V + V_{THN}$. In our $0.15\text{ }\mu\text{m}$ process the NMOS and PMOS devices

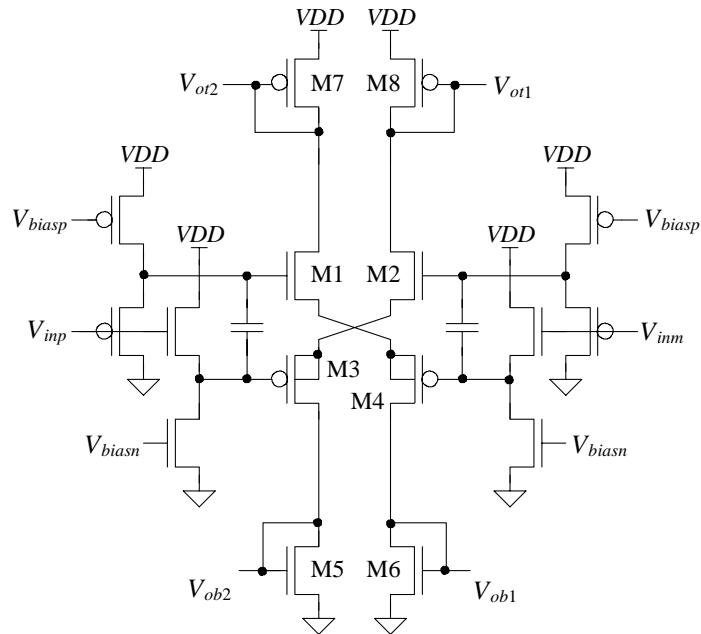


Figure 33.51 Op-amp input diff-amp without slew-rate limitations.

have the same threshold; 0.4 V at room temperature in a typical process run. We will design our circuits so the PMOS and NMOS devices have the same gate-to-source voltage. Writing KVL from VDD to ground through M8, M2, M3, and M5 results in

$$VDD = V_{SG8} + V_{DS2} + V_{SD3} + V_{GS5} \quad (33.18)$$

or

$$VDD = \overbrace{\Delta V + V_{THP}}^{V_{SG8}} + \overbrace{\Delta V + \Delta V}^{V_{DS2}} + \overbrace{\Delta V}^{V_{SD3}} + \overbrace{\Delta V + V_{THN}}^{V_{GS5}} \quad (33.19)$$

noting, in Fig. 33.51, we've eliminated the body effect in M3 and M4 by placing them in their own well. Because $VDD = 1.5$ and $V_{THP} = V_{THN} = 0.4$, we can set our ΔV to at most 175 mV. We know that the threshold voltage will change with process runs and with temperature so we will select our ΔV as 100 mV to provide some margin for these shifts. This means our nominal V_{GS} (or V_{SG} for the PMOS) in the designs presented here is 0.5 V and the drain-to-source voltages are 250 mV.

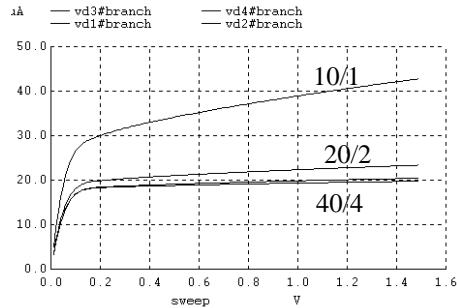
Selecting the Channel Length

The small-signal output resistance decreases with decreasing channel length. Because of this we might just select a channel length, for analog applications, ten times the minimum length and be finished. However, as we'll show in a moment, the speed of the MOSFET is inversely proportional to the channel length and, so, long L equates to slow speed. Also, layout area is always premium real-estate; therefore, using the smallest possible devices is usually desirable.

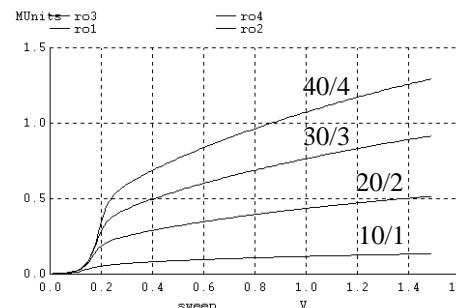
Consider the test circuit shown in Fig. 33.52. We've applied 0.5 V DC to the gate of the NMOS device and swept the drain-to-source voltage of the MOSFET. This circuit was simulated for four different device sizes: 40/4, 30/3, 20/2, and 10/1. Figure 33.52a shows how the drain current changes with changing V_{DS} . More interesting is the plot in 33.52b showing the MOSFET's output resistance, r_o . Notice, as we would expect, the output resistance increases with increasing L . The key point here is that we can increase the output resistance and thus gain by increasing L . Unfortunately, this results in a decrease in speed. For the designs presented here we will use an L of 2 (a 20/2 NMOS device) as a reasonable trade-off between gain and speed. The nominal drain current that flows in a 20/2 NMOS device with a V_{GS} of 0.5 V is 20 μ A. Also note from Fig. 33.52b that the MOSFET appears to enter the saturation region at approximately 250 mV (not the 100 mV we assumed earlier for $V_{DS,sat}$).

Figure 33.53 shows the output resistance plots for the PMOS device. Because of the weaker drive of the PMOS, we used 20/1, 40/2, 60/3, and 80/4 device sizes in an attempt to match the PMOS and NMOS drive strengths. Note that the gate of the PMOS device is connected to a 1 V supply to set the V_{SG} of the PMOS device to 0.5 V. We use a 40/2 PMOS device for the designs presented here.

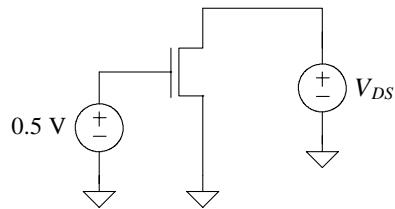
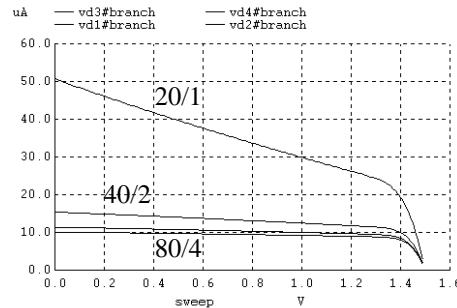
It should be clear, after looking at Figs. 33.52 and 33.53, that we will often not be operating our devices deep in saturation (and so our gain will suffer). If our device's drain-to-source voltage is 100 mV, we will be operating our amplifiers in the triode



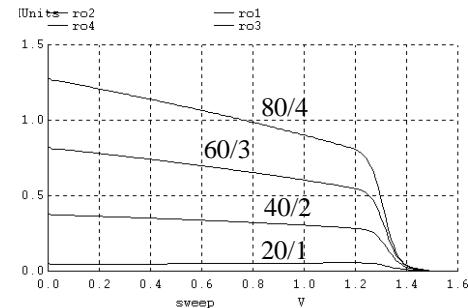
(a) Drain current vs. drain-source voltage



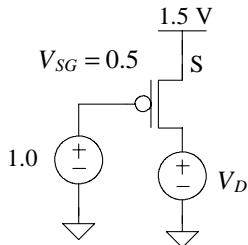
(b) Output resistance vs. drain-source voltage

**Figure 33.52** NMOS curves for 40/4, 30/3, 20/2, and 10/1 devices.

(a) Drain current vs. drain voltage



(b) Output resistance vs. drain voltage

**Figure 33.53** PMOS curves for 80/4, 60/3, 40/2, and 20/1 devices.

region. The circuit-design techniques that we use for submicron circuit design should allow reasonable gains even if our transistors move into the triode region with temperature or process variations.

Small-Signal Transconductance, g_m

Figures 33.54 and 33.55 show the test circuits and simulation results used to determine each device's small-signal transconductance, g_m . Note how we adjusted the DC drain-to-source voltage so that the devices are operating in the triode region (see Figs. 33.52 and 33.53). Note also how we've tried to size the PMOS device to have similar characteristics as the NMOS but the g_m is still half that of the NMOS device.

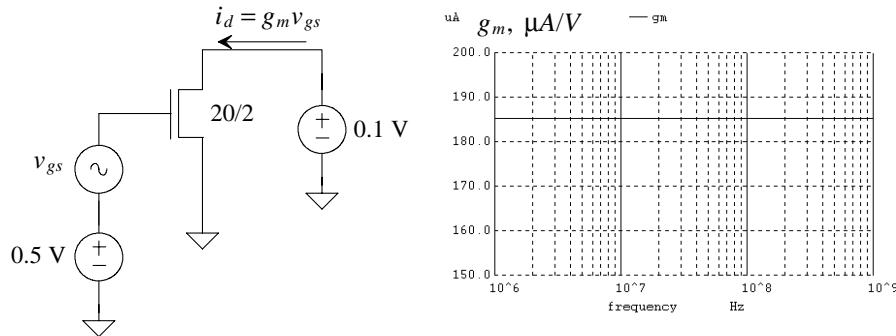


Figure 33.54 NMOS small-signal transconductance.

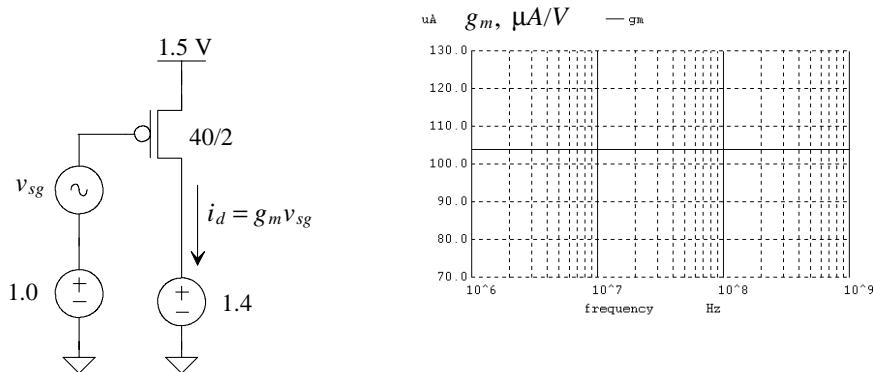


Figure 33.55 PMOS small-signal transconductance.

MOSFET Transition Frequency, f_T

The MOSFET's transition frequency, f_T , is defined as the frequency where the AC gate current is equal to the AC drain current, or, $|i_d/i_g| = 1 = 0 \text{ dB}$. This figure-of-merit was

discussed back in Chs. 9 and 25. Figures 33.56 and 33.57 show how f_T is determined from simulation results. Large f_T indicates high speed. From a circuit point of view

$$f_T \propto \frac{V_{GS}}{L} \quad (33.20)$$

Using minimum-length devices with large gate-to-source voltages results in high-speed operation with low gain (because of the small output resistance, as seen in Figs. 33.52 and 33.53) and reduced output swing (using a large V_{GS} results in devices that enter the triode region with relatively large drain-to-source voltages).

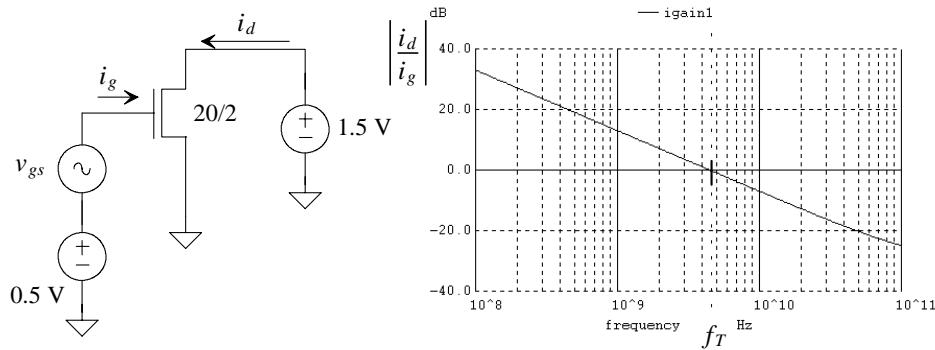


Figure 33.56 NMOS transition frequency.

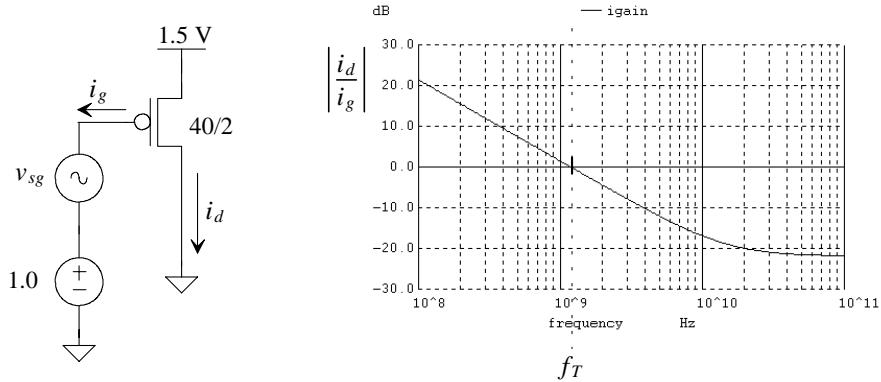


Figure 33.57 PMOS transition frequency.

The Beta-Multiplier Self-Biased Reference

Figure 33.58 shows the schematic diagram of a beta-multiplier self-biased reference designed to provide nominally 10 μ A of current through a 20/2 NMOS or 40/2 PMOS. We select the beta-multiplier because of its robust temperature characteristics. The increase in the resistor with temperature is compensated for by the decrease in the

MOSFET's threshold voltage. The 80/1 device, M2, is large relative to M1 so that its V_{GS} is approximately 0.4 V (the threshold voltage). Because M2's V_{GS} is 0.5 V, there is 100 mV dropped across the 10k resistor (and this sets the current). In a practical circuit, subject to process variations, we can adjust the resistor value by adding series-shorted and parallel resistors to the source of M2. This makes adjusting the current to a specific value possible. Note, as discussed in the homework problems of Ch. 21, adding a capacitance to ground at the source of M2 can result in an unstable circuit. The reference can actually oscillate. This may be a problem if the resistor is bonded out to set the current value. Adding capacitors to ground or VDD at V_{biasn} and V_{biasp} , however, can often be useful to reduce coupled noise to the bias voltages.

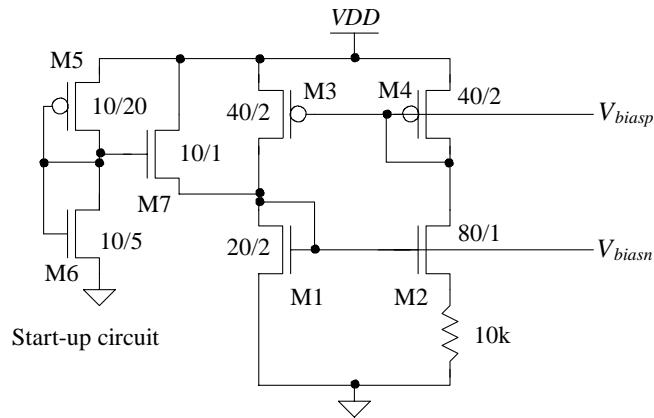


Figure 33.58 Beta-multiplier self-biased reference circuit.

The temperature behavior of the beta-multiplier reference self-biased circuit of Fig. 33.58 is shown in Fig. 33.59. The temp-co of the resistor is set to 2,400 ppm/C. VDD is swept on the x-axis while the current flowing in M2/M4 is shown on the y-axis. While we might think that we can get better power-supply rejection by cascading the MOSFETs, we would find that many of the MOSFETs would operate in the triode region.

Figure 33.60 shows a general biasing circuit for analog design. The voltages V_{biasn} and V_{biasp} are the inputs to the bias circuit and are supplied by the beta-multiplier of Fig. 33.58. The reader who doesn't understand the origin of this circuit is referred back to Ch. 20. Note, in this biasing circuit, how we have made an effort to reduce the number of MOSFET gate-source voltages in between VDD and ground. This is necessary for low-voltage operation. In the general design, as discussed back in Ch. 20, the size of the $\frac{1}{4} \cdot \frac{W}{L}$ device can be reduced (to, say, $\frac{1}{5} \cdot \frac{W}{L}$) to bias the device closest to the power supply rails further into the saturation region. This will generally provide higher gains at the cost of slightly reduced output swing. Figure 33.61 shows the output current through both NMOS and PMOS cascode current mirrors biased with V_{bias1} through V_{bias4} .

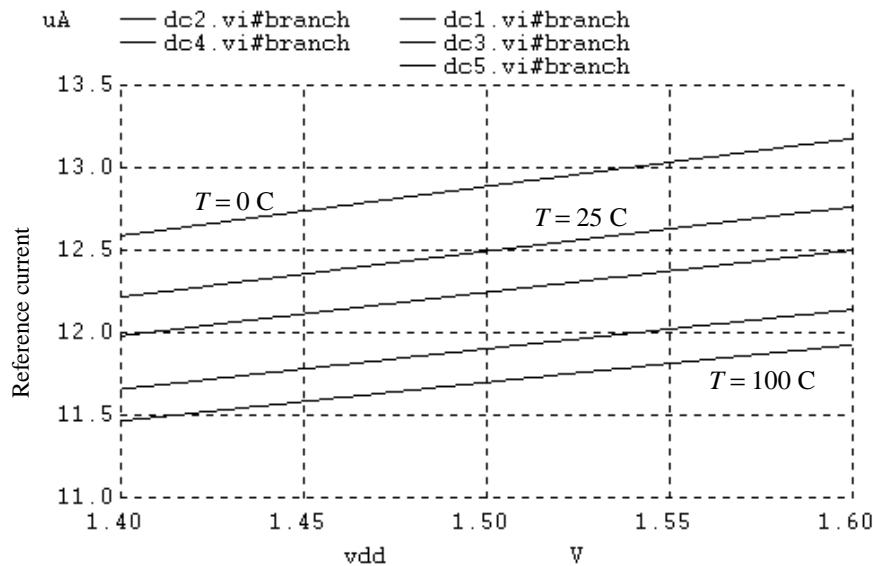
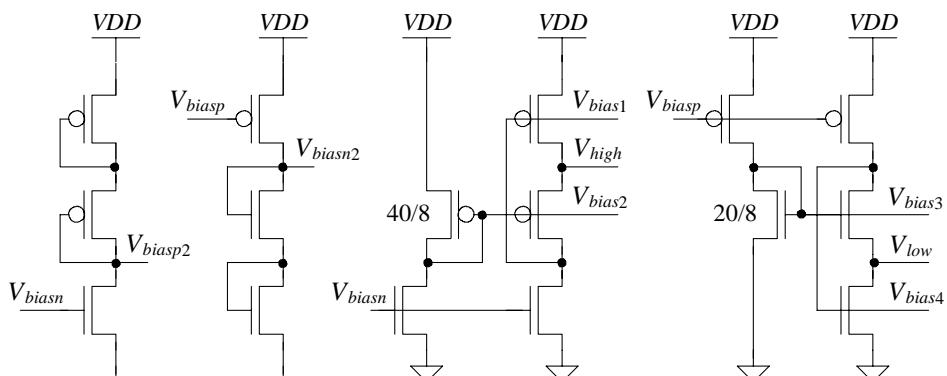


Figure 33.59 Temperature behavior of the reference of Fig. 33.58.



All unlabeled PMOS are 40/2
All unlabeled NMOS are 20/2

Figure 33.60 General biasing circuits for analog design.

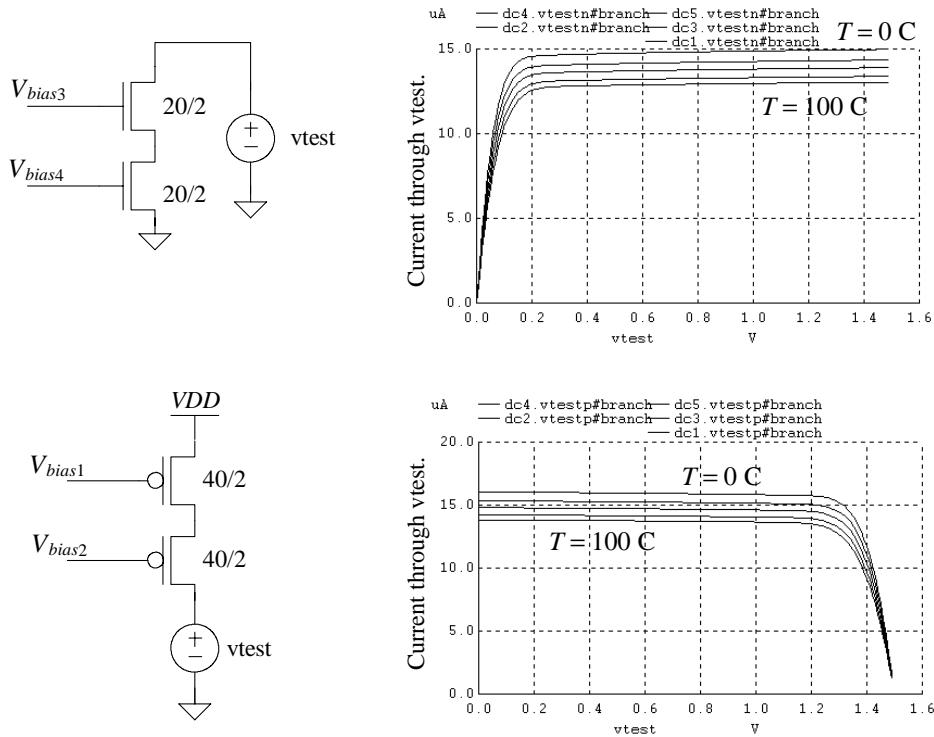


Figure 33.61 Using the general purpose biasing circuit of Fig. 33.60.

Figure 33.62 shows how the currents in each leg of the diff-amp of Fig. 33.51 change as we sweep V_{inp} with V_{mm} held at 0.75 V. Note how, with $V_{inp} = V_{mm} = 0.75\text{ V}$, the current in each leg of the diff-amp is 6 μA or less than the current in the biasing circuits. This is the result of the body effect experienced by the MOSFETs in the diff-amp.

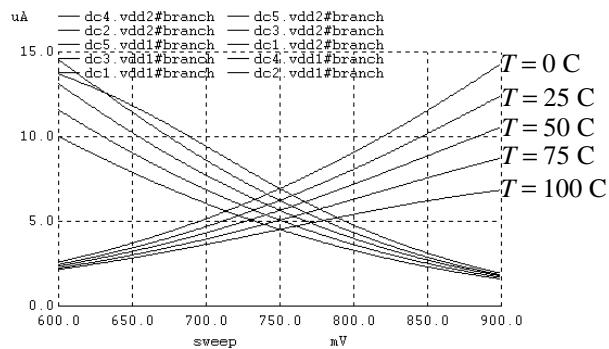


Figure 33.62 DC sweeps showing the currents in each leg of the diff-amp of Fig. 33.51.

33.3.2 Op-Amp Design

In this section we present the design of general-purpose, mixed-signal op-amps. Once again we assume the reader is familiar with the material in Ch. 25. In particular, compensating two-stage op-amps, floating current sources, output buffers, and fully-differential op-amps. We use the biasing circuits developed in the last section in the designs presented in this section. The goals of the designs presented here are robust operation over wide process variations and temperatures.

Output Swing

Let's begin our op-amp development by considering the inverter shown in Fig. 33.63. In order for our op-amp to have the widest possible output swing *we must* use this structure on the output of the op-amp. (Sometimes this output structure is called a push-pull amplifier because it can both source [push] and sink [pull] a current from the load.) Notice how we've increased the size of the MOSFETs used in this structure from our standard 40/2 and 20/2 devices. This was to both increase the output drive of the op-amp and to increase the output stage's input capacitance. This capacitance will be used to compensate the op-amp. Also, because op-amp open-loop gain, A_{OL} , is always important, using an output stage with gain increases the op-amp's A_{OL} .

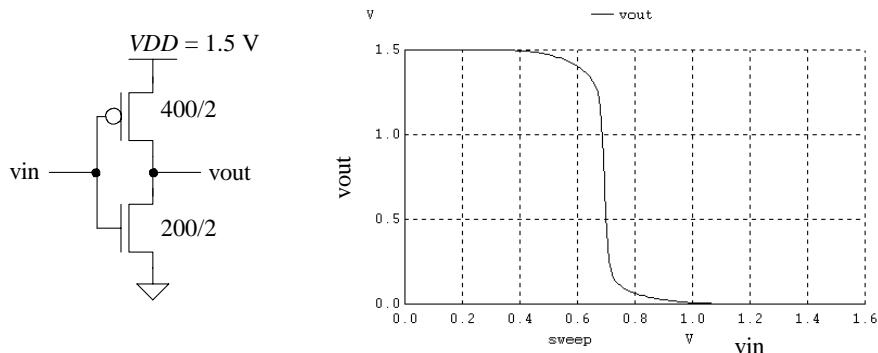


Figure 33.63 Using an inverter on the output of an op-amp.

Looking at the transfer curve shown in Fig. 33.63, we see that the gain of the inverter (the slope of the curve) is largest when the output falls between 0.25 and 1.25 V (an output swing of 1 V). Throwing away 0.5 V, or 33%, of the power-supply voltage on a MOSFET operating in the triode region is, of course, highly undesirable. Thinking about this for a moment we may realize that if this (second) stage is preceded by a high-gain (first) stage the op-amp may still function within specifications when the second stage gain is dropping. (Though the distortion introduced by the output stage may be too high because of this nonlinearity.) A more important concern then is the quiescent current pulled through this stage. For the inverter in Fig. 33.63 this current is > 1 mA. For both gain and power reasons we need to modify this basic output circuit.

To lower the quiescent current pulled from VDD and to increase the linear output swing of the output stage consider adding batteries to the circuit as seen in Fig. 33.64. We've selected the batteries so that when v_{in} is 0.75 V the gate-source voltages of the MOSFETs are 0.5 V. From Figs. 33.52 and 33.53 we can estimate the quiescent current in the output stage as 150 μ A (simulation results verify this estimate). Clearly, increasing the battery voltages will make the output swing approach the power supply rails before the MOSFETs enter the triode region. This increase in linear output swing comes at the cost of speed, as indicated by Eq. (33.20).

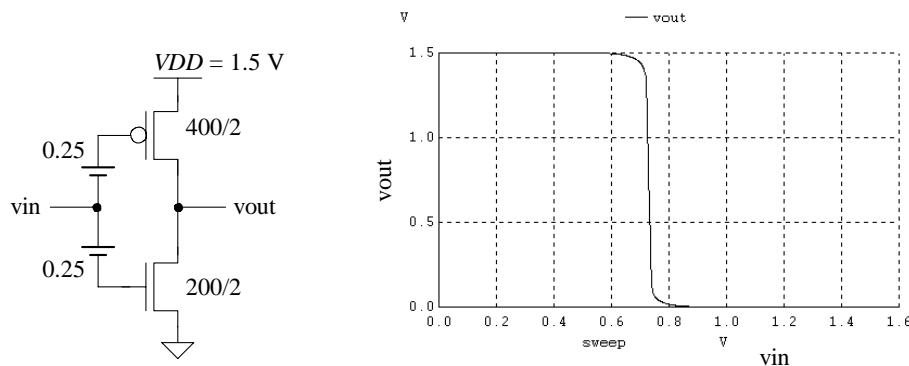


Figure 33.64 Biasing a push-pull amplifier.

The next question becomes, "How do we implement the batteries in Fig. 33.64?" The batteries must track both process and temperature changes. What we want is something like what is seen in Fig. 33.65. Using V_{biasn2} from the circuit of Fig. 33.60, we can precisely set the current in the output stage. Looking at this figure for a moment, we

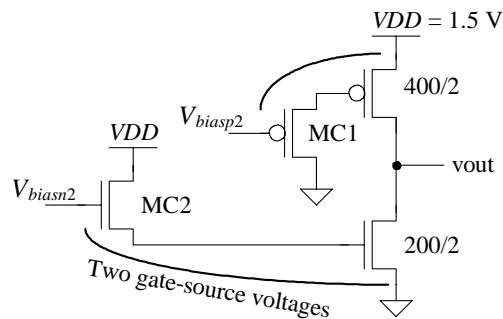


Figure 33.65 Conceptual biasing of a push-pull amplifier with current mirrors.

see that we need to somehow couple our input signal to the gates of the output MOSFETs and we need to provide a path for current flow in the added transistors. As drawn MC1 and MC2 are off.

Figure 33.66 shows the use of a floating current source (discussed in Ch. 25) to set the bias current in the output stage. Note that since the current in the cascoded transistors splits between MC1 and MC2, we have reduced their size by one-half in order to set the current in the output MOSFETs to precisely ten times the current in the remaining MOSFETs. By further reducing the size of MC1 and MC2 (reducing W or increasing L), we can choke off the current flowing in the output transistors. While this will push the frequency of the pole located on the output of the op-amp downwards, ultimately making stability a concern, it may be used to reduce the quiescent power of the op-amp. The input signals, V_{ot1} and V_{ob1} , come from the diff-amp shown in Fig. 33.51. The cascode transistors provide the "first stage gain" so that the circuit shown in Fig. 33.66 is an op-amp minus the diff-amp. The structure is biased so that it can function with very low power supply voltages. The limitation on how low the power-supply voltages can go is set by the diff-amp of Fig. 33.51.

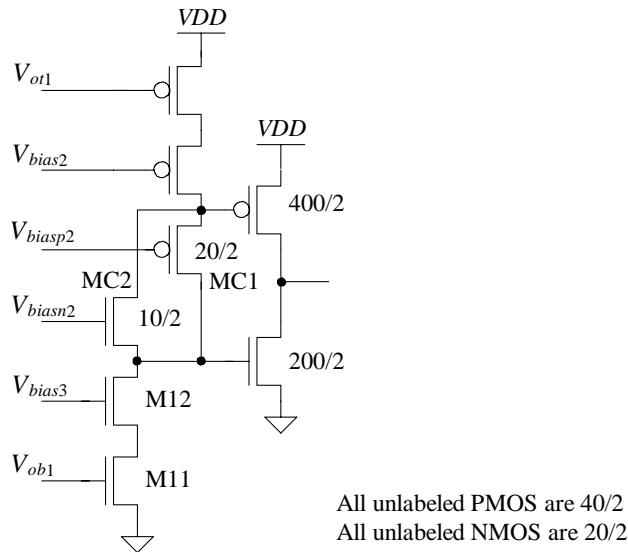


Figure 33.66 Biasing a push-pull output stage with floating current source.

Example 33.6

Consider the AC small-signal simplification of the floating current source shown in Fig. 33.67. Assuming the NMOS cascode output resistance is labeled R_{ncas} , what is the small-signal resistance seen by the test voltage, v_{test} ?

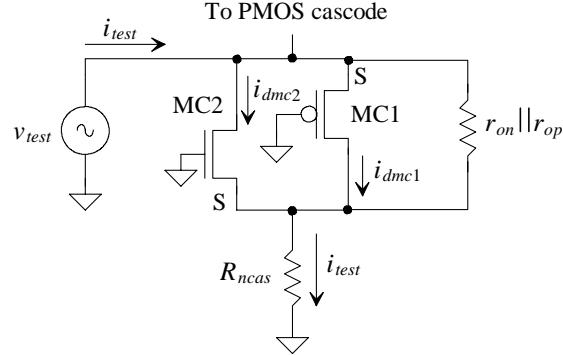


Figure 33.67 Small-signal AC circuit for Ex. 33.6.

What we are going to show is that the floating current source will not load, or decrease, the resistance seen by the cascode structures. Writing a KVL from v_{test} to ground results in

$$v_{test} = (i_{test} - i_{dmc1} - i_{dmc2}) \cdot r_{on} || r_{op} + i_{test} \cdot R_{ncas}$$

where the drain currents of MC1 and MC2 are i_{dmc1} and i_{dmc2} , respectively, and their output resistances are r_{on} and r_{op} . The drain currents can be written as

$$i_{dmc1} = g_{mn} \cdot v_{gs} = g_{mn} \cdot (-i_{test} \cdot R_{ncas})$$

and

$$i_{dmc2} = g_{mp} \cdot v_{sg} = g_{mp} \cdot v_{test}$$

Combining these equations

$$\underbrace{v_{test} \cdot (1 + g_{mp} \cdot r_{on} || r_{op})}_{\approx g_{mp} \cdot r_{on} || r_{op}} = i_{test} \underbrace{(1 + g_{mn} \cdot R_{ncas} \cdot r_{on} || r_{op} + R_{ncas})}_{\approx g_{mn} \cdot r_{on} || r_{op} \cdot R_{ncas}}$$

or

$$\frac{v_{test}}{i_{test}} \approx R_{ncas}$$

This shows that adding the floating current source to our cascode stack in Fig. 33.66 will not affect the gain of the circuit. ■

Slew-Rate Concerns

We know from our discussion in the last chapter that slew rate can be a big concern when designing a mixed-signal system. For example, the drain of the NMOS device, M11, at the bottom of Fig. 33.66 will be at a voltage of approximately 150 mV. If the voltage V_{ob1} increases, the maximum amount this device (M11) can turn on is very limited. If we were able to hold its drain voltage constant, then an increase V_{ob1} would result in an increase in

drain current. Now, however, the current through M12 is constant (its gate is held at V_{bias3} , while its source is held at a fixed potential). What we need to add to this basic circuit is a circuit that will hold the drain of M11 at a fixed potential while at the same time adjust the gate voltage of M12 so that it can turn on.

Figure 33.68 shows adding N and P diff-amps to help with adjusting the biasing so that slewing isn't a concern. Figure 33.69 shows the circuit implementation of the amplifiers. The source followers were added in Fig. 33.69 so the inputs can go to the power supply rails and to reduce the input capacitance. Adding the N and P amplifiers to the amplifier of Fig. 33.66 (called gain-enhancement back in Ch. 25) also increases the output resistance of the cascode stack. This is important because we want the pole associated with this node (the output of the first stage or the input to the second stage) to be dominant so that it compensates the op-amp. The voltages V_{high} and V_{low} are generated with the bias circuit of Fig. 33.60 for, once again, the widest possible operating range. As discussed in Ch. 25, we can increase the gain of the op-amp by increasing the gains of these added amplifiers.

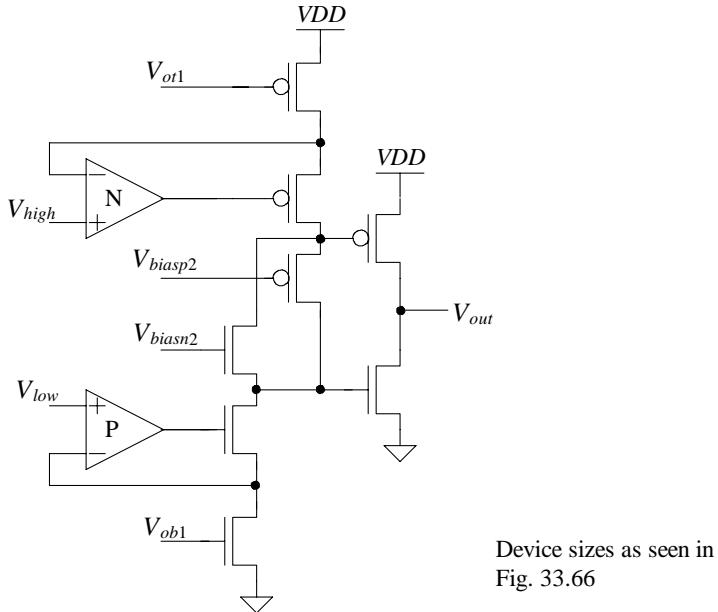


Figure 33.68 Adding amplifiers to our op-amp to boost gain and help slew-rate.

Reviewing Fig. 33.62 we see that the current sourced by our diff-amp is very limited. Let's say that there is $5 \mu\text{A}$ of current available to charge the output transistors in Fig. 33.68. We can estimate the input capacitance of these two transistors using

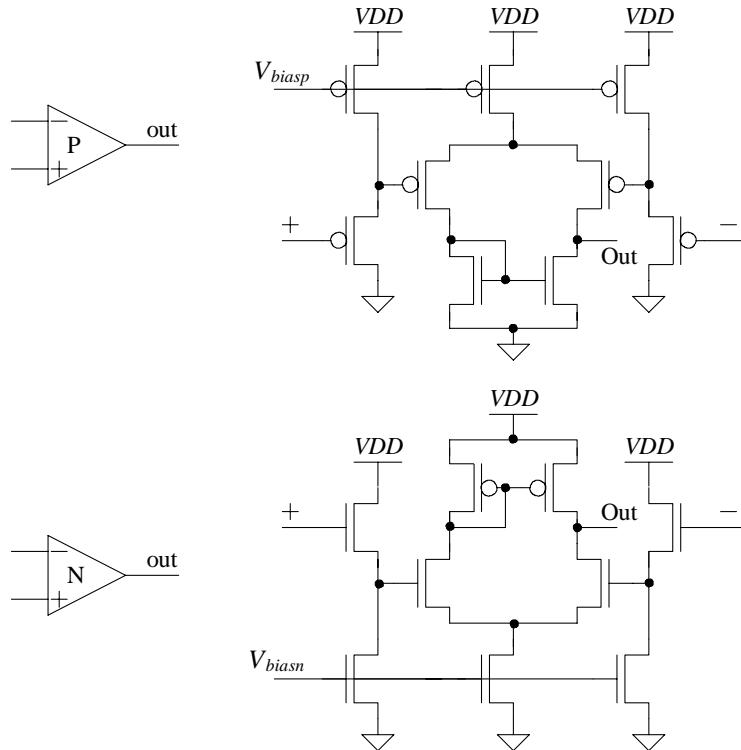


Figure 33.69 Diff-amps with source-follower level shifters for use in Fig. 33.68.

$$C_1 = \left(\overbrace{400}^{W_p} + \overbrace{200}^{W_n} \right) \cdot \overbrace{2}^L \cdot \overbrace{\frac{C'_{ox}}{t_{ox}}}^{\text{(scale)}^2} = 1,200 \cdot (0.15 \mu)^2 \cdot \frac{35.13 \text{ aF}/\mu\text{m}}{0.004 \mu\text{m}} = 237 \text{ fF}$$

(33.21)

The rate we can charge this input capacitance is

$$\frac{5 \mu\text{A}}{237 \text{ fF}} = \frac{dV}{dt} = 21 \text{ mV/ns}$$

(33.22)

At first glance this may appear to be a significant limitation if the output of our op-amp has to change by 1 V or more during a 10 ns clock cycle. However, after reviewing Fig. 33.64 we see that well under 50 mV change on the input of these output transistors is needed to cause the output to change from rail to rail. One final comment: To balance the drive to the output transistors, a capacitor can be added in between each of the gates of

the two transistors. This capacitor doesn't affect the compensation or the speed, it simply makes the biasing appear more battery-like, as seen in Fig. 33.64. We also added capacitors in the diff-amp of Fig. 33.51 so that the source-followers used for biasing appear more battery-like (as discussed in Ch. 24).

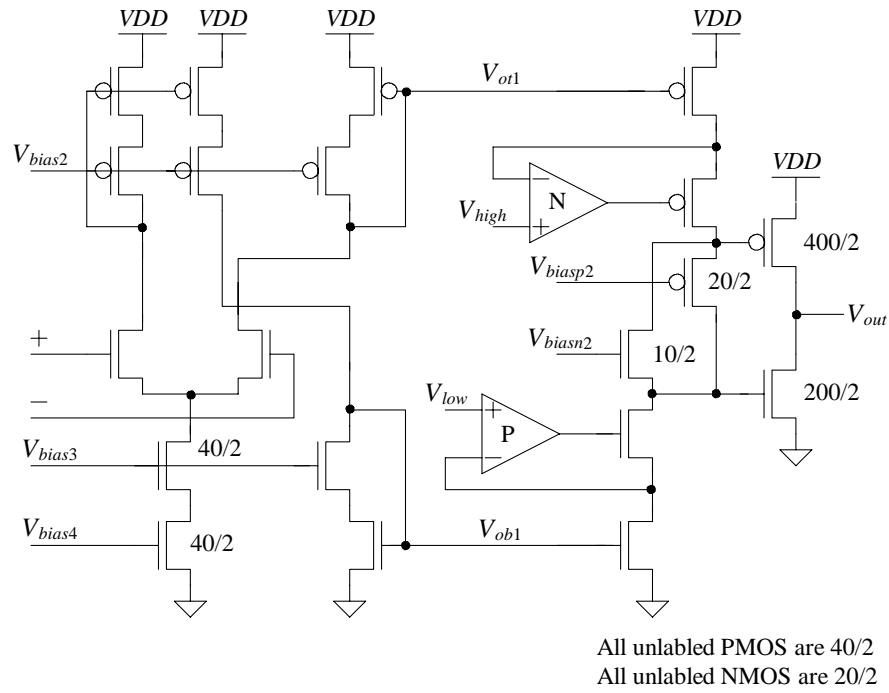
Now that we've calculated the capacitance on the output of the first stage, C_1 , we can estimate the location of the dominant pole. The voltage gain of the cascode section is approximately $(g_m r_o)^2$. When the gain enhancement amplifiers (N and P in Fig. 33.69) are added the voltage gain increases to $(g_m r_o)^3$ noting this is the gain to the output of the first stage and not the final gain of the op-amp. If we pull the transconductance of the diff-amp out of this equation, we estimate the cascode output resistance as $g_m^2 r_o^3 (= R_1)$. Using Figs. 33.52 - 33.55 we can estimate R_1 as 150 MΩ. The location of the dominant pole is then estimated as $f_1 = 1/(2\pi \cdot 150 \text{ M}\Omega \cdot 237 \text{ fF}) = 4.4 \text{ kHz}$. This pole can be pushed lower in frequency, further compensating the op-amp, by adding capacitance from the gates of the output MOSFETs to AC ground (ground or VDD).

The observant reader may ask, "If we can get away with only 5 μA of bias current in our amplifier and avoid slew-rate limitations, why use the diff-amp of Fig. 33.51?" As we discussed earlier, this diff-amp represents the weak link in the minimum power supply voltage we can use with our op-amp and it dissipates more power than an equivalently biased regular diff-amp. Also, the input common-mode range of this diff-amp is very limited. We won't be able to use the op-amp as a simple voltage follower. Because of these concerns/reasons we won't use this topology for our basic op-amp design. (The noise performance, discussed in the next section, is also poorer for this diff-amp mainly because the source followers used for biasing are in series with the input signal.)

Figure 33.70 shows one possible mixed-signal op-amp topology. We used diode connected, cascaded MOSFETs to generate V_{ot1} and V_{ob1} in the diff-amp in an effort to equalize all drain-source voltages. The minimum supply voltage, because of the diff-amp used, can be significantly less than 1.5 V (approaching 1 V for a typical process run). One potentially important concern is the negative common-mode range of the diff-amp. If the inputs are held at 0.75 V, with the gate-source voltage of the diff-amp pair at 0.5 V, there will be only 0.25 V left to drop across the diff-amp's current sink. The result is the diff-amp's biasing current may decrease (and so will the common-mode rejection ratio). Using a single MOSFET to bias the diff-amp or increasing the widths of the diff-pair can improve this situation.

The gain of this topology is very high. This can lead to simulation problems (which has led us to use HSPICE in the following simulation results). Figure 33.71 shows the output of the op-amp as a function of the noninverting input voltage of the op-amp with the inverting input held at 0.75 V. The systematic offset voltage is approximately 3 mV. Not cascading the diode-connected loads of the diff-amp can result in a significantly larger offset voltage. Increasing the widths of the diff-amp pair reduces the systematic offset, and, if laid out properly, reduces the random offsets.

Figure 33.72 shows the open-loop gain of the op-amp (approximately 110 dB at DC). As is, the op-amp will be unstable if used in a unity gain configuration. The dominant

**Figure 33.70** Mixed-signal op-amp.

pole, as discussed above, is at the output of the first stage (input to the second stage). We might think that using different N and P diff-amps (or some other operational transconductance amplifier) with a higher gain will help improve the stability. While increasing the gain of these amplifiers will push the dominant pole to a lower frequency, it will also increase the low-frequency gain having little effect on the stability.

Figure 33.73 shows how adding two 250 fF capacitors to the output stage pushes the dominant pole downwards and makes the op-amp stable. The unity gain frequency of the op-amp is approximately 70 MHz. Unfortunately, the settling time of the op-amp increases. Figure 33.74 shows a test configuration where the op-amp, driving a relatively large 5 pF capacitor, has a settling time of approximately 50 ns. (Figures 33.72 and 33.73 were generated without a load; so the unity gain frequency with a load will be less than what is shown in these figures.)

While reducing the capacitive load decreases the settling time, we may not have this option available. Let's discuss the design of an op-amp and the trade-offs if we use the basic topology of Fig. 33.70 to implement a mixed-signal op-amp.

1. The load is purely capacitive. We may consider eliminating the floating current source and the push-pull output stage (and note that the inverting and noninverting input

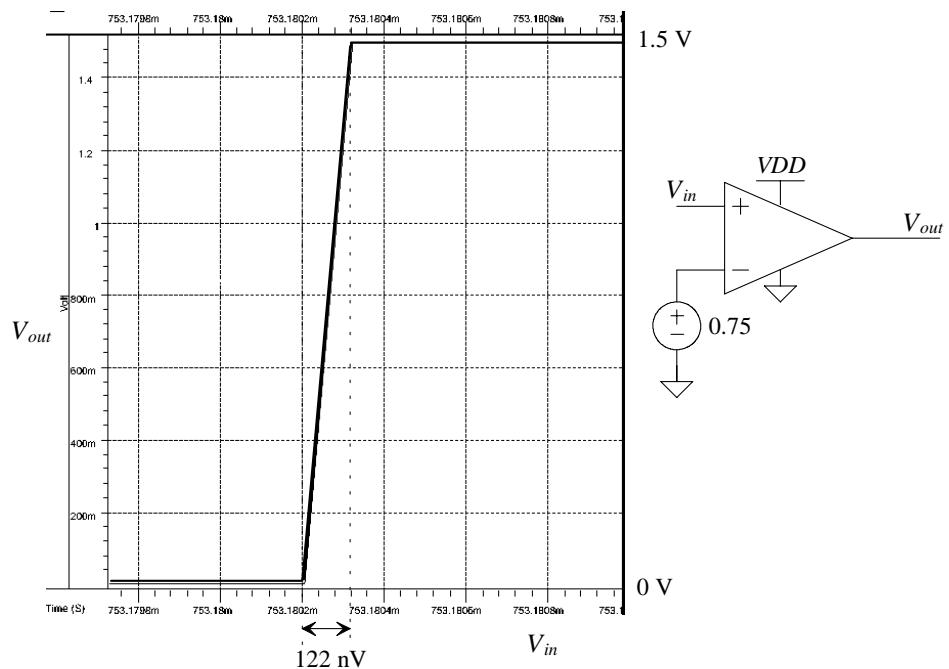


Figure 33.71 DC behavior of the op-amp of Fig. 33.70.

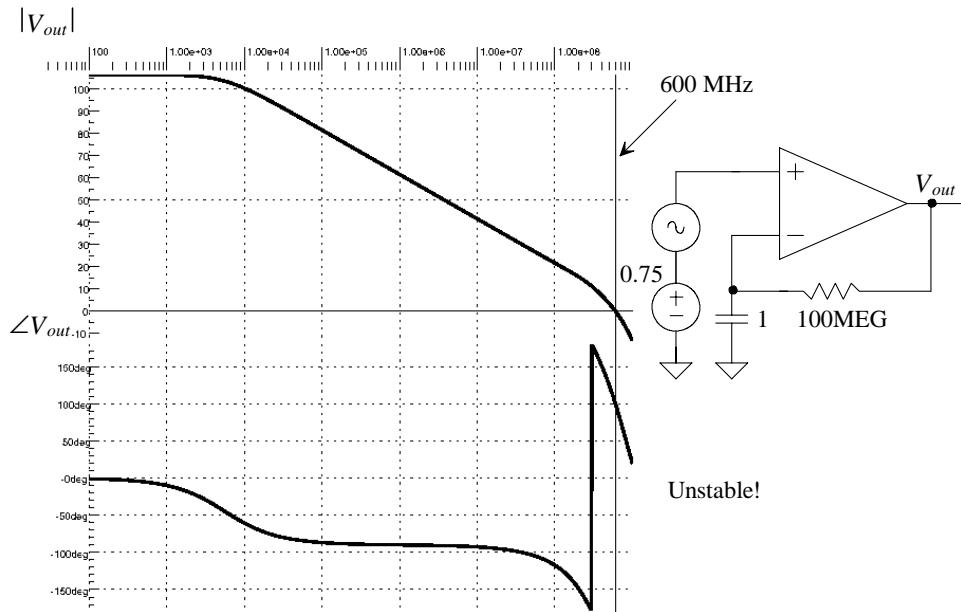


Figure 33.72 Showing open-loop response of the op-amp of Fig. 33.70.

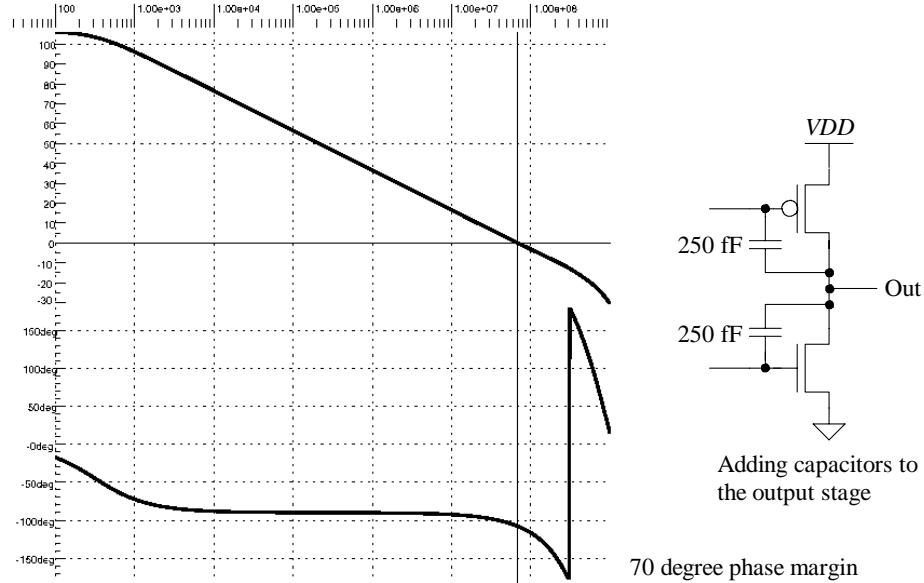


Figure 33.73 Compensating the op-amp of Fig. 33.70.

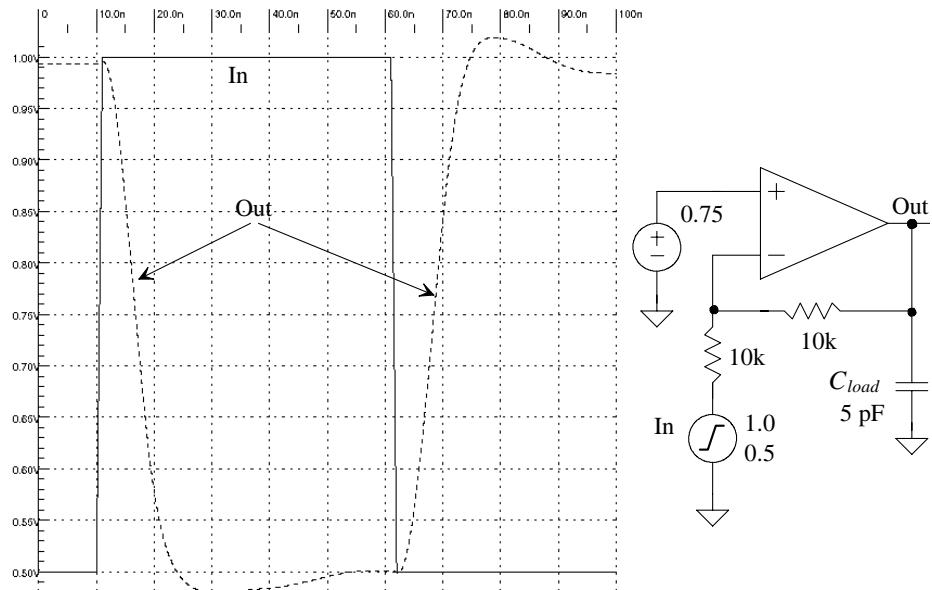


Figure 33.74 Showing settling time of the op-amp in Fig. 33.70 with the compensation capacitors shown in Fig. 33.73.

terminals switch places). While the output range is reduced when not using the push-pull output buffer, the stability is almost guaranteed with a reasonable-sized load capacitance. The biasing current is increased so that it can drive the capacitive load in the time required. A scaling increase in the biasing currents is followed by an increase in the size of the devices. For example, if we increase our biasing current from nominally 15 μA (see Figs. 33.52 and 33.53) to 150 μA , then our NMOS size is increased to 200/2 and the PMOS size is increased to 400/2. Not scaling devices, as discussed earlier, can result in too small of an f_T or too large of a ΔV .

2. We use the circuit as seen in Fig. 33.70 but the settling time is too long. By reducing the lengths of the push-pull output stage, the frequency of the second pole gets pushed out (increases) and the gain of the output stage decreases (both helping with stability). However, the current in the output stage increases, resulting in higher power dissipation. This, in most situations, isn't enough alone to guarantee a stable op-amp.

3. Increasing the biasing current lowers the gain and improves the speed (and thus decreases the settling time). The linear output range decreases (perhaps by too modest an amount to be a concern because of the large first-stage gain) and the input diff-amp minimum common-mode range may become too large, causing the diff-amp to shut off when $VDD/2$ is applied to the op-amp input. This latter concern was discussed earlier. This fix for the settling time is trivial to simulate by changing the resistor value in the beta-multiplier biasing circuit used in the op-amp.

4. Increasing both the biasing current and the size of the devices in Fig. 33.70 so that a given load capacitance becomes, effectively, less difficult to drive. This is the same fix as 1), above, except that we still have the floating current source and output buffer (and so the current needed to charge the load is supplied by the push-pull amp and is not directly related to the biasing current in the diff-amp).

Before listing number 5, let's review from Ch. 27 how the op-amps unity gain frequency, f_u , is related to settling time. Assuming no slew-rate limitations, the op-amps time constant can be written (assuming the op-amp has only a single dominant pole) as

$$\tau = \frac{1}{2\pi f_u \cdot \beta} \quad (33.23)$$

β is the feedback factor. Here we assume $\beta = 1$, where all of the output is fed back to the input. For our op-amp response of Fig. 33.73 $\tau = 2.3$ ns. The output signal for our single-time constant dominant-pole op-amp circuit can be written as

$$V_{out} = V_{outfinal}(1 - e^{-t/2.3\text{ns}}) \quad (33.24)$$

For 0.1% settling accuracy ($V_{out}/V_{outfinal} = 99.9\%$), it takes

$$t = 15.9 \text{ ns} \quad (33.25)$$

While we used 70 MHz for f_u , including the 5 pF load drops the unity gain frequency and increases the bandwidth-limited settling time.

5. Use minimum channel lengths for any high-speed design. This increases the device's f_T while also increasing the drive current strength of the MOSFETs. Using minimum channel lengths with a larger biasing current can push the settling time down to under 10 ns. The decrease in the gain resulting from using minimum length devices and a larger biasing current (larger ΔV) in the topology of Fig. 33.70 shouldn't be too much of a concern since we are starting with 110 dB gain.

Differential Output Op-Amp

Let's build on our basic op-amp of Fig. 33.70 to implement a high-speed, low-power fully-differential op-amp. Consider the gain and output stage shown in Fig. 33.75 and the associated simplified schematic representation. Notice how the inputs labeled "Top" and "Bottom" are low-impedance, cascode-connected current mirrors (with node voltage labels, V_{ot1} and V_{ob1}). Using the simplified model, Figure 33.76 shows the schematic of a fully-differential op-amp (minus the common-mode feedback circuit).

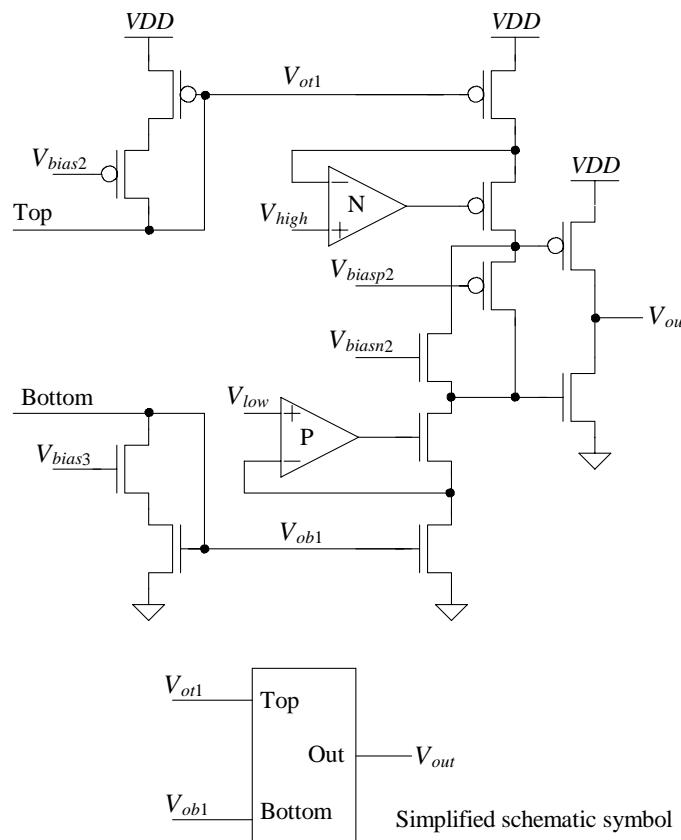


Figure 33.75 Mixed-signal op-amp building block.

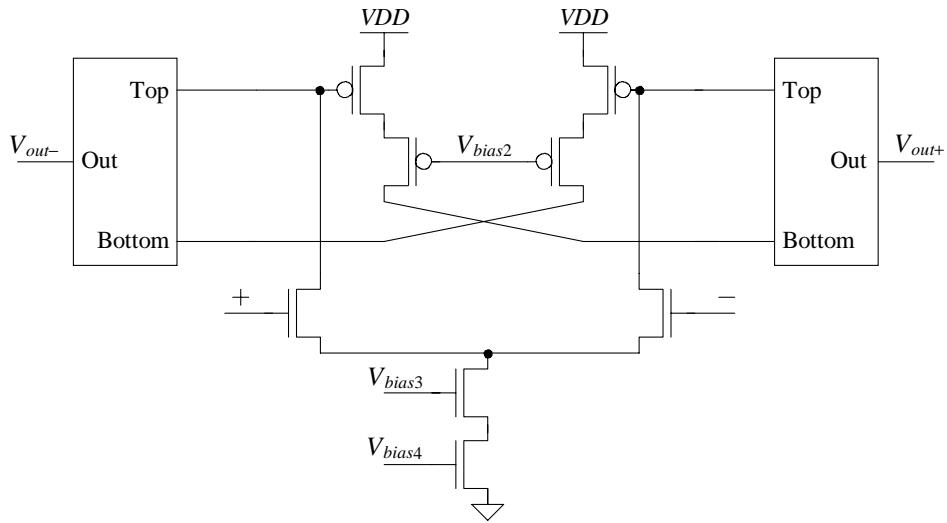


Figure 33.76 Fully-differential op-amp.

An important component of any fully-differential op-amp is the common-mode feedback circuit as discussed back in Ch. 25. While we won't repeat the material in Ch. 25 here, we will comment on two additional methods to balance the op-amp outputs.

Figure 33.77 shows the addition of two weak transistors to the input of the diff-amp of the op-amp. If the op-amp's inputs are not at the common-mode voltage, V_{CM} , the additional transistors either conduct more or less current causing the average of the outputs of the op-amp to move upwards or downwards. This, of course, assumes the output of the diff-amp is single-ended. This would require connecting the "Bottom" input node in the building block stage to V_{bias4} and removing the diode-connected MOSFETs on the input of this stage.

Earlier we discussed using common-mode noise elimination to balance the outputs of a digital input buffer, Fig. 33.39. We can use the same technique to balance the output of our op-amp, Fig. 33.78. We will use inverters to illustrate the technique; however, we must ensure that whatever inverting amplifier we place on the output of the op-amp doesn't load the op-amp's outputs. Adding resistors in series with the inverter outputs will help ensure loading isn't a problem. Since our op-amp outputs are buffered, the op-amp can drive the resistors directly. Using diff-amps in place of the inverters may be necessary to precisely set the common-mode output voltage to V_{CM} . The inverters, as shown in Fig. 33.78, will try to balance the outputs to their switching point voltage, V_{SP} . When a diff-amp is used in place of the inverters, one input is tied to the resistors and the other input is tied to V_{CM} . Also note, for high-frequency operation, the averaging resistors

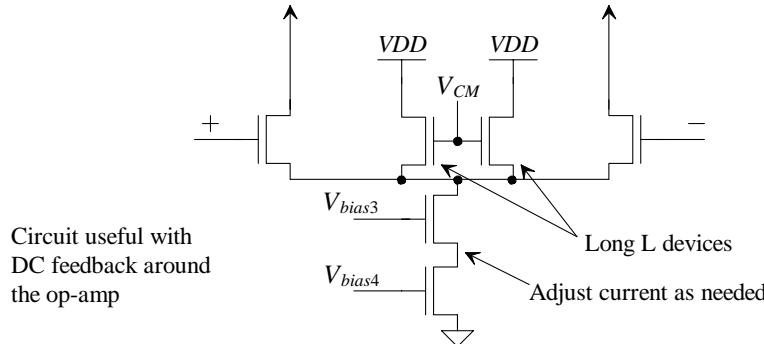


Figure 33.77 Adding auxiliary input to the diff-amp to balance the op-amp's outputs.

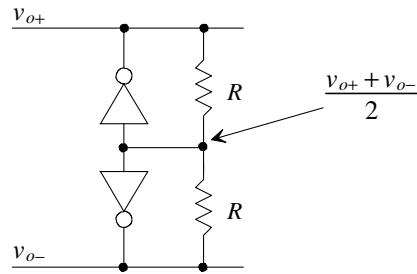


Figure 33.78 Using common-mode noise elimination to balance the outputs of an op-amp.

should have small shunt capacitors placed across their terminals to compensate for the inverters' input capacitance.

We can analyze the operation of the circuit shown in Fig. 33.78 using the steps shown in Eqs. (33.5)-(33.12). Assuming the output resistance of the inverter is large compared to the resistors R , we get

$$i_{diff} = g_m \cdot \frac{v_{o+} + v_{o-}}{2} + \frac{v_{o+} - (v_{o+} - v_{o-})/2}{R} + i_{CM} \quad (33.26)$$

and

$$-i_{diff} = g_m \cdot \frac{v_{o+} + v_{o-}}{2} + \frac{v_{o-} - (v_{o+} - v_{o-})/2}{R} + i_{CM} \quad (33.27)$$

Taking the difference in these equations shows, once again,

$$2i_{diff} = \frac{v_{o+} - v_{o-}}{R} \quad (33.28)$$

and that the common-mode component of the signal is removed (set to V_{SP}).

33.3.3 Circuit Noise

In this section we discuss and review circuit noise. We will repeat discussions of some of the topics we've already covered in Chs. 7, 9, and 22 to provide further, intuitive understanding of random processes (such as the quantization noise and clock jitter discussed in Chs. 30 and 31).

Thermal Noise

Consider the voltage divider shown in Fig. 33.79a. In this figure the ideal voltage out, V_{out} , is 0.75. However, because of the random motion of the electrons making up the current flowing in the circuit (because of lattice vibrations or heat), the voltage V_{out} has a random variation. The effects of this variation are termed thermal noise. As seen in Fig. 33.79b, the variation in the output voltage can be characterized with a Gaussian probability density function (PDF). Notice that time is not indicated in this figure. We can't determine directly, from the information shown in the figure, the spectral characteristics of the noise unless we make some assumptions (like we did with the quantization noise in the previous chapters where we assumed it was bandlimited to the Nyquist frequency). Reviewing Figs 31.12 and 31.13 and the associated discussions in Ch. 31 we can make the following comments:

1. The RMS value of the thermal noise in Fig. 33.79 is $1 \mu\text{V}$. We could also say that the standard-deviation, σ , is $1 \mu\text{V}$ since the RMS value and the standard deviation are equal when the noise has a Gaussian PDF.

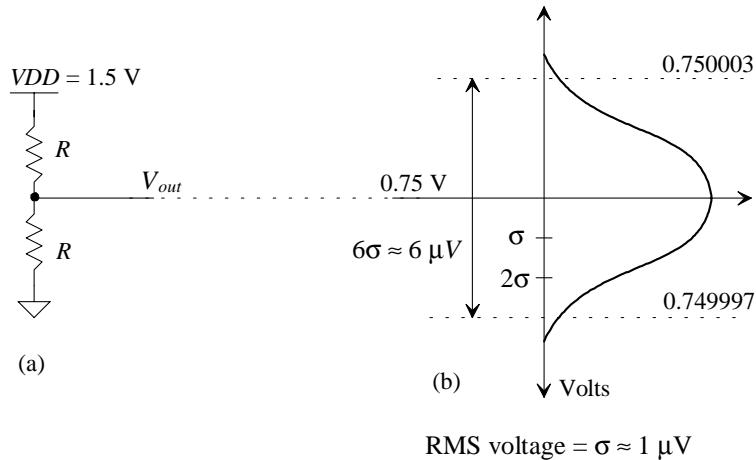


Figure 33.79 (a) A voltage divider and (b) the variation of the output voltage because of thermal noise (an example PDF).

2. The mean-squared value (squaring the RMS value) indicates the average power of the thermal noise. We could also say that the variance, σ^2 , indicates the average power of the noise. Sometimes this is more correctly called the total average *normalized* power because we assume a 1Ω resistor when converting from RMS voltage or current to power and we sum the total power in the spectrum, see Eq. (31.37).

3. The power (variance) in uncorrelated noise sources (random variables) can be added directly, but the RMS voltage or current values (standard-deviation) cannot. For example, if we have an RMS thermal noise voltage, σ_{therm} , an RMS quantization noise voltage, $V_{Qe,RMS}$, and a sampling error power due to jitter of $P_{AVG,jitter}$ (see Ex. 31.15) corrupting a sinewave signal with a peak amplitude of V_p , then we would determine the signal-to-noise ratio of the signal using

$$SNR = 20 \cdot \log \frac{V_p/\sqrt{2}}{\sqrt{\sigma_{therm}^2 + V_{Qe,RMS}^2 + P_{AVG,jitter}}} \quad (33.29)$$

The numerator is, of course, the RMS value of the desired signal while the denominator is the square root of the total error power from each error source, i.e., thermal noise, quantization noise, and clock jitter. It's interesting to note that averaging K samples of a random variable, say thermal noise, results in a reduction of its RMS value

$$\sigma_{K,therm} = \frac{\sigma_{therm}}{\sqrt{K}} \text{ or } \sigma_{K,therm}^2 = \frac{\sigma_{therm}^2}{K} \quad (33.30)$$

The shape of the Gaussian PDF gets taller and narrower as we average the random signal. The area, however, remains constant and equal to one. Rewriting Eq. (33.29) and including the effects of averaging the signal and noise results in

$$SNR = 20 \cdot \log \frac{V_p/\sqrt{2}}{\sqrt{\sigma_{therm}^2 + V_{Qe,RMS}^2 + P_{AVG,jitter}}} + 10 \cdot \log K \quad (33.31)$$

which shows, once again, that averaging can be employed to increase SNR.

The Spectral Characteristics of Thermal Noise

Examine Fig. 33.80 where we've eliminated the DC bias and have shown the random current sources used to model each resistor's thermal noise contributions to V_{out} . The RMS noise current has a value (as given in Ch. 7) of

$$\sqrt{i^2} = \sqrt{\frac{4kT}{R} \cdot B} \quad (33.32)$$

where k is Boltzmann's constant (1.38×10^{-23} Watt·sec/°K), T is the temperature in Kelvin, and B is the bandwidth over which the noise is measured. For the circuit shown in Fig. 33.80, the RMS output noise voltage is

$$\sigma_{therm} = V_{out,RMS} = (R_1 || R_2) \cdot \sqrt{\frac{4kT}{R_1} \cdot B + \frac{4kT}{R_2} \cdot B} \quad (33.33)$$

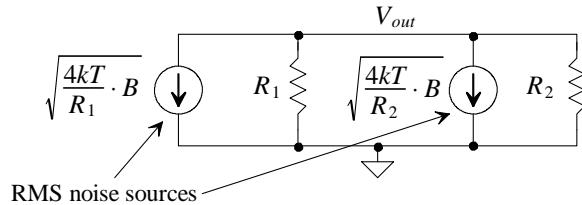


Figure 33.80 Modeling thermal noise in a resistor.

As indicated in Ch. 7, to perform a noise analysis we (1) add RMS noise voltages to the circuit, (2) determine the RMS output noise from each contribution using superposition (i.e., with only one noise source in the circuit at a time), and (3) square each contribution followed by summing and taking the square-root to get the output noise (as a function of the bandwidth B).

Note that even with both sides of a resistor connected to ground (no DC current), a noise current (electrons) will move back and forth from the ground to the resistive material at each connection as long as $T > 0$ K. In other words, a noise current will still be present in the circuit.

Reviewing Eq. (33.32) for a moment would reveal that if we reduce the bandwidth of a measurement (pass the signal plus noise through a narrow band filter) we get a corresponding reduction in the RMS noise present in the signal. Spectrum analyzers use narrow band filtering for this reason: to get extremely large dynamic range. The PSD of the thermal noise voltage specified by Eq. (33.33) is plotted in Fig. 33.81. The output noise power is given by

$$P_{AVG} = \sigma_{therm}^2 = V_{out,RMS}^2 = \int_{f_L}^{f_H} P_{therm}(f) \cdot df = 4kT \cdot (R_1 || R_2) \cdot B \quad (33.34)$$

where

$$B = f_H - f_L \quad (33.35)$$

We now need to discuss how to determine the bandwidth, B .

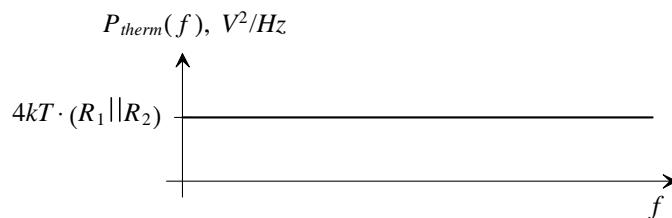


Figure 33.81 Thermal noise power spectral density for Eq. (33.33).

Noise Equivalent Bandwidth

Any real circuit will not operate over the infinite bandwidth indicated in Fig. 33.81. Consider the simple RC lowpass filter shown in Fig. 33.82. The noise circuit, after using superposition (shorting the input voltage source, V_{in} , to ground) is also seen in this figure. Note that a capacitor doesn't generate noise (although, as we'll see in a moment, it does limit the RMS output noise in the circuit).

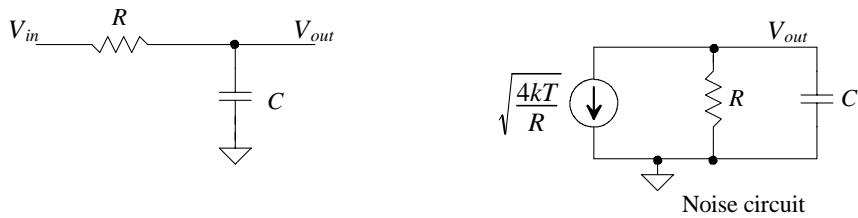


Figure 33.82 Circuit used to determine NEB.

The output noise power for the circuit of Fig. 33.82 can be determined using

$$V_{out,RMS}^2 = \int_0^\infty 4kTR \cdot \frac{1}{1 + (2\pi f \cdot RC)^2} \cdot df \quad (33.36)$$

Knowing

$$\int \frac{du}{a^2 + u^2} = \frac{1}{a} \tan^{-1} \frac{u}{a} + C \quad (33.37)$$

then

$$V_{out,RMS}^2 = 4kTR \cdot \underbrace{\frac{1}{2\pi \cdot RC}}_{f_{3dB}} \cdot [\tan^{-1} 2\pi f \cdot RC]_0^\infty \quad (33.38)$$

or

$$V_{out,RMS}^2 = 4kTR \cdot f_{3dB} \cdot \frac{\pi}{2} \quad (33.39)$$

The noise equivalent bandwidth (NEB) is then

$$NEB = f_{3dB} \cdot \frac{\pi}{2} \quad (33.40)$$

Figure 33.83 shows the interpretation of this equation. The area from DC to the NEB is equal to the area under the actual response. Note that Eq. (33.39) reduces to

$$V_{out,RMS}^2 = \frac{kT}{C} \quad (33.41)$$

which is our familiar result for the RMS thermal output noise power of an RC circuit (key tee over cee noise).

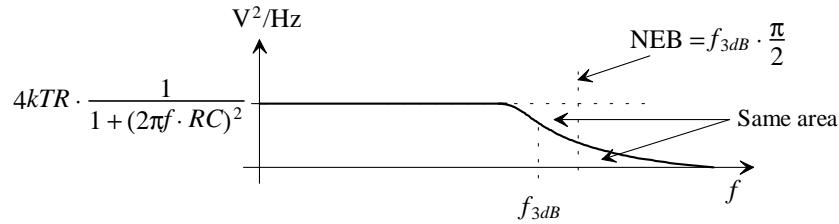


Figure 33.83 Showing the same area in the NEB and the actual spectrum.

As an example of where we can use NEB consider the dominant-pole-compensated op-amp in a unity follower configuration shown in Fig. 33.84a. It's important to note that noise is always measured on the output of a circuit and then referred back to the input, Fig. 33.84b. Also note that we are assuming the op-amp can drive the possibly low input resistance of the spectrum analyzer. If not, as we'll see when characterizing the noise performance of MOSFETs, we'll need to introduce a low-noise amplifier (LNA) in between the circuit-under-test and the spectrum analyzer for isolation.

To calculate the RMS input-referred noise voltage, we can remember that the unity gain frequency of an op-amp, f_u , is related to the op-amp's 3-dB frequency and open loop

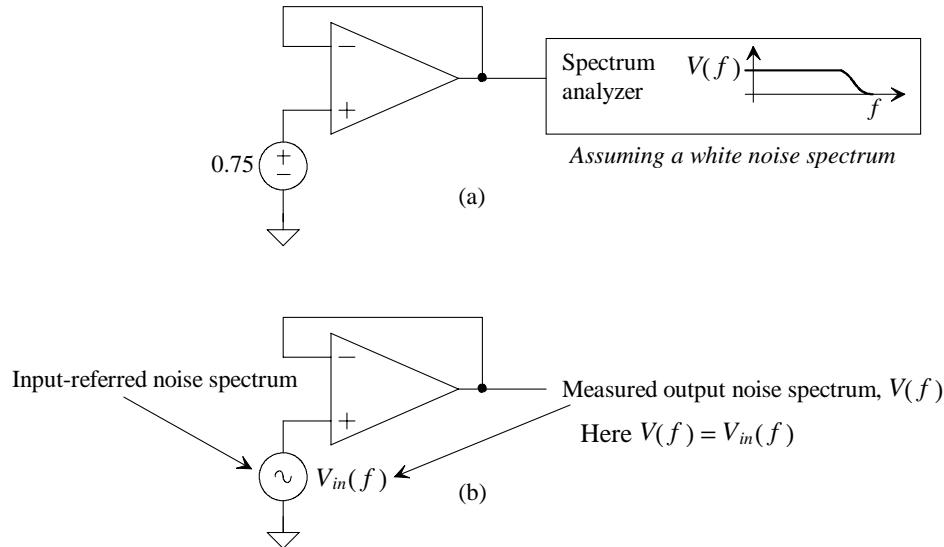


Figure 33.84 (a) Measuring op-amp output noise spectral density and, (b) noise model where output noise is referred back to the input.

gain, A_{OL} , using $f_{3dB} = f_u/A_{OL}$. Assuming the op-amp is the limiting bandwidth factor in the circuit, the RMS input-referred noise is given by

$$V_{in,RMS} = \sqrt{V(f) \cdot \underbrace{\frac{f_u}{A_{OL}} \cdot \frac{\pi}{2}}_{\text{NEB}}} \quad (33.42)$$

This assumes $V(f)$ is a constant magnitude number (white noise) vs. frequency. (White noise is analogous to white light where the spectrum of white light is occupying all frequencies of interest with equal amplitude.)

In a CMOS op-amp we'll be able to use NEB to get an idea for the circuit noise, especially if the circuit bandwidth is wide. However, because of flicker noise ($1/f$ noise discussed in Ch. 9) present in MOSFETs we will need to use Eq. (33.34) to get an exact idea for the output RMS noise. Equation (33.34) is rewritten as

$$V_{out,RMS}^2(f) = \int_{f_L}^{f_H} V^2(f) \cdot df \quad (33.43)$$

where now the measured noise output spectrum, $V(f)$, is not a constant but changes with frequency. The RMS input-referred noise can be determined for a particular circuit configuration using

$$V_{in,RMS}^2(f) = \int_{f_L}^{f_H} \frac{V^2(f)}{|H(f)|^2} \cdot df \quad (33.44)$$

where $H(f)$ is the transfer function of the circuit.

MOSFET Noise

The noise mechanisms present in MOSFETs (thermal and $1/f$) were discussed back in Ch. 9. Figure 33.85 shows how a low-noise amplifier (LNA) keeps the spectrum analyzer from loading the MOSFET-under-test and is used to set the drain voltage of the MOSFET. In order to simplify the calculations in the following discussions, we will write the total noise

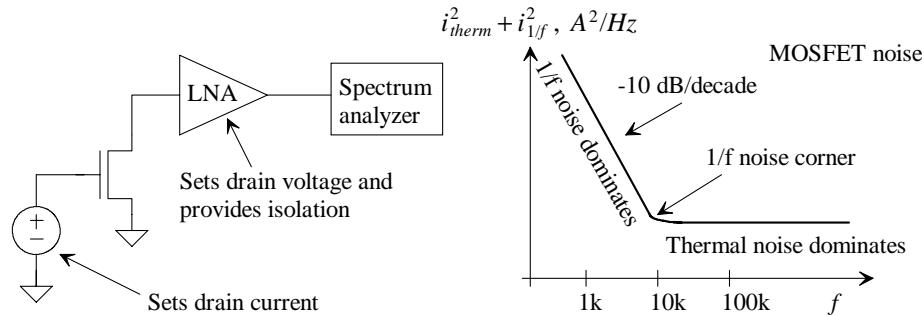


Figure 33.85 Determining MOSFET noise.

power of both contributions from thermal noise and 1/f noise (flicker) in a MOSFET's drain current as i_{noise}^2 or

$$i_{noise}^2 = i_{therm}^2 + i_{1/f}^2 \quad (33.45)$$

Because noise is always measured on the output of a circuit and referred back to the input for comparison with the input signal we can use either of the circuits shown in Fig. 33.86 for performing simple noise analyses in our CMOS circuits.

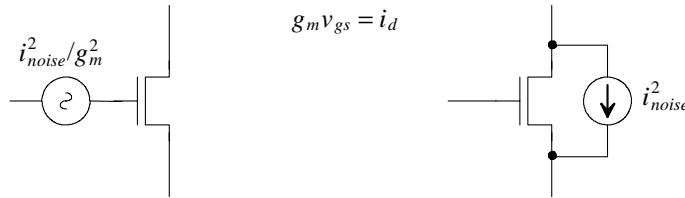


Figure 33.86 Modeling MOSFET noise.

Noise Performance of the Source-Follower

Examine the source-follower configuration shown in Fig. 33.87. The input-referred noise voltage (the noise added to the input signal) is

$$V_{in}^2(f) = (i_{noise1}^2 + i_{noise2}^2) \cdot \left(\frac{1}{g_{m1}^2} + r_o^2 \right) \quad (33.46)$$

By increasing g_{m1} (making the gain of the source-follower approach one), the input-referred noise can be minimized (ultimately approaching the measured output noise $[i_{noise1}^2 + i_{noise2}^2] \cdot r_o^2$). This statement alone isn't too useful because increasing the biasing current flowing in M1 (and M2) will increase g_{m1} but not necessarily decrease the

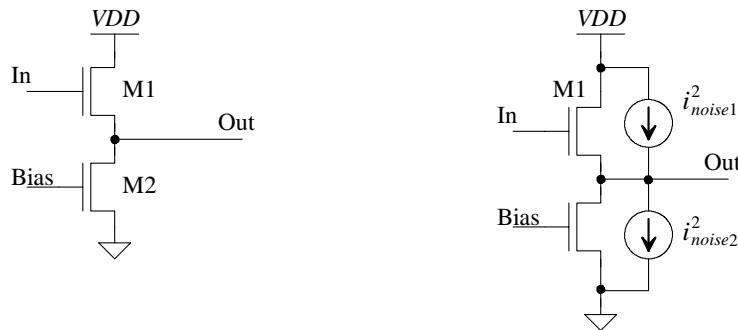


Figure 33.87 Noise performance of the source-follower.

input-referred noise. Increasing the biasing current will also cause the noise currents to increase (as discussed in Ch. 9). To increase g_{m1} without changing the noise currents we must increase the width of M1. While increasing the width improves the noise performance, it slows down the inherent speed of the circuit because of the drop in ΔV (and thus f_T). However, speed is usually not a concern with a source-follower (under light to reasonable load conditions) because its bandwidth approaches f_T . *The point here is that increasing the width of the MOSFET whose gate is connected to an input node can be used in any amplifier to reduce the input-referred noise assuming constant drain current.*

An important use of the source-follower is in small-input capacitance amplifiers (such as charge amplifiers used in charge-coupled devices [CCDs]). Because the AC input voltage is ideally equal to the AC output voltage, the voltage change across the gate-source capacitance is zero. This means that the input capacitance of the source-follower in Fig. 33.87 is set by the gate-drain capacitance of M1 (and, compared to a common-source amplifier, is considerably smaller). However, because the voltage gain of the source-follower is one the input-referred noise contributions from the amplifier connected to the output of the source-follower are referred directly back to the source-follower's input without any amplitude reduction. To understand this statement in more detail, let's consider the noise performance of a cascade of amplifiers.

Noise Performance of a Cascade of Amplifiers

Consider the cascade of amplifiers shown in Fig. 33.88. Here we are assuming the amplifiers have infinite input resistance (the amplifier input is the gate of a MOSFET; the amplifiers amplify an input voltage). If this isn't the case, then we need to model the input-referred noise with both voltage and current generators to account for the loading on the amplifier output. As seen in the figure, referring all three stage's noise contributions back to the overall amplifier input results in

$$V_{in}^2(f) = V_1^2(f) + \frac{V_2^2(f)}{A_1^2} + \frac{V_3^2(f)}{A_1^2 A_2^2} \quad (33.47)$$

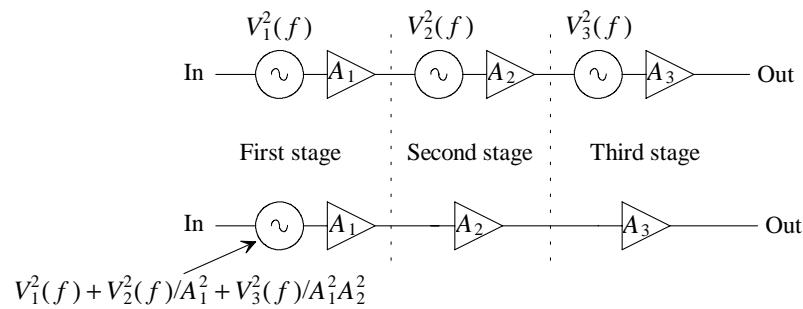


Figure 33.88 Noise performance of a cascade of amplifiers.

Stage two's input-referred noise, $V_2^2(f)$, is referred back to the overall amplifier's input by dividing by the first stage's gain, A_1^2 . If the first stage's gain is one then the second-stage's input-referred noise appears directly on the overall amplifier's input (which is the case when using a source-follower as the first amplifier). However, if the first stage has a large gain, then the contributions from the following stages are negligible. *The point here is that to minimize the input-referred noise the gain of the first stage should be large.*

For the basic op-amp shown in Fig. 33.70 the noise performance is dominated by both the first-stage diff-amp and the second-stage amplifier made up of the cascoded transistors and floating-current source. The voltage gain of the diff-amp will be close to one making the second stage's input-referred noise reflect directly back to the input of the op-amp. Again, for low-noise design, we want large first-stage gain.

To minimize a diff-amp's input-referred noise (see Ch. 24), we can increase the widths of the diff-pair. As discussed earlier, this also minimizes both the systematic and random offsets in an op-amp and increases the input common-mode range. The cost of these benefits, again, is the lowering of the parasitic poles (ultimately affecting op-amp stability) that the diff-pair introduces into the op-amp's overall transfer function. The f_T of the two MOSFETs used in the diff-pair decreases.

It's interesting to note that an amplifier's offset voltage can be thought of as a special case of noise at DC. This means that Eq. (33.47) can be applied to determine the importance of amplifier offset in a cascade of amplifiers. (Replace $V[f]$ with V_{os} in Eq. [33.47].) Note that because of the small voltage gain of the diff-pair used in our mixed-signal op-amp of Fig. 33.70, the topology can have a relatively large systematic offset voltage (e.g. 5 mV). The noise and offset performance of this op-amp, however, is no worse than that of the folded-cascode amplifiers presented in Ch. 25. Folded cascode-based op-amps also use a low-voltage gain in the input diff-pair. For circuit techniques to reduce both noise and offsets (chopper stabilization, offset storage, and correlated double sampling) the reader is referred to [9].

DAI Noise Performance

Figure 33.89 shows the DAI (see Fig. 31.78) with kT/C noise sources shown. The input-referred noise is given by

$$V_{in,RMS}^2 = \frac{kT}{C_I} \quad (33.48)$$

in series with both v_1 and v_2 . A total of $2kT/C_I$ is sampled onto C_I during each clock cycle. If the input signal can swing from VDD to ground, then we can estimate the SNR using

$$\text{SNR} = 20 \log \frac{VDD/(2\sqrt{2})}{\sqrt{2kT/C_I}} \quad (33.49)$$

If $VDD = 1.5$ V, $T = 300$ K, and $C_I = 100$ fF then the maximum SNR is 68 dB (roughly 11-bits of resolution). Equation (33.49) is useful in determining the minimum value of capacitors used in a DAI for a specific application.

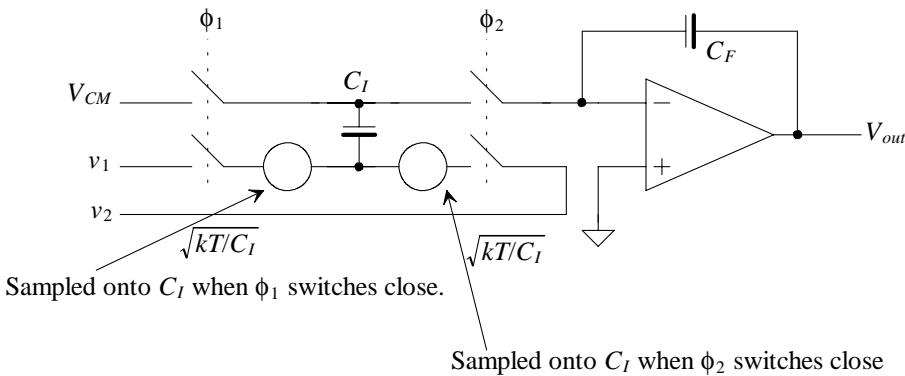


Figure 33.89 Noise performance of the DAI.

REFERENCES

- [1] R. J. Baker, H. W. Li, and D. E. Boyce, *CMOS: Circuit Design, Layout, and Simulation*, Wiley-IEEE, 1998. ISBN 0-7803-3416-7
- [2] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*, Cambridge University Press, 1998. ISBN 0-521-55959-6
- [3] C. Enz, F. Krummenacher, and E. Vittoz, "An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications," *Journal on Analog Integrated Circuits and Signal Processing*, Kluwer Academic Publishers, pp. 83-114, July 1995
- [4] M. Bucher, C. Lallement, C. Enz, F. Théodolz, and F. Krummenacher, Electronics Laboratories, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. Available at <http://legwww.epfl.ch/ekv/index.html>
- [5] D. P. Foty, *MOSFET Modeling with SPICE: Principles and Practice*, Prentice-Hall, 1997. ISBN 0-13-227935-5
- [6] D. I. Hariton, *Floating MOS Capacitor*, U.S. Patent 5,926,064, July 20, 1999.
- [7] J. Yuan and C. Svenson, "High-Speed CMOS Circuit Technique," *IEEE Journal of Solid State Circuits*, Vol. 24, No. 1, pp. 62-70, February 1989.
- [8] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching Properties of MOS Transistors," *IEEE Journal of Solid State Circuits*, Vol. 24, No. 5, pp. 1433-1440, October 1989.
- [9] C. C. Enz and G. C. Temes, "Circuit Techniques for Reducing the Effects of Op-Amp Imperfections: Autozeroing, Correlated Double Sampling, and Chopper Stabilization," *Proceedings of the IEEE*, Vol. 84, No. 11, pp. 1584-1614, November 1996. Available at <http://cmosedu.com>

QUESTIONS

- 33.1** Regenerate the plots shown in Fig. 33.1 using an L of 5 μm and a W of 15 μm . Do the curves look similar? Could we use the level 1 model with long L devices?
- 33.2** Can we use a native MOSFET in an application where we need a small threshold voltage? What are the limitations?
- 33.3** Sketch the implementation of a floating capacitor that uses a MOSFET switch to connect the source/drain/bulk to ground. Show the capacitor used in a DAI, Fig. 31.78. Assume the switch connected to the capacitor is clocked with the ϕ_1 clock (why?). Explain the operation of the circuit.
- 33.4** Sketch the practical layout of a 10k n+ poly resistor using a silicide block.
- 33.5** Verify the error in Fig. 33.18 is due to differing device drain-source voltages. Show that using an even longer length device can result in less error.
- 33.6** If the drawn area of a source implant is 100 m^2 , what is the actual area if a scale factor of 0.15 μm is used.
- 33.7** Using Eqs. (33.3) and (33.4), estimate the high-to-low and low-to-high delays in the circuits shown in Fig. 33.90.



Figure 33.90 Circuits used in problem 33.7.

- 33.8** Using WinSPICE results, tabulate the output amplitude of the circuit in Fig. 33.32 against capacitive load.
- 33.9** Show that kickback noise on the inputs of the comparator shown in Fig. 33.34 using simulations. Show adding the circuit of Fig. 33.91 (a source-follower) to each input of the comparator will drastically reduce kickback noise.
- 33.10** Modify the design of the digital input buffer shown in Fig. 33.39 using the topology shown in Fig. 33.78. Show that the circuit still functions as expected using simulations.
- 33.11** Show, using simulations, that the circuits in Fig. 33.47 do indeed behave as counters.
- 33.12** Verify the operation of the element in Fig. 33.50 as a 1-bit adder.

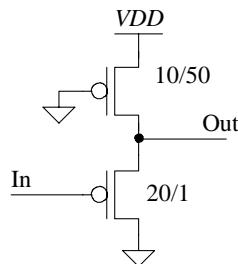


Figure 33.91 A source-follower used to reduce kickback noise.

- 33.13** Do the capacitors in Fig. 33.51 slow down the operation of the diff-amp? Why?
- 33.14** Estimate the input common-mode range of the diff-amp shown in Fig. 33.51.
- 33.15** Regenerate the plots shown in Figs. 33.54 and 33.55 if the drain-source voltages of the MOSFETs are increased to 1.5 V. Why did the transconductance increase?
- 33.16** Show, using simulations, that a MOSFET's transition frequency does increase with increasing gate-source voltage and decreasing length.
- 33.17** Do any of the MOSFETs in Fig. 33.58 move into the triode region if the resistor's value is decreased to 1k? Verify your answers with simulations.
- 33.18** Compare the cascode current mirror performance shown in Fig. 33.61 to a basic current mirror. Show the increase in output resistance when using a cascode and compare the minimum voltages across the mirrors.
- 33.19** What happens in Fig. 33.65 to the current flowing in the push-pull amplifier if we increase the lengths of MC1 and MC2?
- 33.20** Estimate the minimum VDD allowed for proper operation of the diff-amps shown in Fig. 33.69.
- 33.21** If, in Fig. 33.70, $V_{GS} = V_{SG} = 0.5$ V and $\Delta V = V_{DS,\min} = 0.1$ V what is the minimum allowable power supply voltage, VDD , for proper op-amp operation.
- 33.22** Sketch the implementation, based on the topology of Fig. 33.70, of a wide-swing op-amp with rail-to-rail input common-mode range. Assume the tail currents used in the diff-amps use 20/2 (NMOS) and 40/2 (PMOS) devices (half the current flowing each MOSFET of the diff-pair) so that summing the currents in the circuit of Fig. 33.75 at the top and bottom nodes doesn't cause any MOSFET to enter the triode region.
- 33.23** Resketch Fig. 33.79 for the cases when the thermal noise is averaged. Show the cases with $K = 1, 2, 4, 8$, and 16.

- 33.24** Suppose a MOSFET is used as a switch connecting an input signal to a capacitor. The MOSFET can be modeled, for noise purposes while the switch is on, as a simple resistor, Fig. 33.82. When the capacitor is charged, zero current flows in the MOSFET. Is the noise in the circuit due to thermal or $1/f$ (flicker) noise? Why? Note that each time the MOSFET turns on we can think that an RMS noise voltage of $\sqrt{kT/C}$ is sampled on the capacitor.

Chapter

34

Implementing Data Converters

Minimum gate lengths in CMOS technology are falling below 100 nm [1]. This feature size reduction is driven mainly by the desire to implement digital systems of increased complexity in a smaller area. This natural trend in feature size reduction, with accompanying reduction in supply voltage, can present challenges for the mixed-signal design engineer. The accompanying lower supply voltage results in an inherent reduction in dynamic range, decrease in SNR, and increasing challenges when implementing analog circuitry with little, ideally zero, voltage overhead. This chapter focuses on these issues, and others, related to the implementation of data converters in a digital, submicron CMOS technology.

Data converters are fundamental building blocks in mixed-signal systems. This chapter presents and discusses methods and trade-offs for designing CMOS data converters in submicron CMOS. For DAC design, we focus on converters implemented with resistors using R - $2R$ networks. The benefit of, and reason we are focusing on, using R - $2R$ networks over other methods for DAC implementation, such as charge redistribution [2] or current steering topologies [3], is the absence of good poly-poly capacitors in a digital CMOS process, the desire for small layout area, and/or the ability to drive an arbitrary load resistance. R - $2R$ -based DACs can be laid out in a small area while achieving resolutions in excess of 12-bits. Charge-scaling DACs require linear capacitors. The layout area needed for these capacitors can often be very large and practically limit both the resolution and accuracy of the DAC. Similarly, the implementation of current-steering topologies can result in a very large layout area with limited resolutions, generally less than 8 bits if integral nonlinearity (INL) is a concern and, more importantly, if limited output swing and the requirement of known, fixed-load resistances are a concern.

The first section of this chapter reviews current- and voltage-mode R - $2R$ -based DACs. The second section of the chapter discusses the use of op-amps in data converters, while the third section presents an overview of general ADC implementations in

submicron CMOS process. While we briefly discuss the future direction of ADCs, we concentrate our discussion on the implementation of pipeline data converters.

The goal in this chapter is not to provide an exhaustive overview of data converter design but rather to provide discussions and practical insight. We assume that the reader is familiar with data converter fundamentals (discussed in Ch. 28) and data converter architectures (discussed in Ch. 29). For example, the reader knows the difference between differential nonlinearity (DNL) and integral nonlinearity (INL) or the difference between a two-step flash ADC and a pipeline ADC.

34.1 R-2R Topologies for DACs

We begin this section by discussing R - $2R$ DAC topologies. The problems encountered in the traditional R - $2R$ topologies with low-voltage overhead are illustrated. Also, concerns related to the performance of the op-amps used in data converters (both ADCs and DACs) are discussed. Finally, matching and accuracy concerns are presented along with techniques to remove these imperfections using calibration.

34.1.1 The Current-Mode R-2R DAC

The R - $2R$ DAC can be classified into two categories: voltage-mode and current-mode [4]. A current-mode R - $2R$ DAC is shown in Fig. 34.1. The branch currents flowing through the $2R$ resistors are of a binary-weighted relationship caused by the voltage division of the R - $2R$ ladder network and are diverted either to the inverting input of the op-amp (actually the feedback resistor) or the noninverting input of the op-amp (actually V_{REF-}). The voltage on the R - $2R$ resistor string at the X^{th} tap (where X ranges from 0 to $N - 1$), in Fig. 34.1, can be written as

$$V_{TAPX} = \frac{2^X}{2^N} \cdot (V_{REF+} - V_{REF-}) + V_{REF-} \quad (34.1)$$

where V_{REF+} and V_{REF-} are the N -bit DAC's reference voltages. The current that flows through the $2R$ resistor at the X^{th} tap is then

$$I_{TAPX} = \frac{V_{TAPX} - V_{REF-}}{2R} = \frac{1}{2R} \cdot \frac{2^X}{2^N} (V_{REF+} - V_{REF-}) \quad (34.2)$$

This current is summed at the inverting input of the op-amp and flows through the feedback resistor to the DAC output, V_{out} . The output voltage of the DAC can be written as

$$V_{out} = V_{REF-} + R \cdot \sum_{X=0}^{N-1} (b_X \cdot I_{TAPX}) \text{ for } V_{REF+} > V_{REF-} \quad (34.3)$$

where b_X is either a 1 or 0, or

$$V_{out} = V_{REF-} + R \cdot \sum_{X=0}^{N-1} (b_X \cdot I_{TAPX}) \text{ for } V_{REF+} < V_{REF-} \quad (34.4)$$

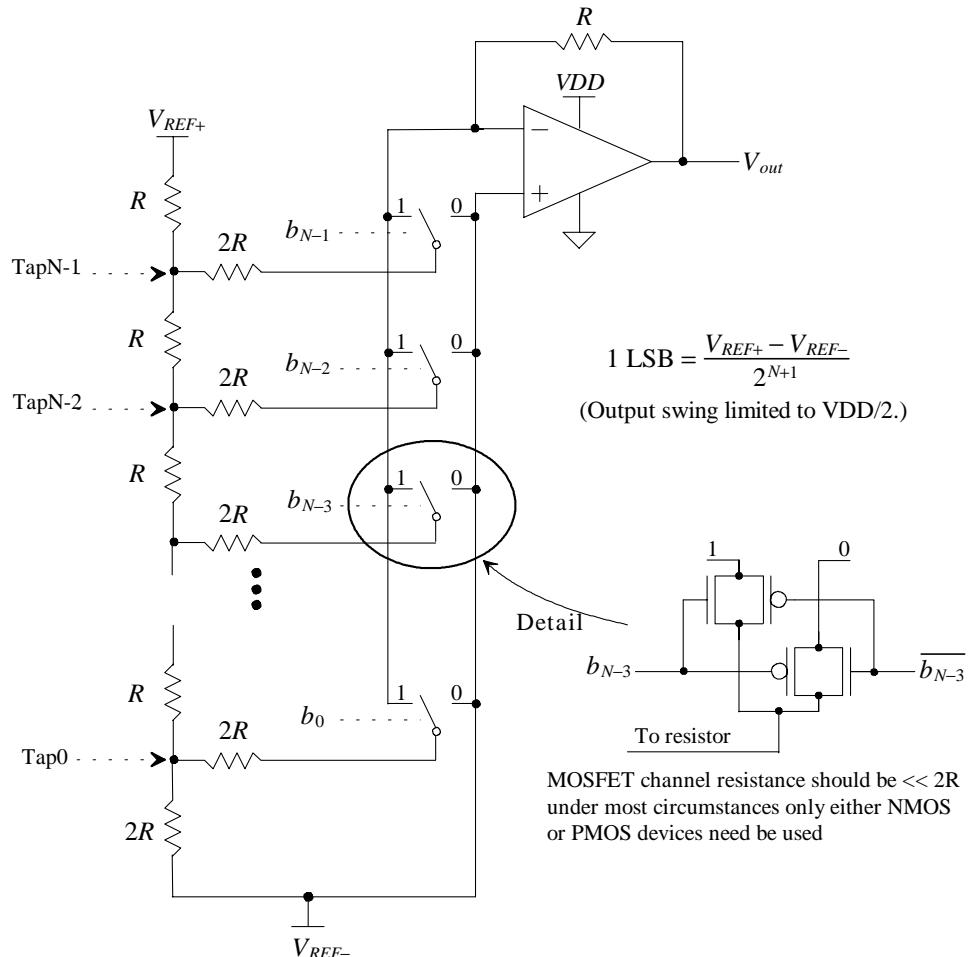


Figure 34.1 Traditional current-mode R-2R DAC.

Using these equations, we can see the main problem with the basic current mode topology of Fig. 34.1 in a submicron CMOS process using low-power supply voltages, namely, limited output swing. If V_{REF-} is set to 0 V, with $V_{REF+} > 0$, then the output of the DAC must be negative, which, of course, can't happen when the only power supply voltage is VDD . If V_{REF-} is set to VDD , then we can see from Eq. (34.4) that this would require $V_{out} > VDD$. After reviewing Eqs. (34.1)-(34.4), we see that the range of output voltages associated with the current mode R-2R DAC is limited to $VDD/2$, e.g., 0 to $VDD/2$, $VDD/2$ to VDD or 0.25 VDD to 0.75 VDD , etc. Giving up half of the power-supply range in a DAC and correspondingly reducing the dynamic range, is usually not desirable.

By removing the requirement that the noninverting input of the op-amp be tied to V_{REF-} and that the feedback resistor be R (the same value used in the R - $2R$ string), we can increase the output range of the DAC. The output of the op-amp is level-shifted by the voltage on the noninverting input of the op-amp and by increasing the closed-loop gain of the op-amp. Similarly, we could add a gain stage to the output of the DAC (two op-amps would then be used) to achieve wider DAC output swing. We don't cover these options any further here because they either put more demand on the op-amp design, such as increased op-amp open-loop gain and speed, or won't, in a practical implementation, result in a rail-to-rail output swing.

34.1.2 The Voltage-Mode R - $2R$ DAC

Figure 34.2 shows a schematic of a voltage-mode DAC. The voltage on the non-inverting input of the op-amp can be written as

$$V_+ = \frac{b_{N-1} \cdot V_{REF+} + \overline{b_{N-1}} \cdot V_{REF-}}{2^1} + \frac{b_{N-2} \cdot V_{REF+} + \overline{b_{N-2}} \cdot V_{REF-}}{2^2} + \dots + V_{REF-} \quad (34.5)$$

or, in general terms,

$$V_+ = \sum_{k=1}^N \frac{b_{N-k} \cdot V_{REF+} + \overline{b_{N-k}} \cdot V_{REF-}}{2^k} + V_{REF-} \quad (34.6)$$

The output of the N -bit voltage-mode DAC can be written as

$$V_{out} = \left[1 + \frac{R_F}{R_I} \right] \cdot \left[\sum_{k=1}^N \frac{b_{N-k} \cdot V_{REF+} + \overline{b_{N-k}} \cdot V_{REF-}}{2^k} + V_{REF-} \right] \quad (34.7)$$

If the input code is all zeroes, with $V_{REF-} = 0$, $V_{REF+} = VDD$, and the op-amp in the follower configuration, then $V_{out} = V_{REF-}$. If the input code is all ones, then the output of the DAC is $V_{REF+} - 1$ LSB.

By using the voltage-mode DAC, we would seem to have solved the problem of the limited output swing associated with the current-mode DAC of Fig. 34.1. However, consider how the finite common-mode rejection ratio (CMRR) of the op-amp in Fig. 34.2 can affect the linearity of the overall DAC design. We know the effects of finite CMRR can be modeled as a variable offset voltage, ΔV_{OS} (see Ch. 25), in series with the noninverting input of the op-amp that is a function of the change in the op-amp common-mode voltage, ΔV_C , or

$$\Delta V_{OS} = \frac{\Delta V_C}{CMRR} \quad (34.8)$$

We should see the problem at this point: that is, ΔV_{OS} is in series with the R- $2R$ resistor string and will ultimately limit the linearity of the DAC. To further illustrate the problem, let's assume the CMRR of the op-amp in Fig. 34.2 is 20 dB at 1 MHz. Since the common-mode voltage on the input of the op-amp, again assuming $V_{REF+} = VDD$, $V_{REF-} = 0$, and the op-amp in the unity follower configuration can range from zero to approximately VDD , the change in the offset voltage used to model finite CMRR when the

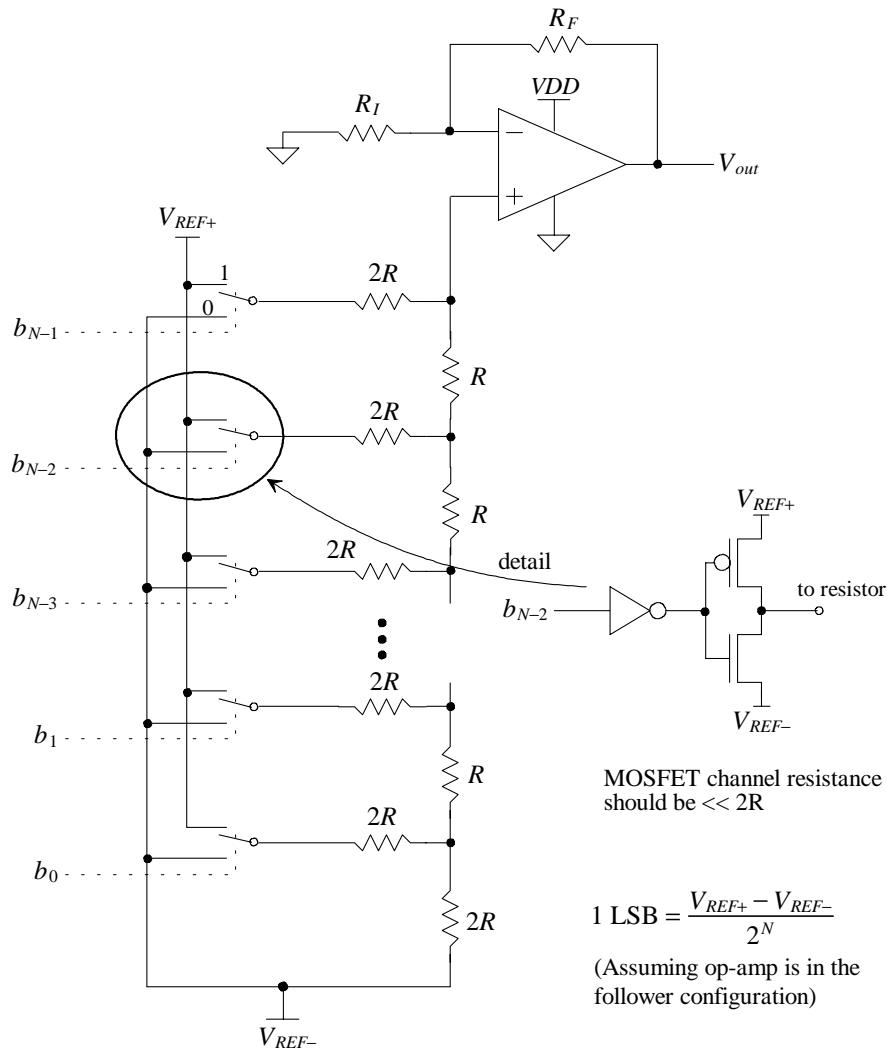


Figure 34.2 Traditional voltage-mode R-2R DAC.

DAC's inputs are changing at 1 MHz is 10% of VDD . At first glance we might simply consider the resulting offset as a nonlinear gain error affecting only the large-signal linearity (INL). However, it is unlikely in any practical op-amp design that the CMRR will vary linearly with changes in the input common-mode voltage and so the small-signal linearity (DNL) will be affected as well. Since, for this example, $1 \text{ LSB} = VDD/2^N$, the resolution of the DAC, because of the finite CMRR and assuming $1 \text{ LSB} > \Delta V_{os}$, is limited to 4 bits! Performing DC or audio-frequency tests on the voltage-mode DAC made with an op-amp with a CMRR of, for example, 120 dB at DC results in no practical

resolution limit (indicating that if DAC speed isn't a concern, the voltage-mode configuration may still be used for high resolutions). Note how CMRR isn't a concern with the current-mode R-2R DAC (assuming no secondary effects, such as common mode substrate noise, are present on the input of the op-amp). *For precision, high-speed data converter design we must use an inverting op-amp topology where the inputs of the op-amp remain at a fixed voltage.*

34.1.3 A Wide-Swing Current-Mode R-2R DAC

We've shown that it is desirable to have a wide output swing, as is provided by the voltage-mode *R-2R* DAC, while at the same time having a fixed input common mode voltage, as is provided by the current-mode *R-2R* DAC. Figure 34.3 shows a wide-swing current-mode *R-2R* DAC configuration that has a rail-to-rail output swing while keeping the input common-mode voltage of the op-amp fixed at the common mode voltage, V_{CM} , or $(V_{REF+} + V_{REF-})/2$.

Like traditional current-mode *R-2R* DACs, the DAC scheme shown in Fig. 34.3 operates on currents. Using superposition and assuming V_{REF-} is the reference for calculations, we can show that the current flowing in the feedback resistor, R_F , is given by

$$I_F = -\frac{V_{REF+} - V_{REF-}}{2R} + \frac{V_{REF+} - V_{REF-}}{2R} \cdot \left[1 \cdot \overline{b_{N-1}} + \frac{1}{2} \cdot \overline{b_{N-2}} + \dots + \frac{1}{2^{N-1}} \cdot \overline{b_0} \right] \quad (34.9)$$

noting the inversion used in the control logic of Fig. 34.3. The output voltage of the DAC is then given, assuming $R = R_F$, by

$$V_{out} = V_{REF-} + \frac{V_{REF+} - V_{REF-}}{2} + I_F \cdot R \quad (34.10)$$

or

$$V_{out} = V_{REF-} + (V_{REF+} - V_{REF-}) \cdot \left[1 - \left(\frac{1}{2} \cdot \overline{b_{N-1}} + \frac{1}{4} \cdot \overline{b_{N-2}} + \dots + \frac{1}{2^N} \cdot \overline{b_0} \right) \right] \quad (34.11)$$

From this equation, we see that as the digital input code is sequenced through 000... to 111... the output of the DAC changes in steps of $(V_{REF+} - V_{REF-})/2^N$ (= 1 LSB) from V_{REF+} (when the input code is 111...) to $V_{REF-} + 1$ LSB (when the input code is 000...). Setting V_{REF-} to ground and V_{REF+} to VDD allows the DAC output to swing from rail to rail.

In practice, since any rail-to-rail output op-amp has high nonlinearity close to its power-supply rails, a slightly “shrunk” output range from power rails is often desired. For example, we can set $V_{REF+} = 0.9 \cdot VDD$ and $V_{REF-} = 0.1 \cdot VDD$. The output will change between 10 and 90% of VDD centered at $VDD/2$. Another way to shrink the output range is to make the feedback resistance R_F smaller than R (as seen in Eq. [34.10]) by either trimming or programming the value of the feedback resistor R_F .

The matching between the resistors of the *R-2R* ladder is one of the most important and limiting factors that determine the linearity (e.g., DNL and INL) of the entire DAC. It is helpful, when designing any type of resistor string DAC, if we can estimate the resistor matching requirements based on a desired resolution.

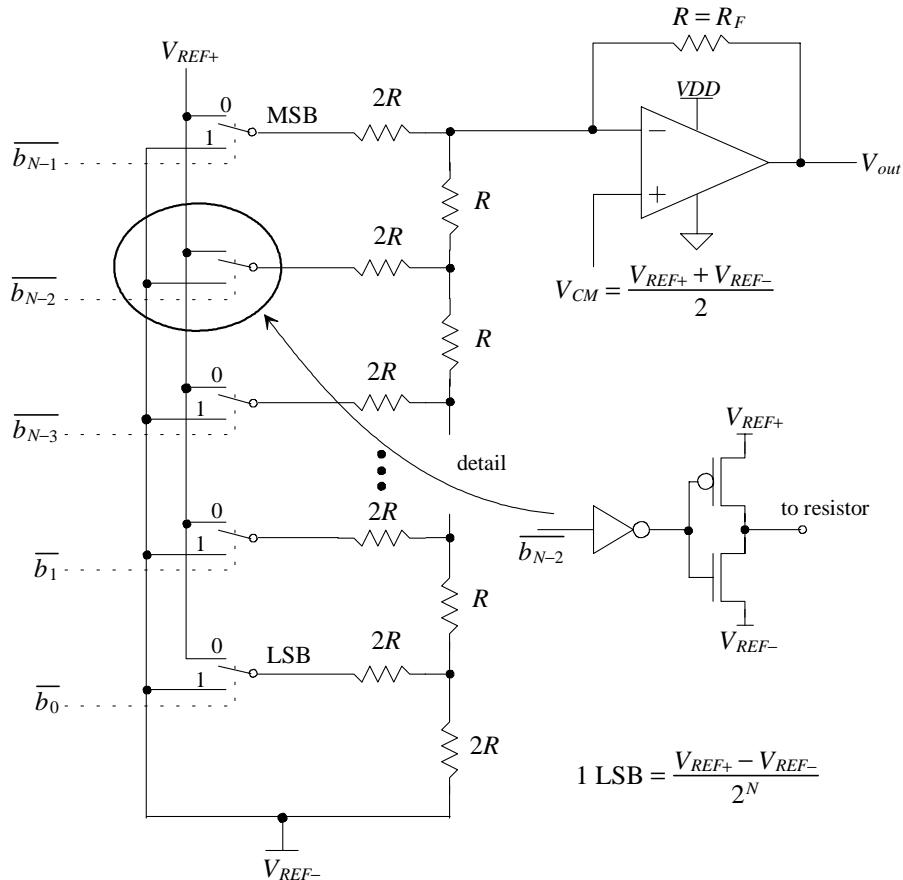


Figure 34.3 Wide-swing current-mode R-2R DAC.

DNL Analysis

It was shown back in Ch. 29 that for a binary-weighted DAC the worst case DNL condition tends to occur at midscale when the code transitions from 01...11 to 10...00. Let's assume in a worst-case scenario the $2R$ resistance of the MSB input in Fig. 34.3 has a maximum positive mismatch of ΔR , and all other resistors have a maximum negative mismatch of $-\Delta R$. In this case, the current provided by the MSB has to match the sum of currents provided by all other lower input bits plus one LSB. Again using the superposition principle we can verify that the step error of the current flowing through the feedback resistor R_F , caused by the resistor mismatch at the midscale transition, is approximately equal to

$$\Delta I = \frac{V_{REF+} - V_{REF-}}{2(R - \Delta R)} \cdot \left[1 - \frac{1}{2^{N-1}} \right] - \frac{V_{REF+} - V_{REF-}}{2(R + \Delta R)} \quad (34.12)$$

Assuming $R_F = R$, the final output step error (DNL) is approximately

$$DNL = \Delta I \cdot R \approx (V_{REF+} - V_{REF-}) \cdot \left[\frac{\Delta R}{R} - \frac{1}{2^N} \right] \quad (34.13)$$

For the DNL to be within 1 LSB (1 LSB equals to $[V_{REF+} - V_{REF-}]/2^N$) the matching required of the resistors is

$$\text{Resistor mismatch} = \left| \frac{\Delta R}{R} \right| \leq \frac{1}{2^{N-1}} \quad (34.14)$$

For a 10-bit data converter to have a DNL of less than 1 LSB requires the MSB resistor to match within 0.2% ($= \Delta R/R$) of the lower resistors (which were assumed to have the same value, i.e., the maximum mismatch from the MSB resistor) in the R-2R string. Equation (34.14) results in a pessimistic estimate for the matching required of the resistors because the variation in resistance along the string does not vary abruptly at the MSB resistor but rather, in most cases, varies linearly from LSB to MSB. As we'll see in the experimental results discussed in the next section, the matching requirements results in a practical limit of 10 bits for an R-2R-based converter with no special layout or circuit techniques (for example, averaging process gradients by using multiple resistor strings or using segmentation).

INL Analysis

Since any change of the $2R$ resistance in the MSB has the largest influence on the ladder output current among that of all the branch resistors ($2R$), the worst case INL tends to occur when the input code is 01...11. (The gain error is nulled from the INL calculation here, and therefore there is no INL error, but a gain error instead, if all the resistors have a maximum mismatch.) Assuming that the $2R$ resistance of the MSB has a maximum positive mismatch of $\Delta R/R$, the error in the current flowing through R_F from its ideal value caused by the resistance mismatch is

$$\Delta I = \frac{V_{REF+} - V_{REF-}}{2(R + \Delta R)} - \frac{V_{REF+} - V_{REF-}}{2R} \approx -\frac{V_{REF+} - V_{REF-}}{2} \cdot \frac{\Delta R}{(R + \Delta R) \cdot R} \quad (34.15)$$

The worst-case INL tends to occur, assuming $R_F = R$, when

$$INL = -\Delta I \cdot R \approx \frac{V_{REF+} - V_{REF-}}{2} \cdot \frac{\Delta R}{R + \Delta R} \quad (34.16)$$

For the INL to be within 1 LSB, this also approximately yields

$$\text{Resistor mismatch} = \left| \frac{\Delta R}{R} \right| \leq \frac{1}{2^{N-1}} \quad (34.17)$$

Again, as was mentioned in the DNL analysis, this is a pessimistic estimate if the sheet resistance varies linearly with distance. Equations (34.14) and (34.17) indicate that a resistance matching to within $1/2^N$ is required for less than $1/2$ LSB of DNL and INL for the DAC scheme in Fig. 34.3. Layout of R-2R resistors was discussed in Ch. 33.

Switches

The switches (MOSFETs) used in the R - $2R$ DAC should have an effective switching resistance (see Eqs. [33.3] and [33.4]), much less than the resistors used in the R - $2R$ ladder. The inherent switching time of the switches is extremely fast (speeds comparable to logic gate delays). Since the switches are in series with the branch resistances of the R - $2R$ ladder, the R - $2R$ relationship is broken if the switch resistance is not negligible, and this affects both the INL and the DNL. Also note that we can try to compensate for the switch-effective resistance by making the length of the $2R$ resistor slightly shorter than the length of the R resistor. However, if not careful, this may lead to problems over the process corners and temperature.

Experimental Results

The wide-swing, current-mode R - $2R$ DAC, based on the scheme in Fig. 34.3, was fabricated in a $0.21\text{ }\mu\text{m}/1.8\text{ V}$ CMOS process (single poly, up to five layers of metal) for resolutions of 8, 10, and 12 bits. The cell dimensions of the 12-bit DAC are $150\text{ }\mu\text{m}$ by $300\text{ }\mu\text{m}$. The goal of the experimental results was to verify that the topology of Fig. 34.3 would indeed perform as predicted by Eqs. (34.14) and (34.17) and to generate a low-power, small-area DAC cell for general-purpose, mixed-signal circuit designs. Unsilicided n+ poly was used for the R - $2R$ resistances as discussed earlier (see Table 33.1). The mismatch indicated in Table 33.1 for an unsilicided n+ poly resistor is 0.005 ($= \Delta R/R$). Using Eqs. (34.14) and (34.17), we would estimate that our resolution is limited to 8.6 bits if we want both INL and DNL less than 1 LSB. The results in Table 34.1, however, show that the resolution is better than estimated. This may be because of our pessimistic assumption of how the resistor values change with position as discussed in the derivation of these equations. The nominal resistor value used in these experimental DACs is 10k. To enhance the resistance matching, dummy resistors are implemented at both ends of the R - $2R$ ladder (see Fig. 33.13). The output range of the DAC is programmable by choosing the value of the feedback resistance or by the setting of the reference voltages V_{REF+} and V_{REF-} .

	8-bit	10-bit	12-bit
DNL (LSB)	0.150	0.450	2.000
INL (LSB)	0.200	1.000	3.000
Settling time	200 ns		
Power	3.88 mW (driving a 1k load)		
Area (mm^2)	0.045		
$f_{clk,max}$	4 MHz		
Output swing	$0 < V_{out} < VDD (= 1.8\text{ V})$		

Table 34.1 Summary of experimental results.

The measured INL and DNL profiles of the three DACs with resolutions of 8, 10, and 12 bits are shown in Fig. 34.4. The outputs of the DACs are configured to swing to both rails ($V_{REF+} = VDD$ and $V_{REF-} = 0$). The first several points, adjacent to the two rails, are not shown in Fig. 34.4 due to the high nonlinearity of the op-amp in those regions. Major performance results are maximum DNLs of 0.15 LSB, 0.45 LSB, and 2 LSB for 8-bit, 10-bit, and 12-bit resolutions, respectively, with no special circuit techniques (laid out as shown in Fig. 33.13) or trimming (adjustments). The corresponding maximum DAC INLs are 0.2 LSB, 1 LSB, and 3 LSB, respectively. Notice that the LSB of the 8-, 10-, and 12-bit DACs are 7.03 mV, 1.75 mV, and 439 μ V, respectively. The DNL/INL can be written in terms of a voltage as 1.05 mV/1.4 mV for the 8-bit DAC, 0.788 mV/1.75 mV for the 10-bit DAC, and 0.878 mV/1.31 mV for the 12-bit DAC. The measurements were taken while the DAC was driving a 1k load. The power dissipated by the DAC, with 1.8 V output, while driving a 1k resistor is 3.88 mW. The unloaded power dissipation of the DAC is approximately 500 μ W. The DACs were designed using op-amps with simulated unity gain frequencies of 10 MHz. The measured DAC settling time was approximately 200 ns.

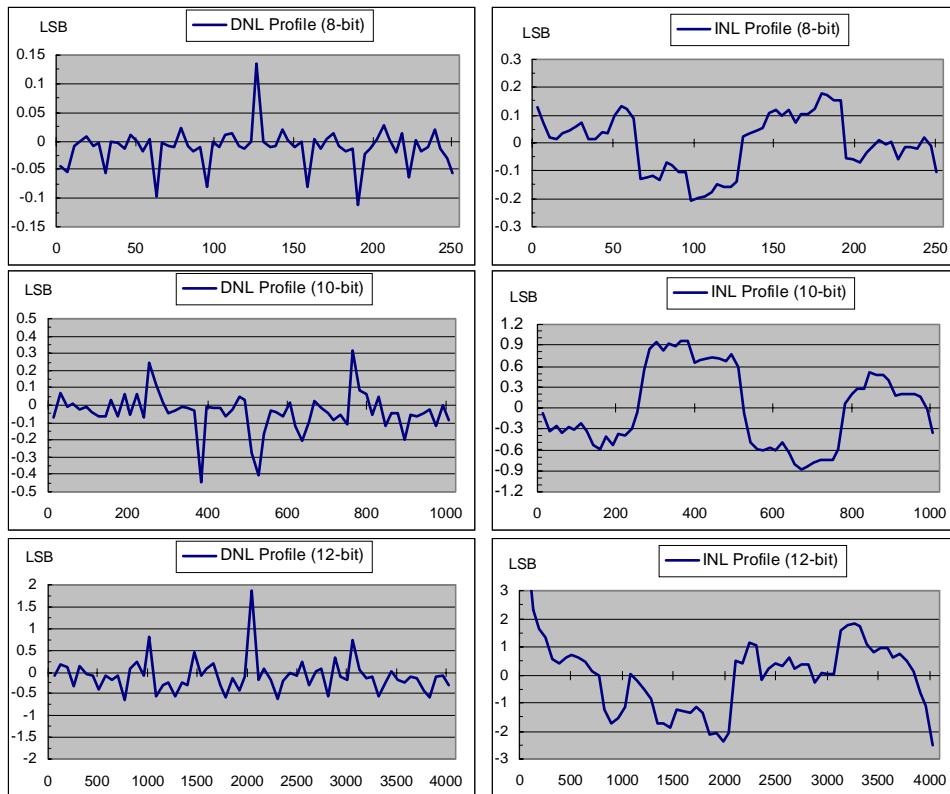


Figure 34.4 Experimental results for the wide-swing DAC of Fig. 34.3.

Improving DNL (Segmentation)

After reviewing the DNL plots in Fig. 34.4, we see that the worst-case DNL occurs when the input code transitions from 01111... to 10000... (midscale) where the current in the top $2R$ should be 1 LSB (equivalent in current) greater than the sum of all of the currents contributed by the lower resistors. As an example, consider the 12-bit R-2R ladder in Fig. 34.5 where we have used 1 μA to indicate an LSB of current contribution to the feedback path. When the input digital code is 0111 1111 1111, the feedback current is 2047 μA . When the code changes to 1000 0000 0000, the feedback current becomes (ideally) 2048 μA . If a 1/2 LSB error (0.5 μA) is the maximum error allowable, then the accuracy required of the currents when transitioning is 0.5/2048 or 0.0244%. If we use fewer bits, say eight, then the accuracy required when transitioning from 255 μA to 256 μA is 0.5/256 or 0.2%.

Let's consider segmenting the upper four bits in Fig. 34.5 so that the four bits control 16 segments each contributing 256- μA to the feedback current. This segmentation

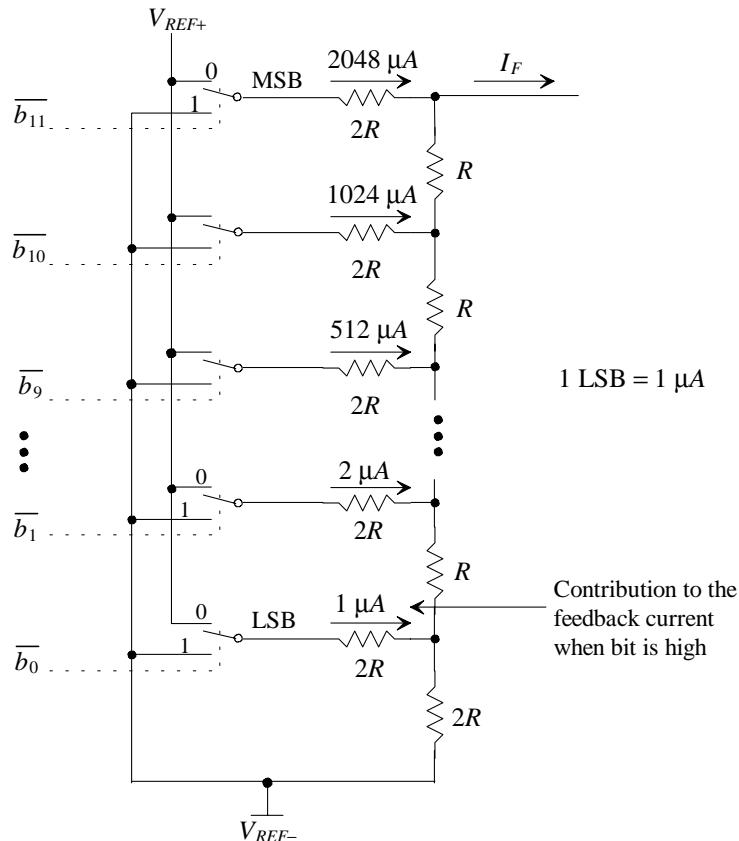


Figure 34.5 Showing how currents sum into the feedback current.

makes attaining good DNL with less accurate components possible [5]. A segmented wide-swing DAC is shown in Fig. 34.6. In this figure we've taken the upper four bits and segmented their current contributions to the feedback resistor. If we use the numbers from Fig. 34.5, then when the code 0000 1111 1111 (255 μ A) transitions to 0001 0000 0000 (256 μ A) the 1 output of the decoder goes high and the bottom resistor connected to the output of the decoder contributes 256 μ A to the feedback path. When the code changes from 0001 1111 1111 (511 μ A) to 0010 0000 0000 (512 μ A), both lower outputs (1 and 2) of the thermometer decoder are high. Since the 1 decoder output continues to contribute to the output current, the step height is set by the difference between the 2 decoder output and the contributions from the lower eight bits. This makes the accuracy

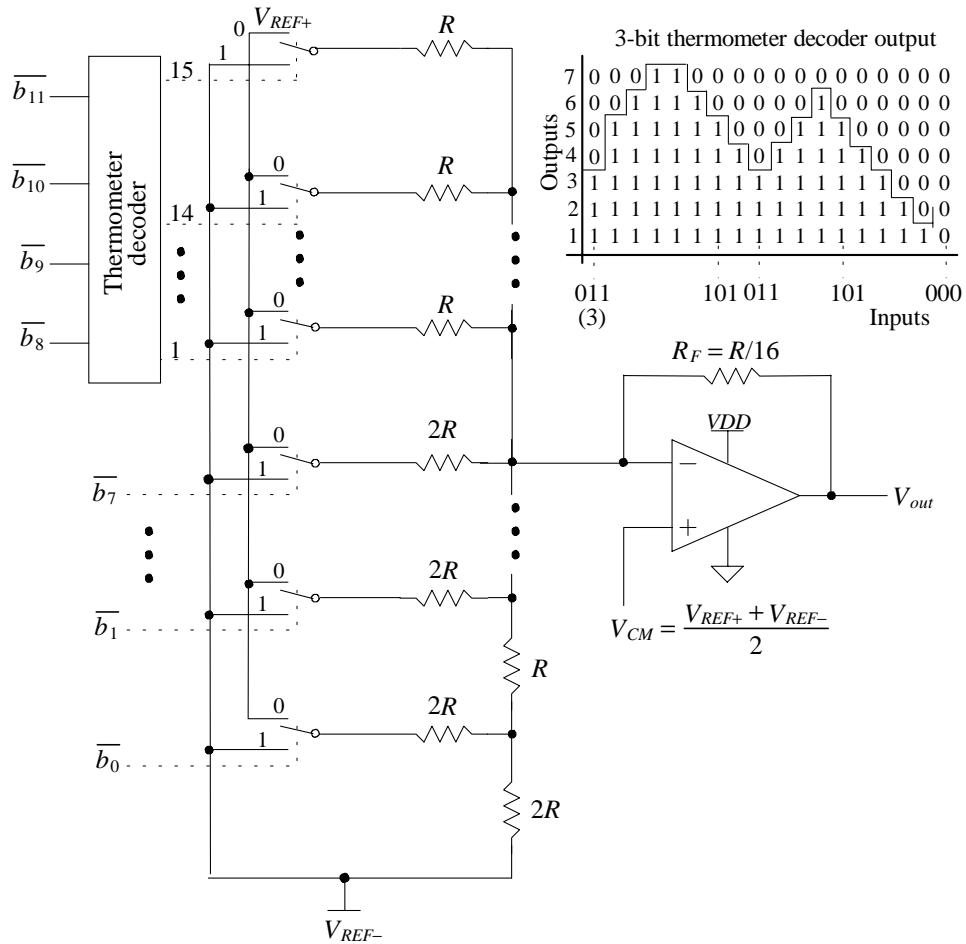


Figure 34.6 Segmentation in a wide-swing R-2R DAC.

requirements for 1/2 LSB DNL in a 12-bit converter set by 8-bit matching. Note that while segmentation reduces DNL error, it does nothing for INL. Segmentation can also be used to reduce the glitch area associated with the changing DAC output.

Trimming DAC Offset

Figure 34.7 shows how the op-amp's offset voltage shifts the DAC's output. It may be desirable in some situations to trim or remove this offset. The offset may be the result of an inherent systematic offset in the op-amp or the result of random variations in the characteristics of the MOSFETs used in the op-amp. An offset may also result because of the voltage dependence of the resistors used to generate the common-mode voltage, V_{CM} .

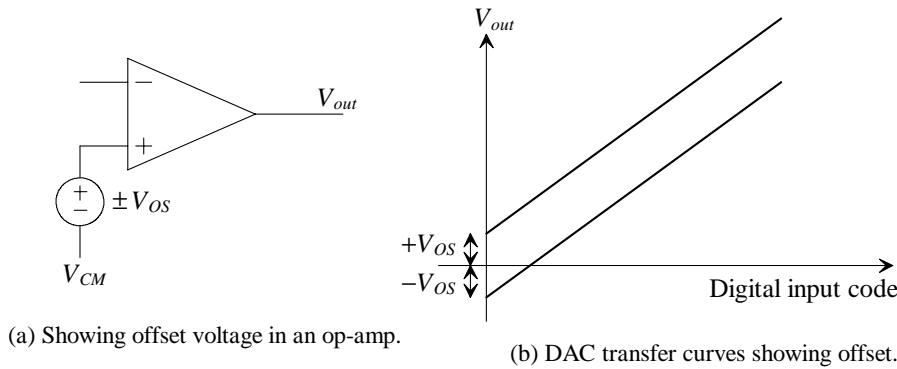


Figure 34.7 Showing how an op-amp offset affects the DACs transfer curves.

Figure 34.8 shows one possible method to generate a common-mode voltage that is adjustable with a digital code. Here again we are assuming that V_{CM} is ideally 0.75 V. We should recognize the R-2R ladder from Fig. 34.2. The output voltage of this ladder, as seen in Eq. (34.6), is an analog voltage related to the digital input word (assuming the voltage divider made up of R_{big} and the two R resistors connected to the output in Fig. 34.8 doesn't load the circuit). Figure 34.9 shows the output of this circuit for all possible digital input words when R is 10k and R_{big} is 100k. The inset in Fig. 34.9 shows that the adjustability of the output is approximately 1 mV. To decrease this value, we can either increase R_{big} (resulting in a decrease in the output swing) or increase the number of bits in the R-2R DAC. The value, R , of the resistors on the output can be decreased, but this can result in an increase in power dissipation.

Note that the accuracy required of the 5-bit DAC can be very loose. n-Well resistors can be used to implement the offset trimming circuit to reduce area and power. The main concerns are considering the possibility of substrate noise injection and making sure that the same resistive material is used for the entire circuit. We wouldn't want the temperature behavior of an n-well resistor used in a circuit with a poly resistor because the temperature dependencies are different (the offset trimming would only be effective at the temperature it was performed). Finally note that in a practical circuit it is a good idea to

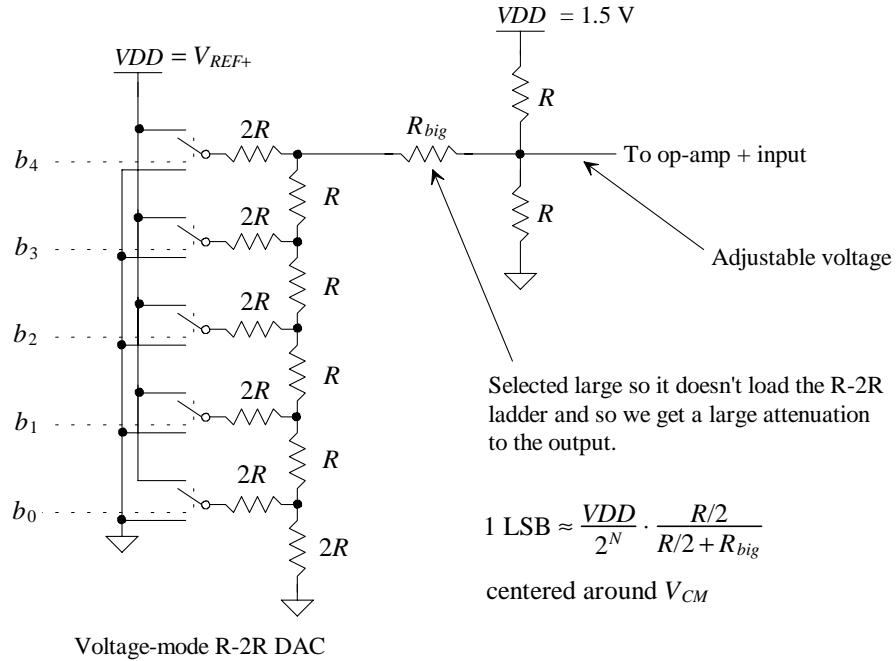


Figure 34.8 Trimming circuit for DAC offset.

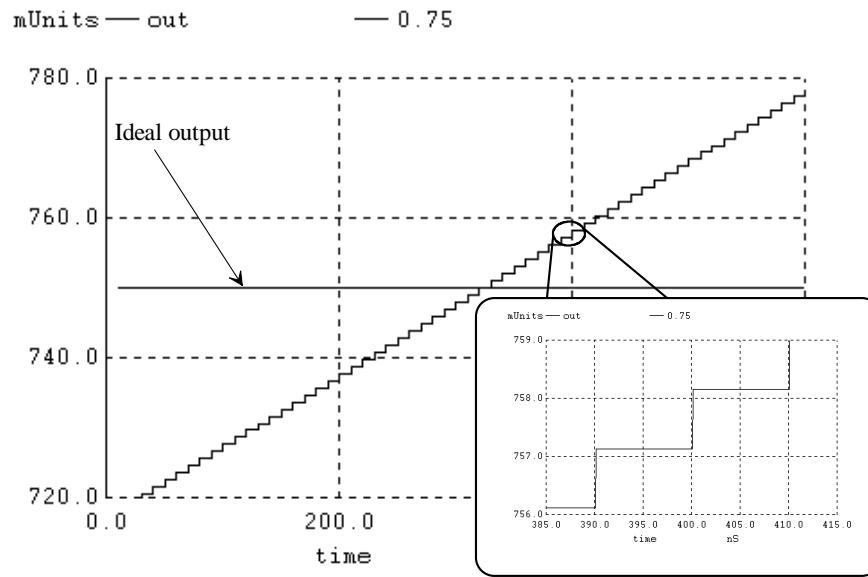
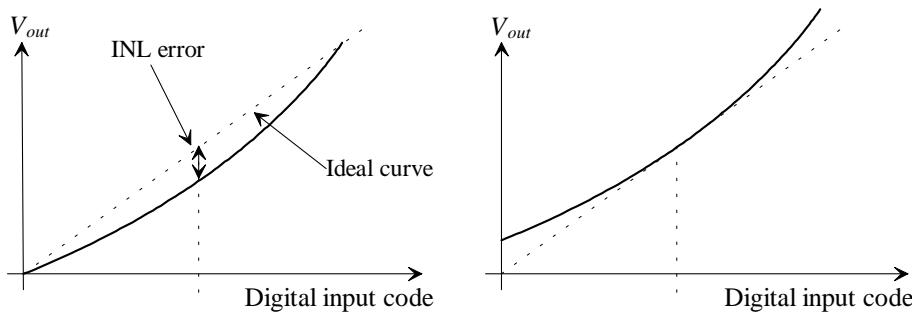


Figure 34.9 Output of the circuit in Fig. 34.8 for all possible digital codes.

add capacitors from the output of the circuit to both VDD and ground to ensure that the + op-amp input is connected to a good AC ground.

Trimming or calibrating out the offset can be performed at a time prior to packaging the chip, or it can be performed with some autocalibration sequence after the chip has been fabricated where the output of the DAC is compared to a known voltage reference. The concern, as with any calibration, is to adjust only one known error at a time (known as orthogonal tuning in filter design). For example, the DAC may not have any offset but may have an INL error for a given input code, Fig. 34.10a. If we were only to look at this one input code, say 10000... (V_{CM} in binary offset), we wouldn't know if the error is an INL error or an offset error. After the offset is calibrated out, Fig. 34.10b, we would then perform an INL calibration to pull the end-points of the transfer curve back to the ideal straight line transfer curve.



(a) DAC transfer curves before calibration. (b) DAC transfer curves after offset calibration

Figure 34.10 Showing how INL can be seen as an offset error.

Trimming DAC Gain

We assumed in Fig. 34.10 that the gain of the DAC was one, in other words, there wasn't any gain error in the DAC's transfer function. If there is a gain error, the offset calibration can lead to poorer INL. Consider Fig. 34.11a showing gain and INL errors without any offset. Performing an offset calibration, Fig. 34.11b, can result in significant INL error. We can avoid this situation by calibrating out the gain error by trimming the op-amp's feedback resistor prior to offset calibration. A reference voltage close to the ends of the transfer curve is used while adjusting the gain of the op-amp used in the DAC. If V_{REF+} is less than VDD (to avoid op-amp saturation as its output approaches the supply rails), then it can be compared directly to the output of the DAC (keeping in mind the maximum output of the DAC may be $V_{REF+} - 1$ LSB). Having gone through all of this discussion, it still would be nicer if we could simply perform two calibrations, offset calibration and INL calibration, effectively using the INL calibration to remove the gain error. The drawback of this two calibration method is the requirement that an INL calibration circuit be capable of removing very large INL errors.

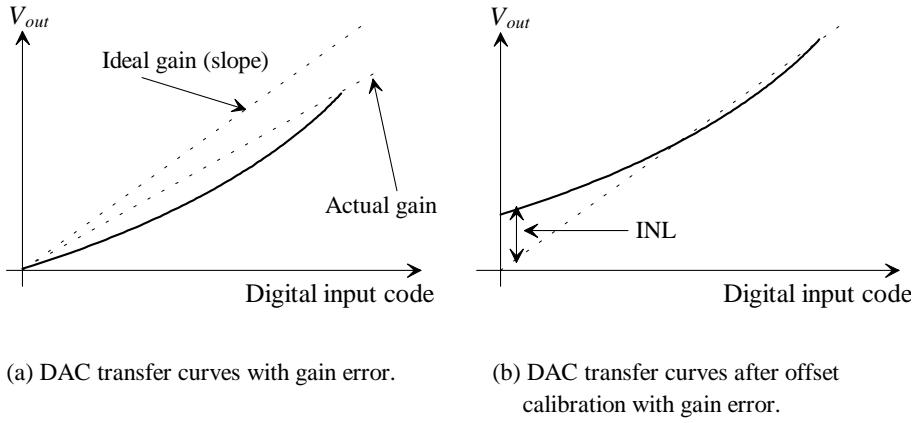


Figure 34.11 Showing gain error and how it can cause problems in an offset calibration.

Improving INL by Calibration

We can calibrate out errors in our wide-swing DAC in two basic ways as seen in Fig. 34.12. The method shown in part (a) adds or subtracts a current from the feedback path to adjust the DAC output to the correct value. In part (b) the noninverting input of the op-amp is varied to force the DAC output to the correct value. The offset calibration described earlier uses the method shown in part (b). Note that the resistance looking from the inverting op-amp terminal back through the ladder to AC ground is simply R , so using the method in part (b) results in a noninverting op-amp configuration with a gain of two. (A variation of 1 mV on the + op-amp terminal causes an output variation of 2 mV.) Because we already have a circuit, Fig. 34.8, to make adjustments to the DAC output and the topology of part (b) doesn't provide any DC load to the calibrating voltage source and provides the least interaction with the main $R-2R$ ladder, we will use this topology to illustrate how we can calibrate out INL errors.

Consider the calibration circuit shown in Fig. 34.13. In this figure the five most significant bits of a 12-bit DAC, that is, b_{11} , b_{10} , b_9 , b_8 , and b_7 are applied to the 12-bit DAC and to the address input of a 32-to-1 MUX with 5-bit input and output words. The MUX drives the $R-2R$ circuit of Fig. 34.8. The 5-bit register feeding each MUX input is used to store the calibration values. Again the calibration can be performed after the DAC is manufactured or during its use by employing a self-calibration sequence. In this scheme the DC offset calibration shown in Fig. 34.10 is simply one case of the 32 calibrations performed. (The top five bits of the input word are 10000 for the DC calibration.) This technique can be used to precisely set the linearity of the DAC, perhaps up to 16-bits. Note that segmentation must still be employed to keep the DNL small. Also note that adding a large capacitance, C , to the noninverting input of the op-amp can result in a long time constant ($\approx \frac{R}{2} \cdot C$), slowing the settling of the circuit and affecting the high-frequency SNDR.

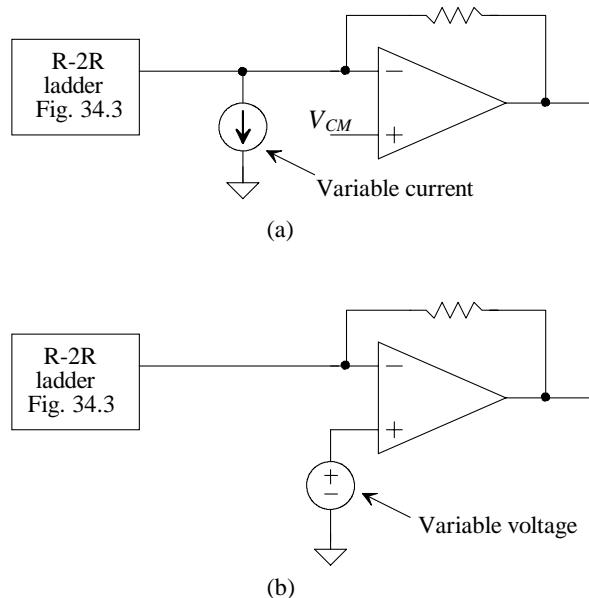


Figure 34.12 Trimming the output of the DAC using (a) current and (b) voltage.

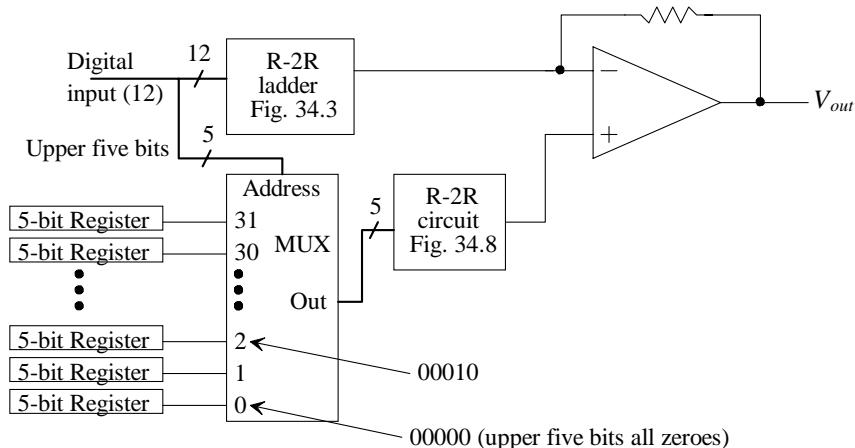


Figure 34.13 Calibration scheme for 12-bit DAC.

34.1.4 Topologies Without an Op-Amp

We discuss the requirements of op-amps used in data converters in detail in the next section. In this section we present an overview of topologies using voltage and current mode DACs based on both R - $2R$ and W - $2W$ topologies (see Fig. 33.17). The benefit of using a DAC with an op-amp is that an arbitrary load impedance (within reason) can be connected to the DAC's output. Without the op-amp, the load impedance must either be known, capacitive, or very large, since it will load the DAC's output and affect INL, DNL, and, ultimately, the SNR. The benefits of not using an op-amp are faster-speed (with light loads) and guaranteed stability.

The Voltage-Mode DAC

The simplest voltage-mode DAC is the R - $2R$ string shown in Fig. 34.14. We should recognize this circuit from both Figs. 34.2 and 34.8. For the moment we assume the load is purely capacitive so that errors resulting from sourcing a DC current are not present in the DAC. Here, through several examples, we discuss settling time, resistor voltage coefficient, matching, and the effects of a DC load.

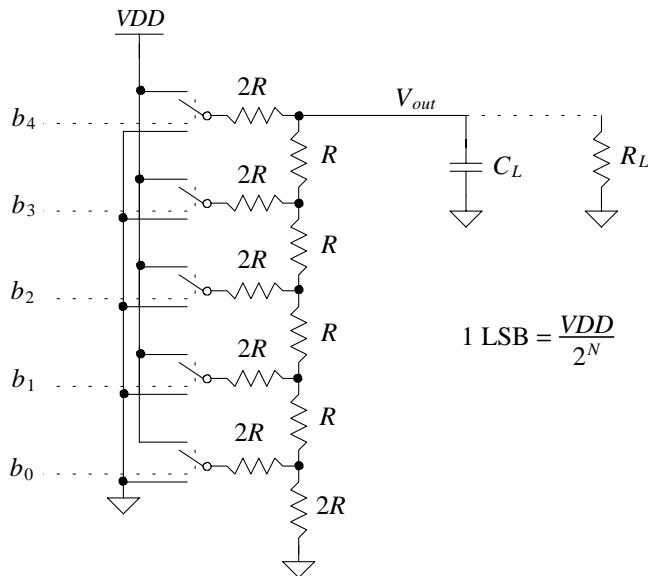


Figure 34.14 Voltage-mode (5-bit) DAC without an op-amp.

Example 34.1

Suppose a 10-bit, voltage-mode DAC with the topology given in Fig. 34.14 is implemented where $R = 10\text{k}$ and $C_L = 10 \text{ pF}$. Estimate the maximum clocking

frequency we can use to clock the register supplying the input words to the DAC. Verify your answer using SPICE.

For complete settling we require that the DAC be 10-bit accurate to within 0.5 LSBs over its full-scale range

$$\text{Accuracy} = \frac{0.5 \text{ LSB}}{\text{Full scale range } (VDD)} = \frac{VDD/2^{N+1}}{VDD} = \frac{1}{2^{11}} = 0.04883\%$$

The time constant associated with the DAC and capacitive load is

$$RC_L = 10k \cdot 10p = 100 \text{ ns}$$

We can use this time constant to relate the final ideal output voltage, $V_{outfinal}$, to the actual output voltage, V_{out} , using

$$V_{out} = V_{outfinal}(1 - e^{-t/RC_L})$$

or, relating this to the required accuracy,

$$\frac{1}{2^{N+1}} = 1 - \frac{V_{out}}{V_{outfinal}} = e^{-t_{settling}/RC_L}$$

The required settling time is then

$$t_{settling} = RC_L \cdot \ln 2^{N+1} \quad (34.18)$$

Using the numbers from this example results in $t_{settling} = 762$ ns. The SPICE simulation results are shown in Fig. 34.15. The maximum clock frequency is then estimated as

$$f_{clk,max} = \frac{1}{t_{settling}} = \frac{1}{RC_L \cdot \ln 2^{N+1}} \quad (34.19)$$

For this example, $f_{clk,max} = 1.3$ MHz. Note that the fundamental way to decrease the settling time is to decrease the resistance in the R - $2R$ ladder (assuming we have no control over the load capacitance). The practical problem then becomes implementing the switches (MOSFETs) with a resistance small compared to R . ■

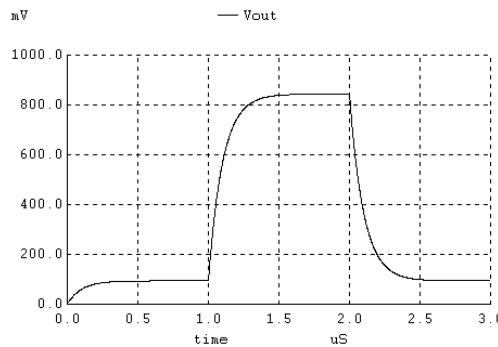


Figure 34.15 Example output for the 10-bit DAC in Ex. 34.1 showing settling time limitations.

Example 34.2

Suppose that the $2R$ MSB resistor in the DAC described in Ex. 34.1 experiences a 0.5% mismatch. Estimate the resulting DACs INL and DNL. Use SPICE to verify your answer.

The 0.5% mismatch ($\Delta R/R$ or 1σ [standard deviation]) is the mismatch specified for the unsilicided n+ polysilicon resistors specified in Table 33.1. Again it is desirable to use poly resistors because they sit above the substrate on the field oxide and are more immune to substrate noise. Note that the voltage coefficient can (will) also cause nonlinearities. However, instead of the worst-case situation of an abrupt mismatch between the lower resistors and the MSB $2R$ resistor, as used in this example, a first-order voltage coefficient error will cause a linear variation of the resistor values from the LSB resistor up to the MSB resistor (and so the effects of the voltage coefficient, for reasonably small values, are generally not significant compared to the random mismatch effects).

Rewriting Eqs. (34.14) and (34.15) to estimate the maximum number of bits possible with 1 LSB INL or DNL results in

$$N = 1 - 3.3 \cdot \log\left(\frac{\Delta R}{R}\right) \quad (34.20)$$

Using this equation with $\Delta R/R = 0.005$ results, again, in $N = 8.6$ bits. For a 10-bit DAC, we would estimate both the INL and DNL as 2.4 bits.

To verify these results using SPICE, let's input a code of 01 1111 1111 (ideally 748.5 mV) and then step the input code to 10 0000 0000 (ideally, 750 mV). With the MSB $2R$ resistor changed to 20.1k (a 0.5% mismatch from its ideal 20k value) the simulation results are shown in Fig. 34.16. With this mismatch the output of the DAC is 750.4 mV when the input is 01 1111 1111. The INL with this input code is 1.25 LSBs (roughly 1.9 mV). The INL when the input digital code is 10 0000 0000 is -1.25 LSBs. The DNL at this worst-case point is -2.5 LSBs. Note

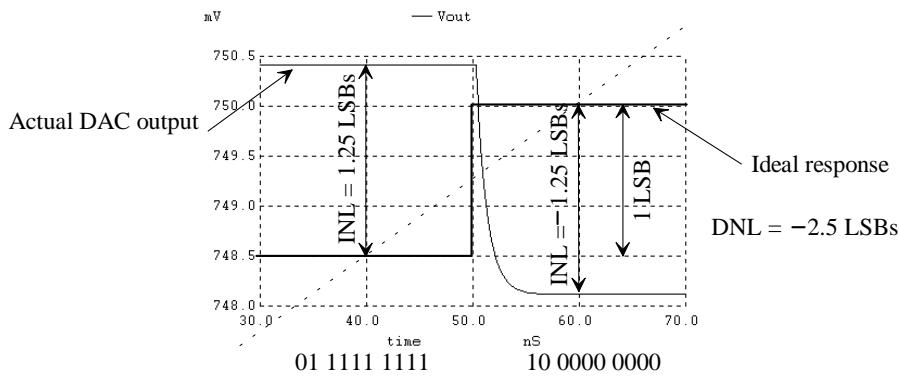


Figure 34.16 Output if MSB resistor in Fig. 34.14 experiences a 0.5% mismatch.

that the DAC is nonmonotonic ($DNL < -1$ LSB). An increase in the digital input code results in a decrease in the output voltage. Nonmonotonic DACs can result in circuits that don't function properly (an example being a successive approximation ADC). A DNL of -1 LSB would indicate the output voltage of the DAC doesn't change when the input code changes.

To improve the DNL, the upper bits of the DAC must be segmented as seen in Fig. 34.6. Improving the INL relies on calibrating out the mismatch errors (see Figs. 34.12 and 34.13). Also note, again, that mismatch can be improved by layout techniques (e.g., common-centroid) and by averaging the outputs of multiple resistor strings. ■

We can characterize the effects of a DC load resistance, R_L , as seen in Fig. 34.14, by noticing that R_L forms a divider with the R - $2R$ ladder. The LSB with a load can be written as

$$1 \text{ LSB} = \frac{VDD}{2^N} \cdot \frac{R_L}{R+R_L} \quad (34.21)$$

Notice that if $R_L \rightarrow \infty$, this equation reduces to the LSB value given in Fig. 34.14. The time constant associated with driving an output capacitance can now be written as

$$\tau = R || R_L \cdot C_L \quad (34.22)$$

Two Important Notes Concerning Glitches

Note that we have assumed that the RC delay through the resistors used in the R- $2R$ ladder is negligible. This may not be the case in many practical situations (especially if diffused or implanted resistors are used), resulting in a DAC output glitch. Also, we have been simulating with perfectly aligned digital signals, that is, signals that change at the exact same moment. When the digital signals are slightly misaligned, a significant glitch can occur in the DAC's output. *This means that the inputs to the DAC should be provided by the same digital hold register.* Using segmentation with the required thermometer decoder can result in the digital signals driving the R - $2R$ ladder seeing differing delays. Care must be exercised when designing the DAC input clocking circuit (e.g., add small dummy delays).

Example 34.3

Repeat Ex. 34.2 if a 200 ps skew is experienced by the lower nine bits in the digital inputs with relation to the MSB.

The simulation results are shown in Fig. 34.17. When comparing this result to Fig. 34.16, the magnitude of the glitch in relation to the much smaller final step in the output voltage should be obvious. The small 200 ps skew in the digital inputs causes a code of 00 0000 0000 to be applied to the DAC for 200 ps. For this very short period of time the output begins to discharge from 750 mV down to ground. Note in this figure and in Fig. 34.16 the load capacitance was reduced to 0.1 pF to decrease the settling time. ■

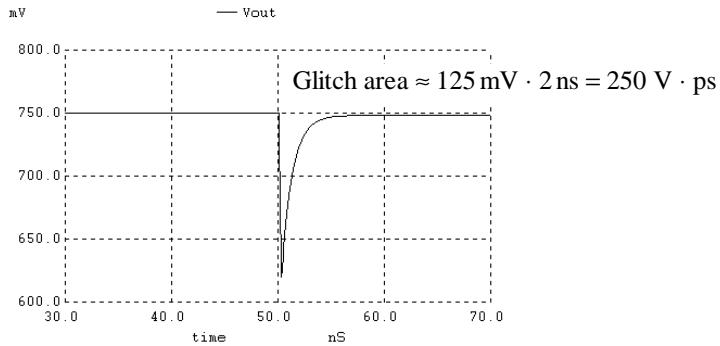


Figure 34.17 Showing glitch if the lower 9-bits are skewed by 200 ps in Ex. 34.2.

The Current-Mode (Current Steering) DAC

Figure 34.18 shows the two basic cells used in the implementation of a current-mode DAC. In this section we focus on the use of the current source-based cell. The advantage of the current-source cell is the fact that the value of the current can be adjusted, via the bias voltage, to compensate for process variations, while the value of the resistor, in the resistor-based cell is fixed. The advantages of the resistor-based cell are wider output swing (the MOSFET current source must remain in the saturation region), better voltage coefficient (no channel length modulation or other finite MOSFET output resistance effects), and better substrate noise immunity (assuming the resistors are integrated on the top of the field oxide and not down in the substrate with the MOSFETs). Notice, in this figure, that we've implemented the cells with two complementary outputs. Having complementary outputs is useful, for example, in a DAC used with a video monitor (driving two complementary 75Ω loads). The load resistors are labeled the left load resistor, R_{LL} , and the right load resistor, R_{RL} .

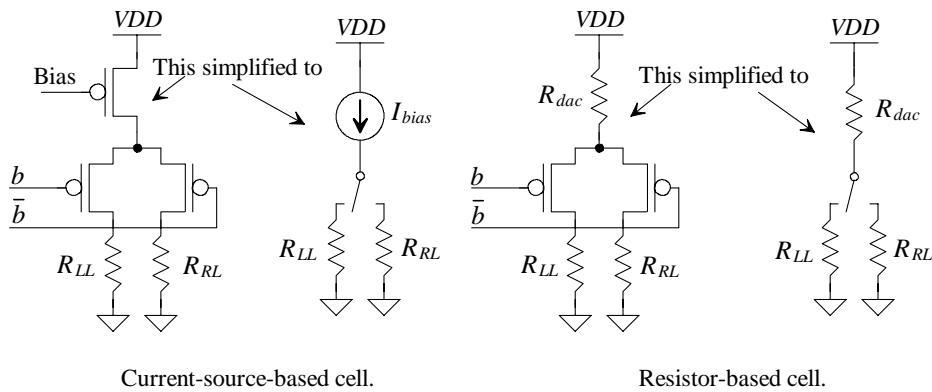


Figure 34.18 Basic cell used in a current-mode DAC.

Figure 34.19 shows the block diagram implementation of a current-mode DAC. The output voltages depend on both I_{REF} and the load resistors. The current sources are implemented using the PMOS cell in Fig. 34.18 and a PMOS W-2W mirror (see Fig. 33.17 in the last chapter) to improve layout area. The combination of binary-weighted current sources must be used together with the required segmentation of the upper bits to reduce DNL. Although not drawn so in Fig. 34.18, the current sources can be cascaded to increase their output resistance (decrease their voltage coefficient). Again the switches connected to the loads should be controlled by signals from the same register to avoid significant glitches in the outputs.

While the matching requirements of current-mode (current steering) DACs were discussed in Ch. 29, we should comment on ways to improve matching before leaving this section. The layout of the W-2W ladder should follow the basic techniques discussed in Ch. 20, i.e., devices oriented the same way, use of dummy poly and diffusions, attempt to keep the source-drain voltages constant, and use of long L devices. The layout of the W-2W mirror should look similar to the layout of the R-2R string in Fig. 33.13, but it can also employ two or more W-2W segments to average variations. The upper segmented bits

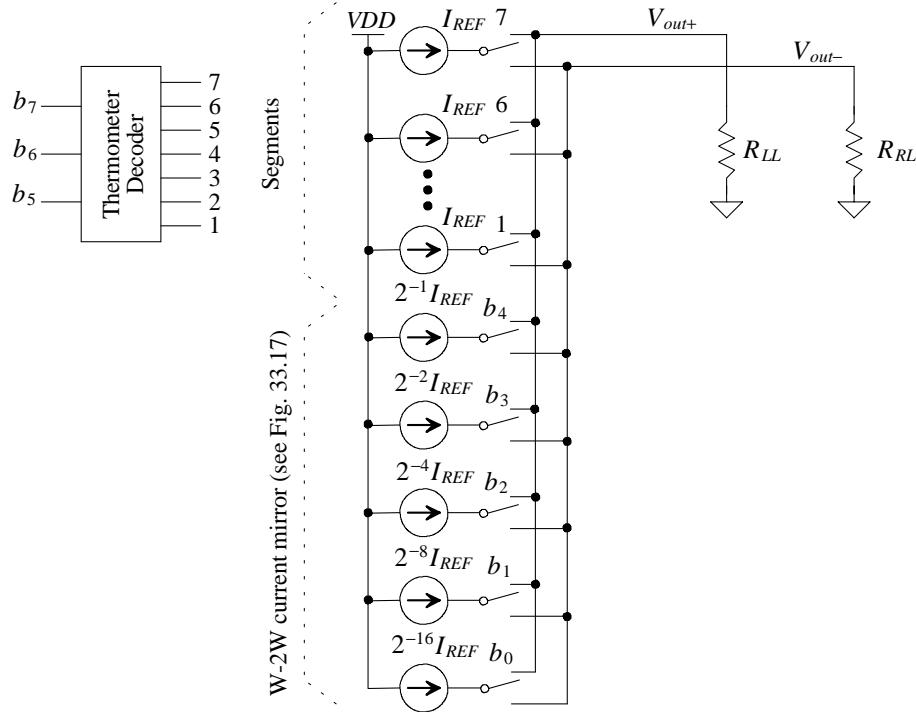


Figure 34.19 Implementation of a current-mode DAC.

can be laid out adjacent to the W - $2W$ and can also benefit from averaging the outputs of several DAC layouts. In some cases the number of bits used in the W - $2W$ ladder equals the number of bits used for the upper segments. For example, a 10-bit DAC would use a 5-bit W - $2W$ ladder (whose MSB is $I_{REF}/2$) and 31 segments with values of I_{REF} (a total of 36 outputs as seen in Fig. 34.20a). To improve INL, the layouts are connected (see the example common-centroid layout in Fig. 34.20b) together to, hopefully, average the random mismatch effects and increase the linearity of the DAC. Of course, the current output in the circuit of 34.20b is four times the current in 34.20a for the same reference biasing levels.

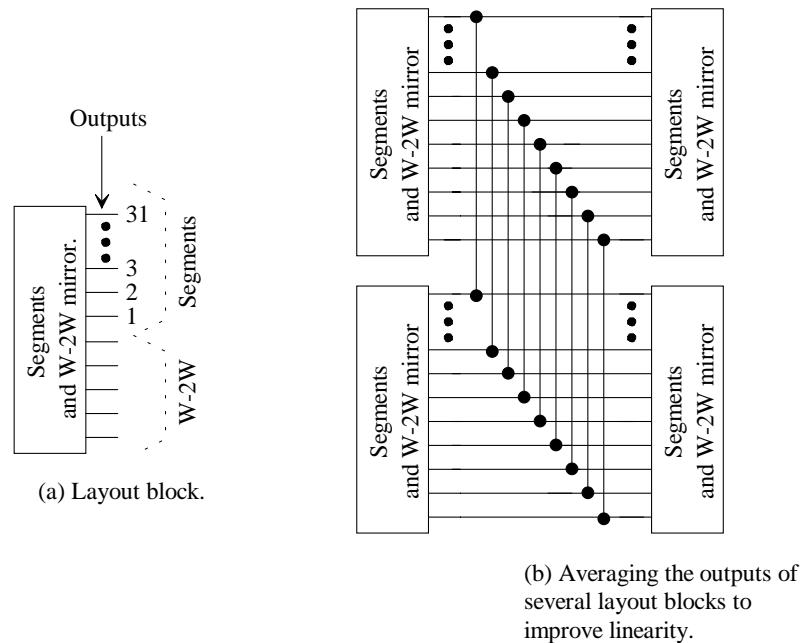


Figure 34.20 Layout of a current-mode DAC.

34.2 Op-Amps in Data Converters

The open-loop magnitude and phase responses of a typical op-amp are shown in Fig. 34.21. In this section we discuss the gain and bandwidth requirements of op-amps used in either a DAC or an ADC. We assume that the op-amp is designed to have a phase margin of 90 degrees under full load conditions and over process variations. (We should point out that this assumption is easily met using an OTA that is compensated by a load capacitance as discussed in Ch. 25.) It's important to understand why having a 90-degree phase margin is important, namely, to avoid a second-order step response with the associated ringing. If the phase margin is 90 degrees, we get an RC-like settling response shape as seen in Fig. 34.15. Figures 33.73 and 33.74 show the AC responses and step response of a basic

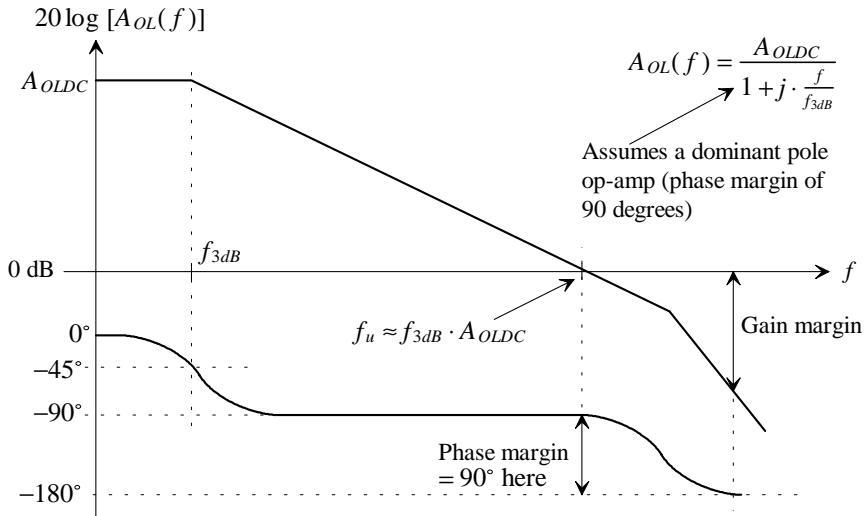


Figure 34.21 Magnitude and phase responses of an op-amp.

mixed-signal op-amp. The phase margin of this op-amp was 70 degrees. The step response shows a moderate amount of ringing. Decreasing the phase margin increases the peak amplitude of the ringing and can lengthen the settling time (the time it takes the op-amp's output to settle to within 1/2 LSB of the ideal final value). Note that the settling time, in Fig. 33.74, was measured using an inverting op-amp topology. While we concluded, in Sec. 34.1.2, that we must use an inverting op-amp (or a fully-differential topology where the input common-mode voltage remains constant), it is still useful to look at the basic speed (bandwidth) differences between noninverting and inverting topologies.

Gain Bandwidth Product of the Noninverting Op-Amp Topology

Figure 34.22 shows the basic topology of a noninverting op-amp amplifier. The voltage on the inverting op-amp input can be written as

$$v_- = V_{out} \cdot \underbrace{\frac{R_1}{R_1 + R_2}}_{\beta} \quad (34.23)$$

where β is the feedback factor for this series-shunt feedback amplifier (the ideal closed-loop gain, A_{CL} , is $1/\beta$ or $1 + R_2/R_1$). The output of the amplifier is

$$V_{out} = (V_{in} - v_-) \cdot A_{OL}(f) \quad (34.24)$$

Solving these equations for the closed-loop bandwidth of the amplifier, $f_{CL,3dB}$, gives

$$f_{CL,3dB} \approx \beta \cdot A_{OLDC} \cdot f_{3dB} = \beta \cdot f_u \quad (34.25)$$

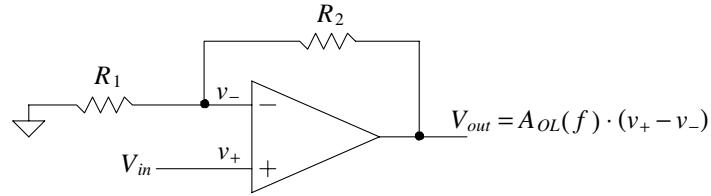


Figure 34.22 Noninverting op-amp topology.

The gain bandwidth product of the noninverting amplifier is then

$$\text{Gain} \cdot \text{bandwidth} = f_u \quad (34.26)$$

Gain Bandwidth Product of the Inverting Op-Amp Topology

Figure 34.23 shows the schematic diagram of an inverting op-amp topology using an op-amp. Summing the currents at the inverting input node gives

$$\frac{V_{in} - v_-}{R_1} = \frac{v_- - V_{out}}{R_2} \quad (34.27)$$

The output of the amplifier is related to the op-amp's input terminals using

$$V_{out} = (-v_-) \cdot A_{OL}(f) \quad (34.28)$$

Solving these two equations for the closed-loop bandwidth once again results in Eq. (34.25) with β defined as indicated in Eq. (34.23). This can be confusing because the feedback factor, β , for the inverting amplifier is not the same as for the noninverting amplifier. The inverting op-amp is an example of a shunt-shunt amplifier (current input and voltage output). The feedback factor for this amplifier is $-1/R_2 (= \beta)$ where

$$\frac{V_{out}}{i_{in}} = -R_2 \quad (34.29)$$

If we assume the input current source (Norton equivalent) has a source resistance of R_1 so that $V_{in} = i_{in} \cdot R_1$, we can write

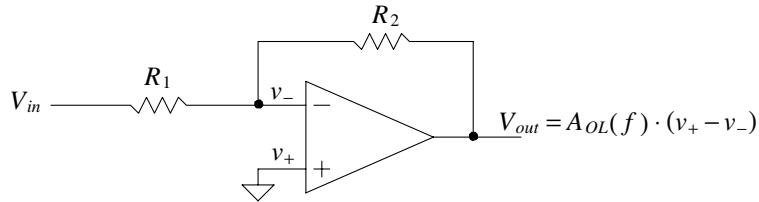
$$|A_{CL}| = \left| \frac{V_{out}}{V_{in}} \right| = \frac{R_2}{R_1} \quad (34.30)$$

Keeping in mind that closed-loop bandwidth of the inverting amplifier is still, from Eq. (34.25),

$$f_{CL,3dB} \approx \frac{R_1}{R_1 + R_2} \cdot f_u \quad (34.31)$$

we can write

$$\text{Gain} \cdot \text{bandwidth} = \frac{R_2}{R_1 + R_2} \cdot f_u \quad (34.32)$$

**Figure 34.23** Inverting op-amp topology.**Example 34.4**

Compare the bandwidth of a +1 gain amplifier implemented using a noninverting op-amp topology (Fig. 34.22) to the bandwidth of a -1 gain amplifier using the inverting op-amp topology (Fig. 34.23).

Using Eq. (34.26), the bandwidth of the +1 gain amplifier is f_u . This amplifier is commonly known as a unity voltage follower and has $R_1 = \infty$ (an open) and $R_2 = 0$ (a short). The bandwidth of the inverting, -1, gain amplifier can be determined using Eq. (34.32) with $R_1 = R_2$ and is $0.5f_u$. *This result is important because it shows that for the fastest speed the noninverting op-amp topology offers the best choice.* Practically, however, the nonlinearities related to the finite CMRR (see Eq. [34.8]) force the use of inverting op-amp topologies. As discussed earlier, the input common-mode voltage must remain constant in any precision application. Note that a fully-differential op-amp topology is also a shunt-shunt amplifier with a gain bandwidth product given by Eq. (34.32). ■

Example 34.5

Comment on the derivations of Eqs. (33.23)-(33.25) in the last chapter.

The equations are still valid; however, it would be more correct to use the value of β given by the resistive divider in Eq. (34.23) instead of $\beta = 1$. For the test setup shown in Fig. 33.74 $\beta = 0.5$ and so the settling time would more accurately be estimated as 31.8 ns. ■

34.2.1 Op-Amp Gain

In this section we answer the question of how large the DC open-loop gain of the op-amp, A_{OLDC} , must be in a data converter with a resolution of N bits. We know that the op-amp must amplify signals to within 1/2 LSB of the ideal value. Further we know that the closed-loop gain of an amplifier can be written as

$$A_{CL} = \frac{A_{OL}(f)}{1 + \beta \cdot A_{OL}(f)} \quad (34.33)$$

The feedback factor can be written, after reviewing Fig. 34.23 or Fig. 34.24, as

$$\beta = \frac{R_1}{R_1 + R_2} \text{ or } \frac{C_F}{C_F + C_I} \left[= \frac{1/j\omega C_I}{1/j\omega C_I + 1/j\omega C_F} \right] \quad (34.34)$$

where the capacitive dependence, or second term in this equation, is used when estimating the feedback factor present in a DAI.

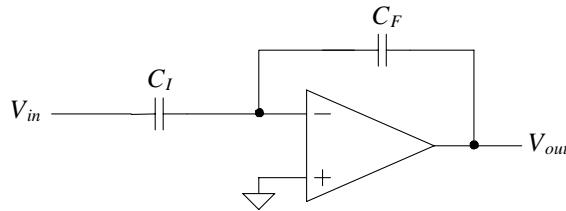


Figure 34.24 Inverting op-amp topology.

As we discussed in Ch. 29 the output of the amplifier will be equal to its ideal value minus some maximum deviation, ΔA . We can write the gain of the DAI over one clock cycle (treating the integration, $z^{-1}/[1-z^{-1}]$, as an initial DC condition on the feedback capacitor) as

$$|A_{CL}| = \frac{C_I}{C_F} \quad (34.35)$$

Then we can write

$$|A_{CL}| = \frac{C_I}{C_F} - \Delta A = \frac{A_{OLDC}}{1 + A_{OLDC} \cdot \frac{C_F}{C_F + C_I}} \quad (34.36)$$

If the maximum value of ΔA is at most 1/2 LSB of the ideal gain, or,

$$\Delta A = \frac{C_I}{C_F} \cdot \frac{1/2 \text{ LSB}}{\text{Full scale output}} = \frac{C_I}{C_F} \cdot \frac{1/2 \cdot (V_{REF+} - V_{REF-})/2^N}{(V_{REF+} - V_{REF-})} = \frac{C_I}{C_F} \cdot \frac{1}{2^{N+1}} \quad (34.37)$$

then we can estimate the minimum required DC open-loop gain as

$$|A_{OLDC}| \geq \frac{1}{\beta} \cdot 2^{N+1} \quad (34.38)$$

If $\beta = 1/2$, as in the $R-2R$ DAC of Fig. 34.3 or in a DAI with $C_I = C_F$, then this equation can be reduced to

$$|A_{OLDC}| \geq 2^{N+2} \quad (34.39)$$

A 12-bit ADC or DAC requires the use of an op-amp with a gain greater than 16k while a 16-bit converter must have $|A_{OLDC}| \geq 256k$. Clearly, this estimate can present a real design concern. Note that Eq. (34.38) is optimistic. For a general design, an error of 1/2 LSB due just to op-amp gain is not desirable (so a larger value of A_{OLDC} must be used).

34.2.2 Op-Amp Unity Gain Frequency

The speed of a data converter is mainly limited by the op-amp used. In general, the minimum op-amp gain-bandwidth product (f_u) required for a specific settling time t (where t is less than $1/f_{clk}$, within a dead band of $\pm 1/2$ LSB) can be estimated, assuming no slew-rate limitations (see also Eqs. [34.18] and [34.19]), by

$$V_{out} = V_{outfinal} \left(1 - \frac{1}{2^{N+1}} \right) = V_{outfinal} (1 - e^{-t/\tau}) \quad (34.40)$$

where, once again,

$$\tau = \frac{1}{2\pi \cdot \beta \cdot f_u} \quad (34.41)$$

The minimum required op-amp unity gain frequency is then given by

$$f_u \geq \frac{f_{clk} \cdot \ln 2^{N+1}}{2\pi \cdot \beta} \quad (34.42)$$

or, again assuming $\beta = 1/2$,

$$f_u \geq 0.22 \cdot (N+1) \cdot f_{clk} \quad (34.43)$$

If we design a 12-bit ADC that is clocked at 100 MHz, we need to use op-amps with unity gain frequencies, f_u , of 286 MHz (and a DC gain of at least 16k). Again, this estimate for the unity gain frequency is optimistic. A good design would use a larger f_u than what is specified by Eq. (34.43).

34.2.3 Op-Amp Offset

A critical characteristic of any op-amp used in a data converter is its offset voltage. We introduced the concept of reducing the offset voltage of an op-amp back in Ch. 27. Here we provide additional comments and possibilities for offset reduction.

Adding an Auxiliary Input Port

A simple method of nulling the offset voltage of an op-amp is shown in Fig. 34.25 [6]. In this figure the added MOSFETs, M1 and M2 (which operate in the triode region) are essentially used to balance the current flowing in the current mirror load. We can think of the added MOSFETs as providing an auxiliary input port for offset calibration.

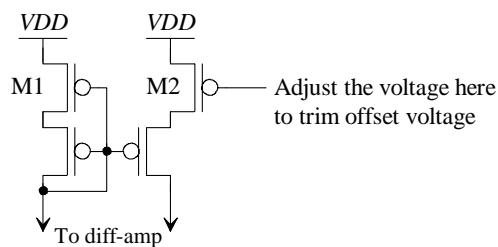


Figure 34.25 Trimming offset using an auxiliary input port.

Figure 34.26 shows how we would use the auxiliary input port to remove (lower) the offset. When zeroing out the offset, the op-amp is removed from the circuit by opening S1 (and possibly a switch [not shown] in series with the op-amp's output). This is followed by closing S2 and S3 so that a control voltage is stored on C. Note that we have assumed that the op-amp is used in an inverting configuration (that is, the noninverting input of the op-amp, +, is tied to V_{CM}). The offset removal is dynamic and will have to be performed periodically. We could also use a simple $R-2R$ DAC with a topology similar to what is seen in Fig. 34.8 to calibrate out the offset (eliminating the dynamic nature of the method). The output of the DAC would be connected to the auxiliary input port. Note that an increase in voltage on the auxiliary input port must result in a decrease in output voltage. (There must be negative feedback when connecting the output of the op-amp to the input port.)

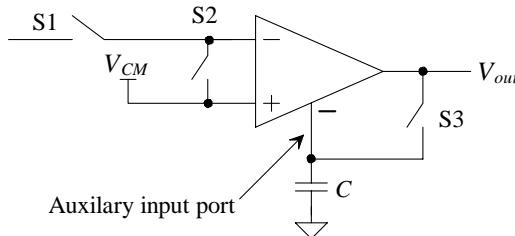


Figure 34.26 Using an auxiliary input port to lower offset.

The practical problem with the topology of Fig. 34.26 is the charge injection and capacitive feedthrough resulting from shutting off (opening) S3. This "glitch" of charge causes a change in the auxiliary port's input voltage and can place a significant limitation on the minimum possible offset voltage attainable after calibration. The amplitude of the glitch can be reduced by increasing C or by increasing the length of the MOSFET used in the op-amp (M2 in Fig. 34.25). Increasing the length results in a decrease in the MOSFET's transconductance (keeping in mind that the MOSFET is operating in the triode region) making the amplitude of the glitch less harmful. The drawback of increasing the MOSFET's length is that the range of offset voltages we can remove is reduced.

Example 34.6

Suppose perfect switches are available for the circuit of Fig. 34.26. Estimate the residual offset voltage in terms of the op-amp's gain, A_G , from the auxiliary port to the op-amp output.

If the offset voltage before reduction is V_{os} , then the offset voltage after reduction is V_{os}/A_G . For reasonable values of A_G the final inherent offset voltage is negligible. The point of this example is that the charge injection and capacitive feedthrough from the switches is the dominant source of offset error using this technique. ■

We've seen the problem of charge injection and capacitive feedthrough before. The most common technique for reducing its effect is to use a fully-differential topology. Figure 34.27 shows a modification of Figs. 34.25 and 34.26 to compensate for charge injection. The idea is that when S4 and S3 turn off (open) the variation in voltages on the gates of M1 and M2 are equal resulting in a common change in each MOSFET's resistance. Ideally then the current will remain balanced in the diff-amp. Note that while we've shown the use of triode-operating MOSFETs M1/M2 in Figs. 34.25 and 34.27 in series with the load of a diff-amp on the input of an op-amp, we could also use this concept in later stages of the op-amp.

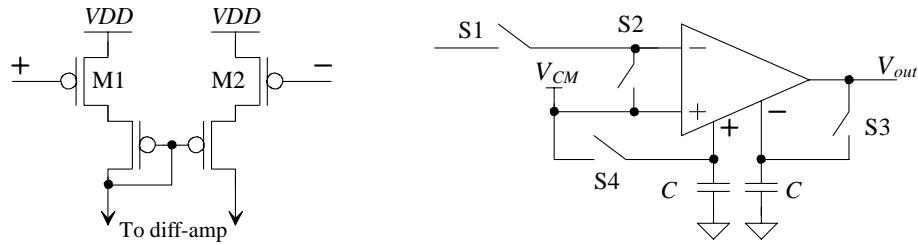


Figure 34.27 Using an auxiliary input port to lower offset (two terminals).

Figure 34.28 shows another possible topology for offset removal using an auxiliary input [7]. An additional diff-amp is added in parallel to the main input diff-amp stage of an op-amp to balance the currents and zero out the offset voltage. Again, long length MOSFETs are used in the added input so that the glitches resulting from the imperfections in the MOSFET switches (S4 and S3 in Fig. 34.27) have the least effect on the operation of the circuit.

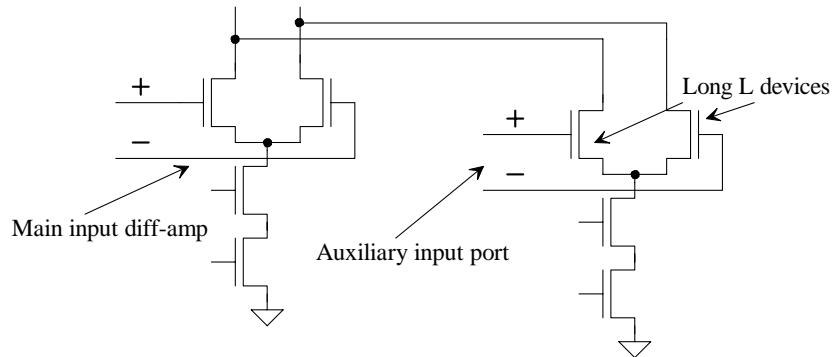


Figure 34.28 Using an auxiliary diff-amp for balancing current in an op-amp's input.

We can estimate the maximum offset voltage we can zero out using the technique of Fig. 34.28 by writing the imbalance in the main diff-amp's currents because of its offset voltage as

$$g_m \cdot V_{OS,max} = i_d \quad (34.44)$$

The auxiliary input must sum the opposite of this current in the main diff-amp's load to balance the currents in the main diff-amp (and hence eliminate the offset voltage). If we label the transconductance of the diff-amp used in the auxiliary input g_{aux} and the maximum allowable differential voltage on the auxiliary input for linear operation $V_{aux,max}$ then we can write

$$g_m \cdot V_{OS,max} = g_{aux} \cdot V_{aux,max} \quad (34.45)$$

Because we are using long length devices in the auxiliary input $g_{aux} \ll g_m$ for the same biasing current levels. If $V_{aux,max} = 200$ mV (a differential voltage of ± 200 mV will cause all of the diff-amp tail current to flow through one side of the diff-amp) and $g_m/g_{aux} = 10$, then we can zero out at most 20 mV of op-amp offset.

The offset storage technique shown in Fig. 34.27 relies on the removal of the op-amp from the circuit while autozeroing the offset. The scheme in Fig. 34.29 shows how the technique can be extended to remove the offset while leaving the main op-amp, O1 in the circuit at all times. When S2, S3, and S4 are closed, the offset of O2 is zeroed out. At this time switches S1 and S5 are open. After O2's offset is stored, S2, S3, and S4 are then opened. Next S1 and S5 close. O2 is used to precisely set the inverting input of O1 to V_{CM} through the feedback around O1 (not shown). When O2 goes back to zeroing out its own offset (S2-S4 close) the capacitor connected to the auxiliary port of O1 retains the charge, and thus voltage, needed to keep O1's offset nulled out to zero. Again this capacitor should be large to avoid problems from the imperfections of S5.

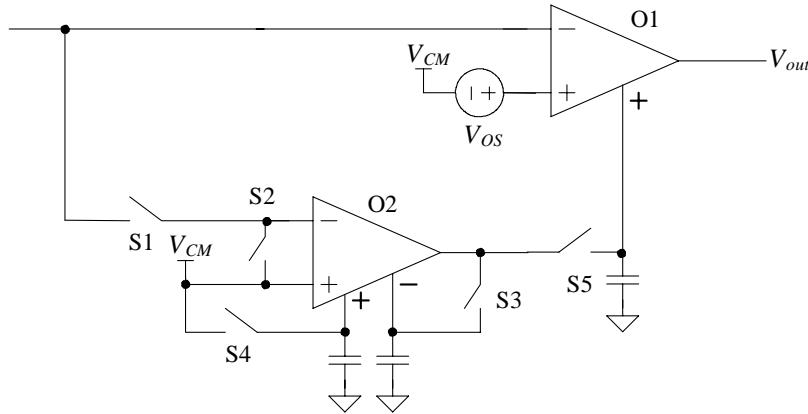


Figure 34.29 Continuous-time offset removal.

34.3 Implementing ADCs

In this section we continue to discuss implementing data converters with design concerns for S/Hs, cyclic ADCs, and pipeline ADCs.

34.3.1 Implementing the S/H

We assume the reader is familiar with the fundamental implementation of a CMOS S/H discussed in Ch. 27. Figure 34.30 shows the more general implementation of a S/H. Note that if C_I goes to 0 (an open) this topology reduces to the basic S/H given in Ch. 27 (repeated in Fig. 34.31 for convenience).

We can determine the relationship between the input of the S/H and its output by writing the charge stored on C_I and C_F when the ϕ_1 and ϕ_2 switches are closed (the ϕ_3 switches are open) as

$$Q_{I,F}^{\phi_1} = C_{I,F} \cdot (V_{in} - V_{CM} \pm V_{os}) \quad (34.46)$$

where V_{os} is the offset voltage of the op-amp and the input (and output) voltages are referenced to ground (V_{in} [V_{out}] varies from 0 to $2V_{CM}$ [= VDD here]). Note that the reason why the ϕ_2 switches turn off slightly after the ϕ_1 switches is shown in Figs. 30.31 and 30.32 and in the associated discussion (bottom plate sampling). When ϕ_3 goes high, the charge on C_I is

$$Q_I^{\phi_3} = C_I \cdot (V_{CM} - V_{CM} \pm V_{os}) \quad (34.47)$$

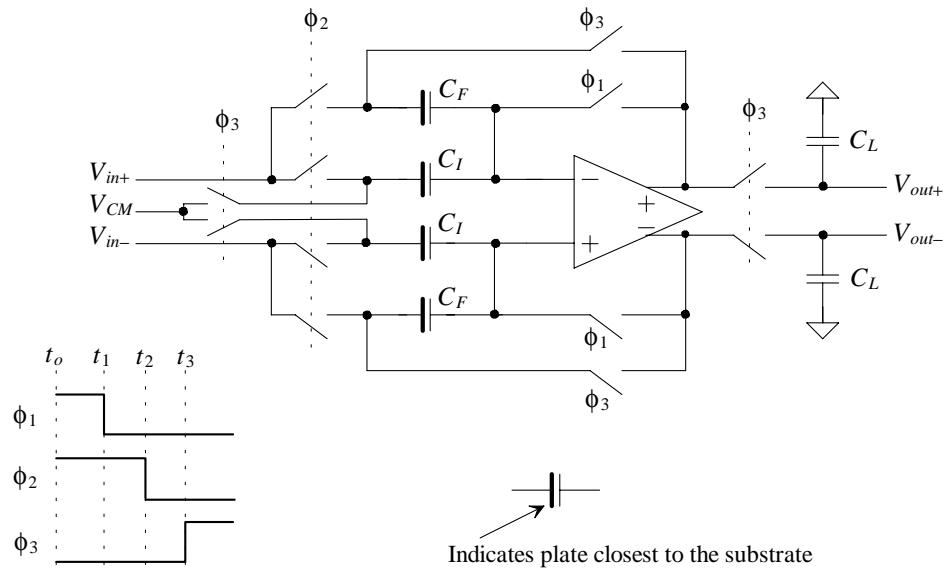


Figure 34.30 Data converter S/H building block.

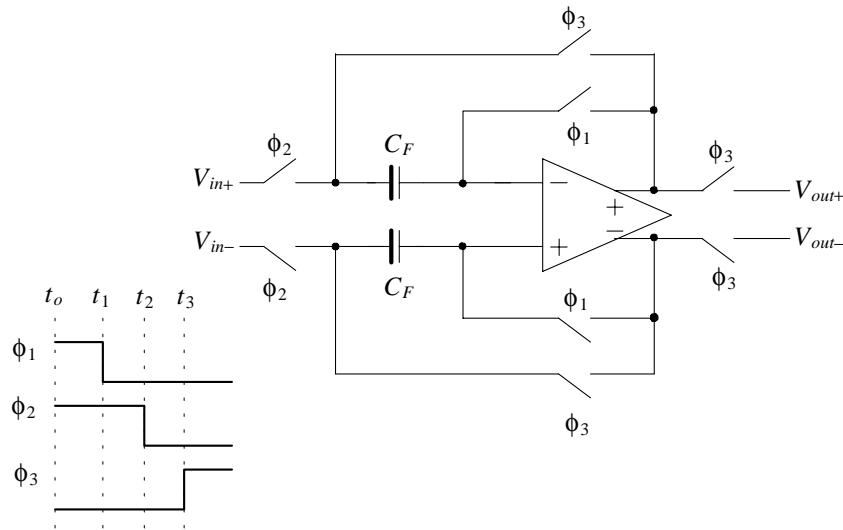


Figure 34.31 S/H differential topology from Ch. 27.

The difference between $Q_I^{\phi_1}$ and $Q_I^{\phi_3}$ is transferred to C_F when ϕ_3 goes high. The output voltage is then determined knowing charge must be conserved

$$\begin{aligned}
 & C_F \cdot (V_{out} - V_{CM} \pm V_{OS}) \\
 &= \overbrace{C_F \cdot (V_{in} - V_{CM} \pm V_{OS})}^{\frac{Q_F^{\phi_1}}{C_F}} + \overbrace{C_I \cdot (V_{in} - V_{CM} \pm V_{OS})}^{\frac{Q_I^{\phi_1}}{C_I}} - \overbrace{C_I \cdot (V_{CM} - V_{CM} \pm V_{OS})}^{\frac{Q_I^{\phi_3}}{C_I}}
 \end{aligned} \tag{34.48}$$

or when ϕ_3 goes high

$$V_{out} = \left(1 + \frac{C_I}{C_F}\right) \cdot V_{in} - \frac{C_I}{C_F} \cdot V_{CM} \tag{34.49}$$

Notice how the op-amp offset is autozeroed out. The ideal residual offset is V_{OS}/A_{OL} . Practically the residual offset is limited by the imperfections in the switches (which, once again, forces us to use fully-differential topologies). Also note, in Fig. 34.30, that we have drawn the input capacitance of the next stage as a load, C_L . This was so that the output of the S/H would appear to change only on the rising edge of ϕ_3 (plus the output settling time). Finally, if the S/H is clocked at $f_{clk} = 1/T_{clk}$, the output of the S/H must settle to less than 1/2 LSB, worst-case, in a time of $T_{clk}/2$. This means that the value of op-amp unity gain frequency given by Eq. (34.42) should be doubled.

Equation (34.49) can be used to determine the relationship between V_{in} and V_{out} for fully-differential signals

$$V_{out} = V_{out+} - V_{out-} = \left(1 + \frac{C_L}{C_F}\right) \cdot (V_{in+} - V_{in-}) \quad (34.50)$$

Note how, as we would expect, the common-mode voltage subtracts out of the relationship when we take the difference between V_{out+} and V_{out-} . A block diagram representing the S/H of Fig. 34.30 is shown in Fig. 34.32. The use of block diagrams, as we saw in Ch. 32 when discussing noise-shaping, can be very useful to describe data converter architectures.

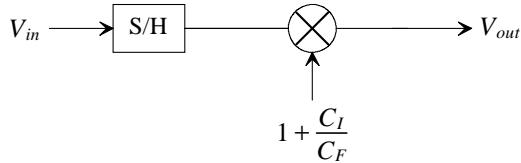


Figure 34.32 Block diagram for the S/H of Fig. 34.30.

Example 34.7

Simulate the operation of the data converter S/H building block shown in Fig. 34.30. Assume $C_t = C_F = 1 \text{ pF}$ and $f_s = 100 \text{ MHz}$.

The simulation results are shown in Fig. 34.33. In part (a) the clock signals are shown. Unlike the clock signals shown in Fig. 34.30 where the falling edge of ϕ_2 is delayed from ϕ_1 , the simulation sets the signals so they go low at the same time. This was to avoid the outputs of the op-amp changing to very large values for the small amount of time the op-amp operates open-loop with an input signal applied.

In part (b) we show the op-amp outputs. Note how, when ϕ_1 goes high, both outputs are set to the common-mode voltage by forcing the op-amp into a follower configuration (which may lead us to use switches to short the terminals of the op-amp to V_{CM} when ϕ_1 is high if offset isn't important). When ϕ_3 goes high, the circuit behaves as an S/H with a gain of two. Part (c) of the figure shows the outputs connected through ϕ_3 switches, as seen in Fig. 34.30, driving 10 pF load capacitances. ■

A Single-Ended to Differential Output S/H

Note how we have assumed in the S/H of Fig. 34.30 that the input voltage was fully-differential. In most practical situations at the input of an ADC, this isn't the case. While we can connect V_{in-} to V_{CM} in an attempt to change the single-ended input into a fully-differential sampled output, the practical problem is the variation of the op-amp's input common-mode voltage. As we've already discussed, precision data converters must use op-amp configurations where the input common-mode voltage is constant. Also, and

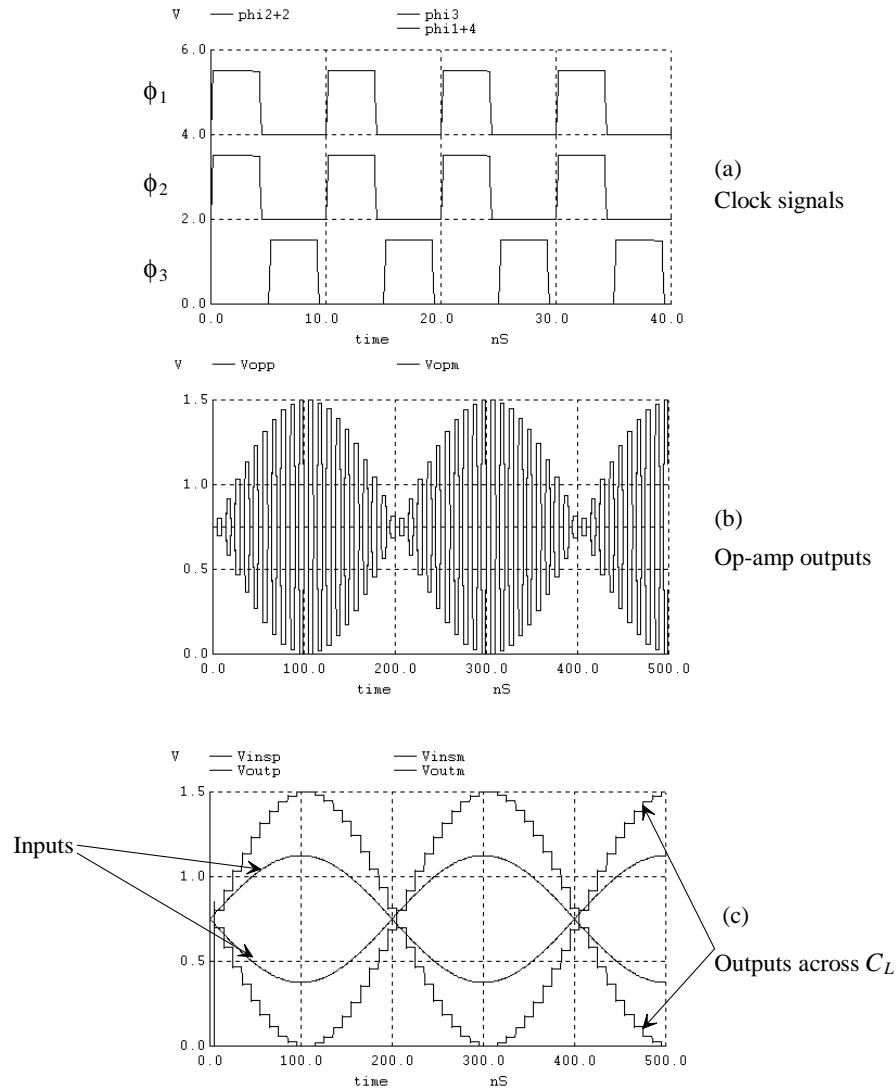


Figure 34.33 SPICE simulations of the operation of Fig. 34.30.

perhaps more practically, the range of allowable common-mode voltages can be very restricted when designing low-voltage circuits. As we saw with our basic mixed-signal op-amp in the previous chapter, Fig. 33.70, the minimum input common-mode voltage can be very close to V_{CM} when only an NMOS diff-pair is used. A technique to force the op-amp's input common-mode voltage to V_{CM} when a single-ended input is applied to the S/H is seen in Fig. 34.34 [8]. The error amplifier senses the op-amp's input common-mode

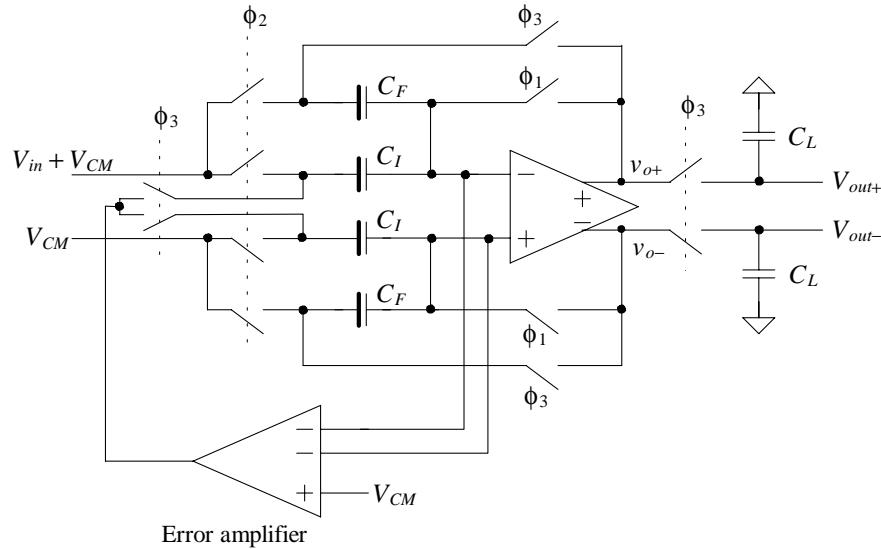


Figure 34.34 Single-ended to differential S/H.

voltage. It adjusts the value of the voltage applied to the bottom plates of the input capacitors when the ϕ_3 switches are closed until the common-mode voltage is approximately V_{CM} . We say "approximately" to indicate that we don't want the gain of the error amplifier to be so large that stability is a concern. The settling time of this circuit is not too important because any variation in its output simply represents a deviation from V_{CM} on the op-amp inputs. As long as the deviation is small and falls within the common-mode range of the op-amp, the single-ended to differential S/H functions properly. The addition of this error amplifier will increase the CMRR and thus reduce op-amp distortion (see Eq. [34.8]). Of course, when the ϕ_1 and ϕ_2 switches are closed, the op-amp's input common-mode voltage is $V_{CM} \pm V_{OS}$. Figure 34.35 shows a possible design for the error amplifier. The error amplifier is simply an operational transconductance amplifier.

A common-mode feedback (CMFB) circuit is still required to precisely balance the outputs of the op-amp. Figure 34.36 shows one possible design. In this figure we assume the reader is familiar with the designs and notation used in the last chapter for op-amp design in a submicron process. When the ϕ_1 switches are closed, the outputs are connected to the CMFB amplifier (see Fig. 34.36a). Also when the ϕ_1 switches are closed, from Figs. 34.30 or 34.34, the op-amp is placed in the unity feedback configuration. Because the gain of the op-amp is large, the inputs must be at the same voltage. The CMFB circuit is used to ensure that this voltage is V_{CM} ($\pm V_{OS}$). A typical mixed-signal op-amp is seen in Fig. 34.36b. This op-amp is derived from the topology given in the last chapter, without the output buffers. The benefit of removing the output buffers when driving purely capacitive loads is the ease of attaining a 90-degree phase margin (and so clean settling behavior).

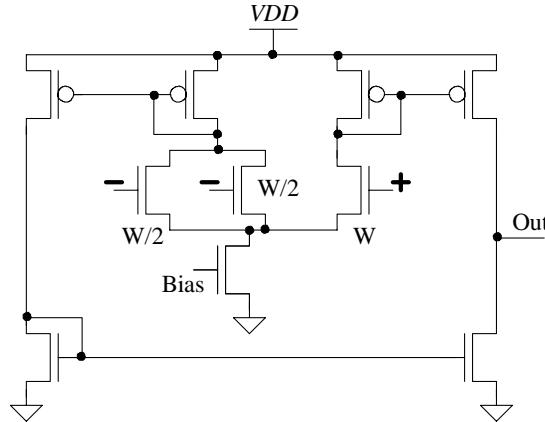


Figure 34.35 Schematic of the error amplifier.

The drawbacks of not using an output buffer are the reduced gain and the need to increase the biasing currents and device sizes to drive a given load capacitance. Again, the gain-boosting amplifiers labeled N and P in part (b) can be compensated, if needed, using capacitors at their outputs to ground (or VDD). If a basic diff-amp, as seen in Fig. 33.69, is used then in most situations no additional capacitance is needed. The CMFB amplifier is seen in Fig. 34.36c. It is simply a PMOS diff-amp with diode loads. The gain of this amplifier should be similar to the gain of the diff-amp used in the main op-amp so that the same load capacitances can be used for compensating both the op-amp and the CMFB loop. Note that when the ϕ_1 switches are open in Fig. 34.36a, the capacitors essentially average the outputs maintaining a balanced condition.

Example 34.8

Simulate the operation of the S/H shown in Fig. 34.31. Assume the S/H is clocked at 100 MHz, V_{in+} is a sinewave that swings from ground to VDD , and V_{in-} is connected to V_{CM} (the input signal is single-ended and covers the entire supply range). Show how the op-amp's input common-mode voltage range changes (that is, doesn't remain at V_{CM} as it would if the input signals were fully-differential). Show how a 10% mismatch in the two capacitors affects the output of the S/H.

Figure 34.37 shows the simulation results. In part (a) the input common-mode voltage of the op-amp is shown. When the ϕ_1 switches are closed, the voltage returns to V_{CM} (the op-amp is placed in a follower configuration where the input signal charges the two capacitors). In part (b) the rail-to-rail input signal is converted into a differential S/H output signal. Note that the gain of the S/H is one. Note also how, unlike the op-amp outputs in Fig. 34.33, the outputs are limited to $V_{CM} + V_p/2$ where V_p is the peak amplitude of the input sinewave ($= 0.75$ V here). When the input sinewave has an amplitude of 1.5 V (0.75 V above the

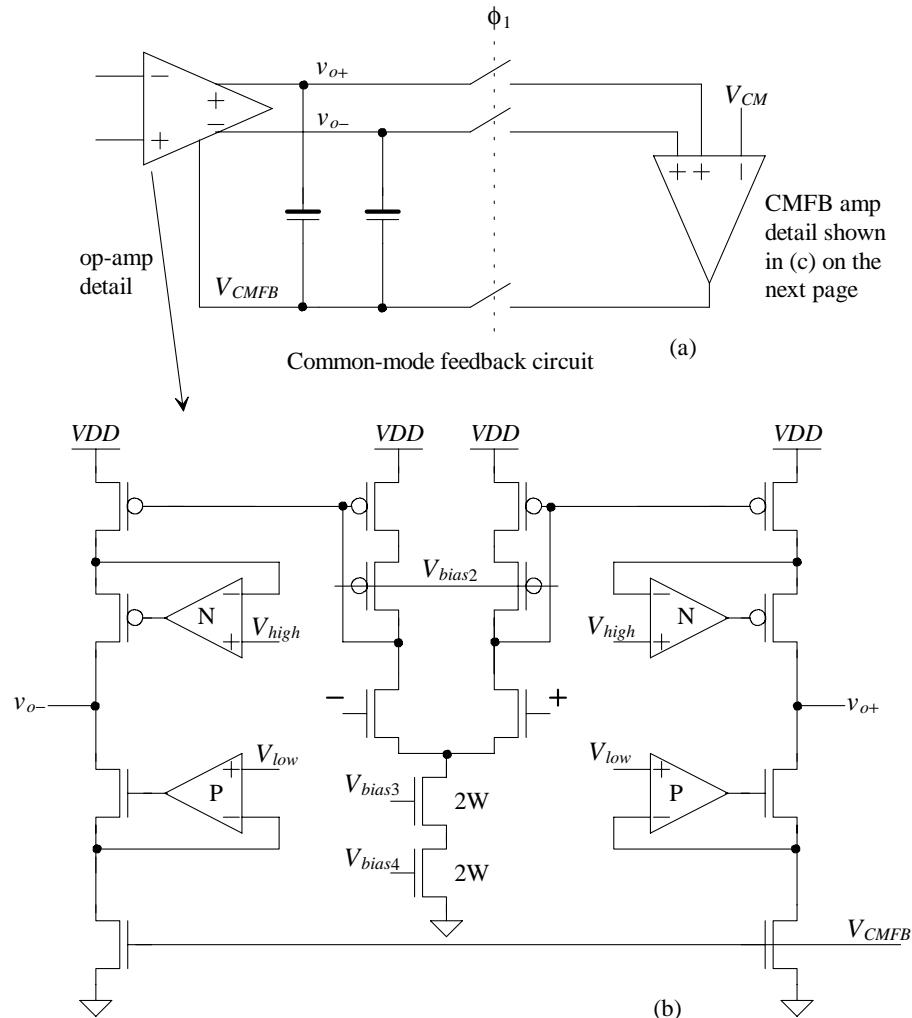
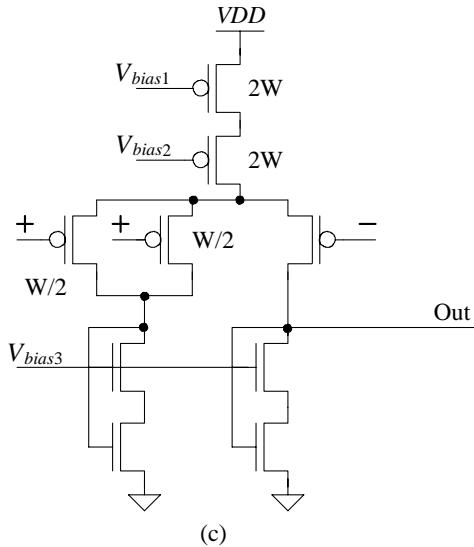
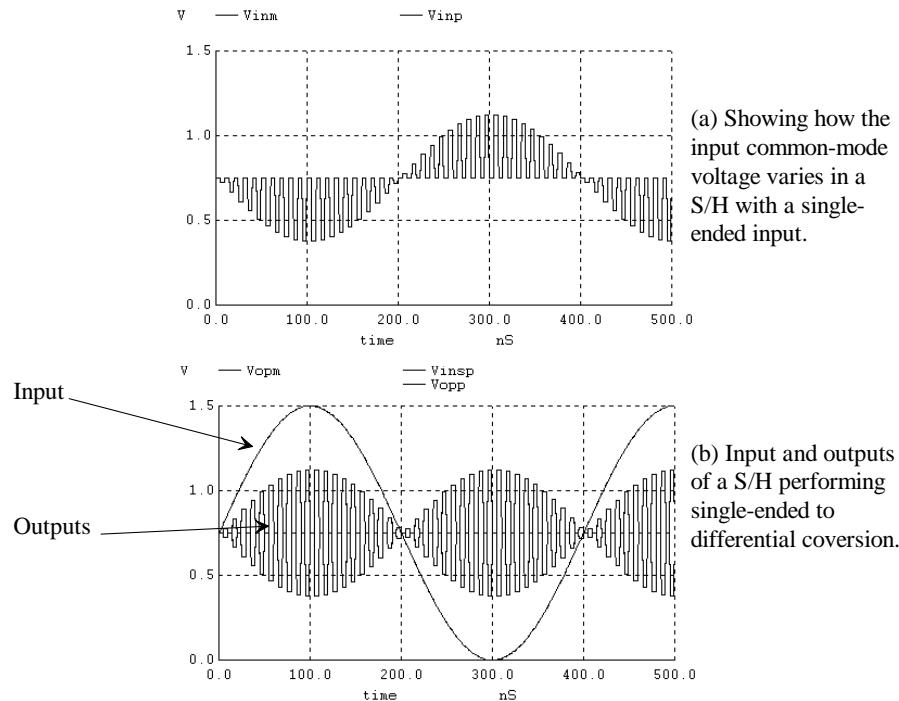


Figure 34.36 Mixed-signal op-amp for use in a S/H with CMFB.

V_{CM}), the positive output is 1.125 and the negative op-amp output is 0.375. Subtracting the op-amp outputs results in 0.75 V (the same voltage as the input signal referenced to V_{CM}). It's important to understand how going from a single-ended signal to a fully-differential signal results in a reduction in the op-amp output swing.

Finally, the simulation results were generated using 0.9 pF and 1.0 pF sampling capacitors (labeled C_F in Fig. 34.31). Because the feedback factor is unity, these capacitors will not affect the gain of the S/H. The point here is that the matching of the capacitors isn't important for a precise gain of one when using this (Fig. 34.31) S/H topology. (The op-amp open-loop gain, however, is still important.) ■

**Figure 34.36** (cont'd) CMFB amplifier circuit.**Figure 34.37** Simulation results for Ex. 34.8.

34.3.2 The Cyclic ADC

Cyclic or algorithmic DACs were first discussed back in Ch. 29. Here we present the concept of a cyclic ADC. A block diagram of a cyclic ADC is seen in Fig. 34.38 assuming an 8-bit ($N = 8$) conversion. The input signal is sampled on the rising edge of every *eighth* (N) clock pulse. On the rising edge of every clock pulse the comparator determines if the S/H input is above or below the common-mode voltage. If it is below V_{CM} , nothing is subtracted from the S/H output. If it is above V_{CM} , then V_{CM} is subtracted from the S/H output. In either case the resulting output is multiplied by two and cycled back to the S/H input. Each time the comparator output goes high the value is stored in a shift register. When the conversion is complete, the digital word stored in the shift register, which corresponds to the analog input voltage, is shifted into a hold register. The next conversion then begins on the following clock pulse starting with sampling the input voltage, V_{in} . Note that it takes N clock cycles for one conversion.

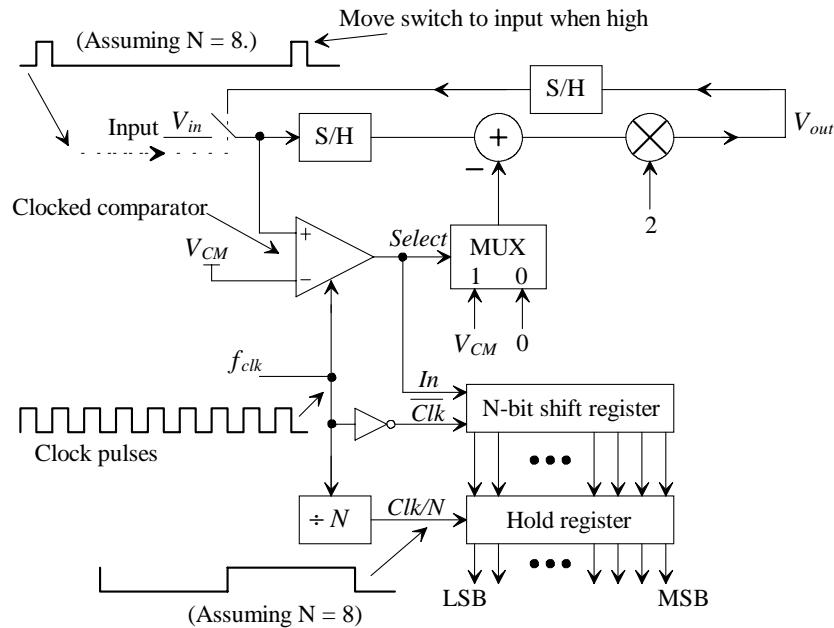


Figure 34.38 Block diagram of a cyclic ADC.

Example 34.9

Determine the output of the ADC in Fig. 34.38 if the input voltage is 1.5 V.

We begin by sampling the input voltage of 1.5 V. The output of the comparator is a logic 1 (MSB). Next, V_{CM} ($= 0.75$ V) is subtracted from the S/H output resulting

in an output of 0.75 V. This output is multiplied by 2 resulting in 1.5 V. This 1.5 V output (the output of the multiplier) is cycled back to the S/H input.

Next, we sample the fed back voltage of 1.5 V, the output of the comparator is, again, a logic 1 (MSB – 1). V_{CM} (= 0.75 V) is subtracted from the S/H output resulting in 0.75 V. This output is multiplied by 2 resulting in 1.5 V. This 1.5 V output (the output of the multiplier) is cycled back to the S/H input.

This continues and the final output of the ADC hold register is 1111 1111 (binary offset format). ■

Example 34.10

Repeat Ex. 34.9 if the Cyclic ADC input is 1.1 V.

1. Sample the 1.1 V input voltage. The comparator output goes high (MSB, b_7 , = 1). The output of the multiply by 2, after subtracting V_{CM} (= 0.75) from the S/H output, is 0.7 V.
2. Sample the 0.7 V fed back voltage. The comparator output goes low (b_6 = 0). The output of the multiplier is 1.4 V.
3. Sample 1.4 V. The comparator output goes high (b_5 = 1). The output of the multiplier is 1.3 V.
4. Sample 1.3 V. The comparator output goes high (b_4 = 1). The output of the multiplier is 1.1 V.
5. Sample 1.1 V (b_3 = 1) output of the multiplier is 0.7 V.
6. Sample 0.7 V (b_2 = 0) output of the multiplier is 1.4 V.
7. Sample 1.4 V (b_1 = 1) output of the multiplier is 1.3 V.
8. Sample 1.3 V (b_0 = 1) output of the multiplier is 1.1 V.
9. Sample the new input voltage and begin conversion again. The output word in the hold register is 1011 1011 (binary offset). ■

Comparator Placement

We showed the inverting input of the comparator in Fig. 34.38 connected to the common-mode voltage. In practice, however, we know that the comparator will have an offset or that the fed back signal may have a common-mode voltage slightly different than the ideal value. If the common-mode voltage of the fed back signals was, for example, 10 mV different than the ideal value, the comparator can make a wrong decision. Further, if the common-mode voltage is varying because of power supply, noise, or temperature changes, we can make a wrong decision even if some calibration scheme is employed. To avoid a wrong decision, the comparator is most often used in a fully-differential configuration, as seen in Fig. 34.39, with offset storage.

In Fig. 34.39 the clocked comparator shown in Fig. 33.34 is used without the NAND gates on the output. This comparator can have significant kickback noise. By

adding the ϕ_2 switches in series with the comparator input, we ensure the kickback noise doesn't corrupt the S/H input voltage. Note that since ϕ_3 and ϕ_2 are nonoverlapping, we guarantee that the comparator and S/H are disconnected when the comparator is clocked (and the kickback noise is generated). When the ϕ_1 switches are closed, the offset voltage of the comparator or the op-amp is zeroed out. The performance requirements of the comparator (gain and offset) can be greatly reduced (offset storage is not required) if we use 1.5 bits per clock cycle instead of the 1 bit per cycle used here [9, 10]. We discuss this further in the next section.

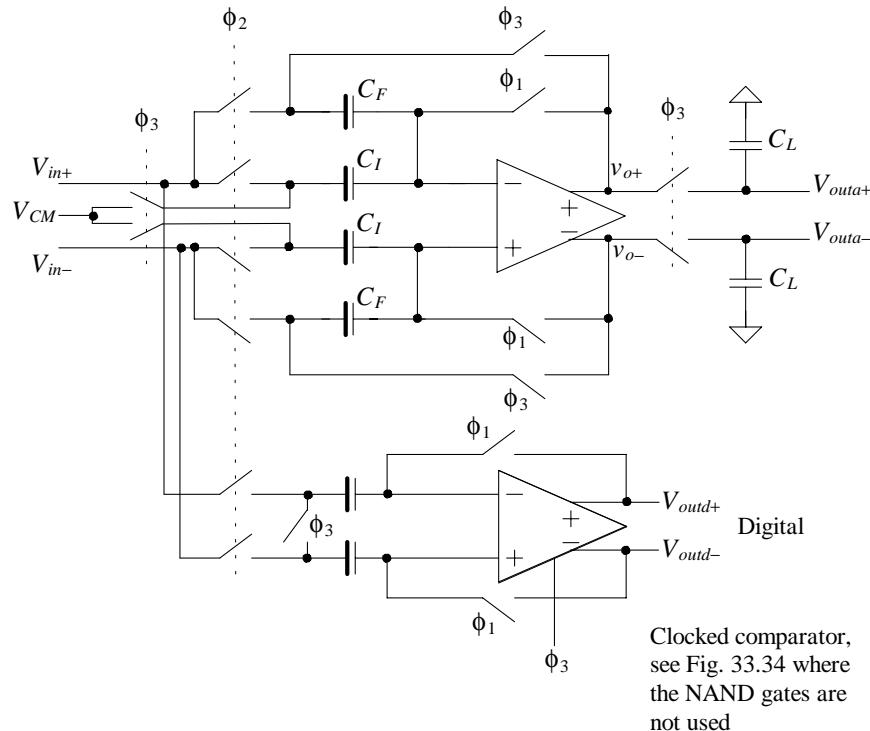


Figure 34.39 Implementation of the comparator with an S/H for use in a cyclic ADC.

Example 34.11

Estimate the gain required of a comparator used in a 10-bit cyclic ADC if $V_{REF+} = 1.5$ V and $V_{REF-} = 0$.

The LSB of this converter is $1.5/2^{10} = 1.465$ mV. This means that the comparator gain must be large enough so that the output can fully transition to either VDD or ground with less than 1.465 mV difference on its inputs (assuming the offset

voltage is zeroed out). In other words, the gain must be well above $1.5/1.465$ mV or 2^{10} or 1,024. Clearly this presents a real design concern. The gain of the positive feedback comparator of Fig. 33.34 may be very large because of the positive feedback used. However, the delay time of the comparator (time delay between the clock going high and the outputs transitioning all the way to VDD and ground) may be too long with such a little input voltage difference. Increasing the gain of the comparator, without care, can result in the comparator being unstable when placed in the unity feedback condition (the ϕ_1 switches closed in Fig. 34.39). To increase the comparator gain and avoid instability, a diff-amp (or two) can be added as a pre-amp in front of the basic comparator of Fig. 33.34. The offset of the diff-amp can then be zeroed out by placing the ϕ_1 switches between the diff-amp's output and its input. ■

Implementing Subtraction in the S/H

Notice in Fig. 34.38 how we can implement the S/H and then multiply by two by simply setting $C_F = C_I$ in Fig. 34.30. Reviewing Fig. 34.38 we see that it would also be nice to implement the subtraction in the S/H. In this figure we see that if the output of the MUX is 0 V, nothing needs to be changed in Fig. 34.30. However, if the MUX output is V_{CM} , then the S/H output must be reduced by V_{CM} . Consider what happens if, when ϕ_3 goes high, instead of connecting the bottom plate of C_I to V_{CM} in Fig. 34.30 we connect it to a voltage V_{CI+} (see Fig. 34.40). Doing this, after reviewing Eqs. (34.46) to (34.49), results in

$$Q_I^{\phi_3} = C_I \cdot (V_{CI+} - V_{CM} \pm V_{OS}) \quad (34.51)$$

or

$$V_{out+} = \left(1 + \frac{C_I}{C_F}\right) \cdot V_{in+} - \frac{C_I}{C_F} \cdot V_{CI+} \quad (34.52)$$

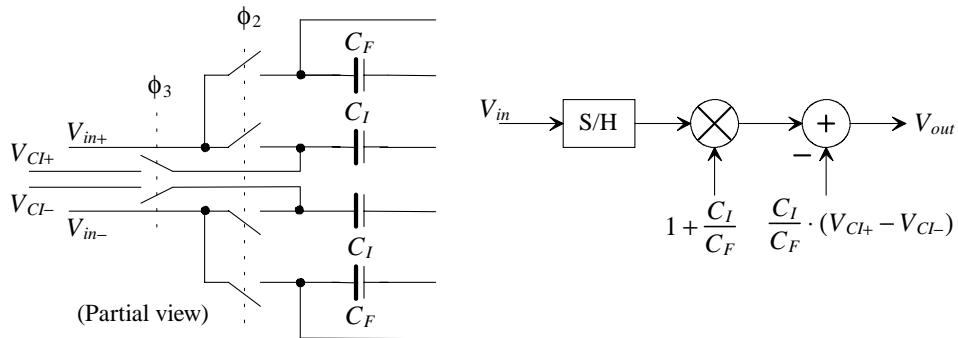


Figure 34.40 Implementing subtraction in the S/H.

The differential output voltage is then given by

$$V_{out} = V_{out+} - V_{out-} = \left(1 + \frac{C_I}{C_F}\right) \cdot (V_{in+} - V_{in-}) - \frac{C_I}{C_F} \cdot (V_{CI+} - V_{CI-}) \quad (34.53)$$

We can easily rearrange the block diagram of Fig. 34.40 so that it more closely resembles the block diagram of the cyclic converter (Fig. 34.38), as seen in Fig. 34.41. If $C_F = C_I$, for our needed gain of two, we end up subtracting V_{CM} when V_{CI+} is $1.5V_{CM}$ ($= 0.75VDD$) and V_{CI-} is $0.5V_{CM}$ ($= 0.25VDD$). For the fully-differential I/O signal case then we are actually *subtracting* $V_{CM}/2$ when $V_{in+} > V_{in-}$ and *adding* $V_{CM}/2$ when $V_{in+} < V_{in-}$.

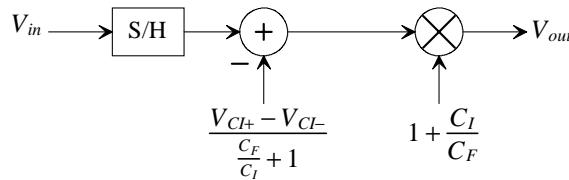


Figure 34.41 Block diagram of Fig. 34.30 with bottom plates of C_I tied to V_{CI} .

Example 34.12

Simulate the operation of the S/H shown in Fig. 34.42 if $f_s = 100$ MHz, $C_F = C_I = 1$ pF, $V_{CI+} = 1.5V_{CM}$, and V_{CI-} is $0.5V_{CM}$. Comment on the resulting output.

The simulation results are shown in Fig. 34.43. We only show the situation when we would want to subtract $V_{CM}/2$ from the differential input signal of the cyclic ADC, that is, when $V_{in+} > V_{in-}$. When the inputs are approximately the same voltage, the + input is approximately the same voltage as the - input and V_{in} is 0.

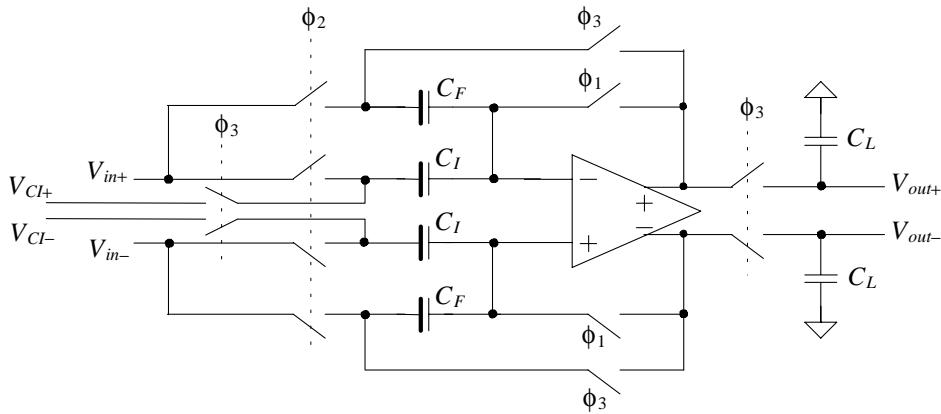


Figure 34.42 S/H used in Ex. 34.12

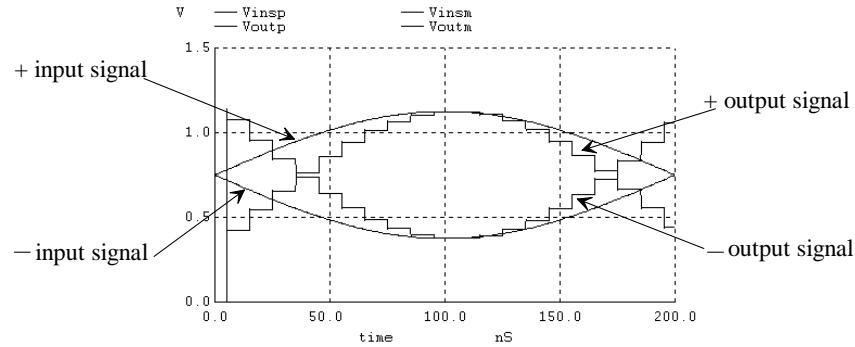


Figure 34.43 Simulation results for Ex. 34.12.

We subtract 0.375 V ($V_{CM}/2$) from the input and multiply by 2. The result, when the input is 0, is -0.75 V ($-V_{CM}$). When this happens, V_{out+} is 375 mV and V_{out-} is 1.125 V . Taking the difference in these signals results in -0.75 V . At 100 ns in Fig. 34.43, for example, V_{in} is $+V_{CM}$. After we subtract $V_{CM}/2$ and multiply by 2, we get V_{CM} again (as indicated in the figure). ■

Example 34.13

Repeat Ex. 34.12 if we want to add $V_{CM}/2$ to the input signal.

We only want to add $V_{CM}/2$ to the input signal when $V_{in+} < V_{in-}$. In this situation we set $V_{Cl+} = 0.5V_{CM}$ and V_{Cl-} to $1.5V_{CM}$. The simulation results are shown in Fig. 34.44. ■

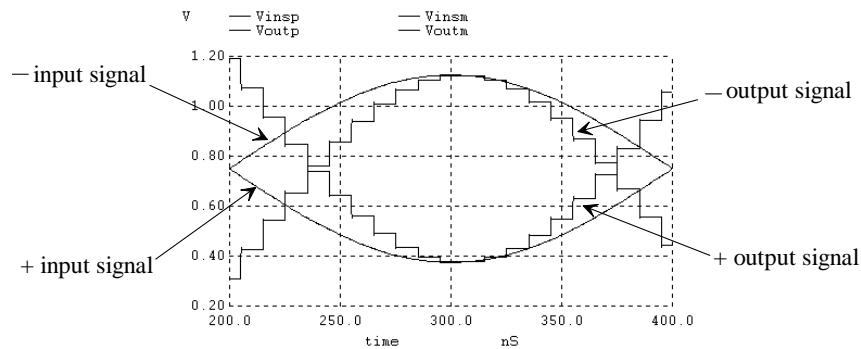


Figure 34.44 Simulation results for Ex. 34.13.

Let's write the analog output of the single-ended cyclic stage as

$$V_{out} = 2 \cdot (V_{in} + \bar{b} \cdot 0 - b \cdot V_{CM}) \quad (34.54)$$

where b is the digital (1 or 0) output of the comparator. Figure 34.45 shows the transfer curve for the cyclic stage. If the comparator output, b , is a 1 ($V_{in} > V_{CM}$), then we subtract V_{CM} from V_{in} before multiplying by two. Note that we have assumed

$$\frac{C_I}{C_F} = 1 \quad (34.55)$$

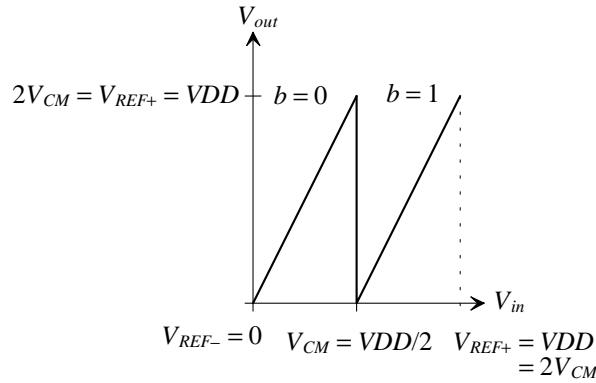


Figure 34.45 Transfer curve for the cyclic ADC (single-ended case).

Understanding Output Swing

After reviewing Fig. 34.45, the reader may wonder how the ADC will perform when the op-amp output must swing all the way up to VDD and down to ground. Any practical op-amp output voltage will become nonlinear as it approaches the supply rails. What must be realized is that the single-ended voltage, which ranges from 0 to VDD , is changed into a fully-differential voltage using the circuit of Fig. 34.34, which varies from $VDD/4$ to $3VDD/4$ (as seen in Fig. 34.37). If we use our common-mode voltage as a reference ($V_{CM} = VDD/2$) for single-ended inputs, then we can show some conversions from single-ended to differential.

Single-ended input	Differential outputs
1. $V_{in} = 0.5VDD$	$V_{out+} = V_{out-} = V_{CM}$, $V_{out} = V_{out+} - V_{out-} = 0$
2. $V_{in} = VDD$	$V_{out+} = 0.75VDD$, $V_{out-} = 0.25VDD$, or $V_{out} = VDD/2$
3. $V_{in} = 0$	$V_{out+} = 0.25VDD$, $V_{out-} = 0.75VDD$, or $V_{out} = -VDD/2$
4. $V_{in} = 0.6VDD$	$V_{out+} = 0.55VDD$, $V_{out-} = 0.45VDD$, or $V_{out} = 0.1VDD$

Figure 34.46 shows the transfer curve of Fig. 34.45 redrawn to indicate that the signals, both input and output, are fully-differential. Note, as seen in Fig. 34.39, that the comparator output transitions from a 0 to a 1 when $V_{in+} > V_{in-}$ ($V_{in} > 0$).

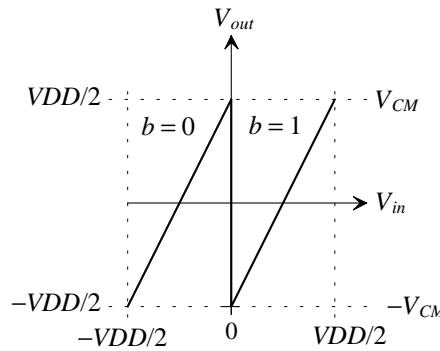


Figure 34.46 Transfer curve for the cyclic ADC when using fully-differential signals.

We can rewrite Eq. (34.54) for fully-differential signals by looking at Fig. 34.46 and noticing that now because the inputs and outputs are fully-differential and referenced to V_{CM} instead of 0 so must the voltages we add and subtract from the input. Equation (34.54) can be written as

$$V_{out+} - V_{out-} = 2 \cdot \left(V_{in+} - V_{in-} + \bar{b} \cdot 0 - b \cdot V_{CM} \right) + \bar{b} \cdot V_{CM} + b \cdot V_{CM} \quad (34.56)$$

or

$$V_{out+} - V_{out-} = 2 \cdot \left(V_{in+} - V_{in-} + \bar{b} \cdot 0 - b \cdot V_{CM} + \bar{b} \cdot \frac{V_{CM}}{2} + b \cdot \frac{V_{CM}}{2} \right) \quad (34.57)$$

and finally

$$\overbrace{V_{out+} - V_{out-}}^{V_{out}} = 2 \cdot \left(\overbrace{V_{in+} - V_{in-}}^{V_{in}} + \bar{b} \cdot \frac{V_{CM}}{2} - b \cdot \frac{V_{CM}}{2} \right) \quad (34.58)$$

Example 34.14

Using the fully-differential S/H stage of Fig. 34.42 simulate the transfer curve shown in Fig. 34.46.

The results of the simulation are shown in Fig. 34.47. To simulate V_{in} , the V_{in+} input is a ramp that varies from 0.375 V to 1.125 V while, at the same time, the V_{in-} is a ramp that varies from 1.125 V to 0.375 V. This results in V_{in} changing linearly from $-V_{CM}$ to $+V_{CM}$ (knowing $V_{CM} = VDD/2$). When the two ramps are equal, that is, when they are both V_{CM} ($V_{in} = 0$), V_{Cl+} changes from $0.5V_{CM}$ to

$1.5V_{CM}$, and V_{CL} changes from $1.5V_{CM}$ to $0.5V_{CM}$ so that we go from adding $V_{CM}/2$ when $V_{in} < 0$ to subtracting $V_{CM}/2$ when $V_{in} > 0$. ■

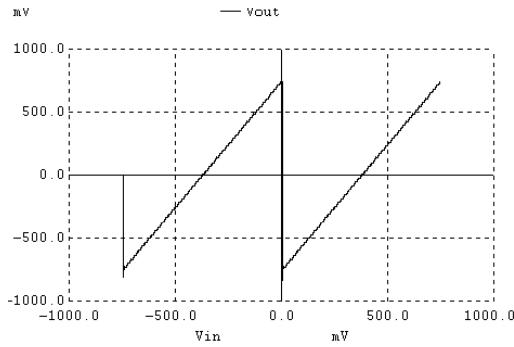


Figure 34.47 Simulating the transfer curves of a cyclic ADC stage.

34.3.3 The Pipeline ADC

One drawback of the cyclic ADC discussed in the previous section is the requirement of N clock cycles for each N -bit conversion. From Ch. 29 we know that the flash and pipeline (after an N -bit latency) topologies can perform an analog-to-digital conversion in one clock cycle. Another possibility for an N -bit conversion in one clock cycle is to use a time-interleaved topology [11]. The basic idea is seen in Fig. 34.48. The S/Hs are sequentially clocked so that during each clock cycle the input voltage, V_{in} , is sampled, held, and applied to the input of an N -bit ADC. If the outputs of each ADC are then sequentially available through a MUX, the overall topology behaves as if it were a single ADC with flash-like performance. The practical problem with this topology is the matching between the ADCs. Differences in the DC characteristics of each ADC, for example, can result in measuring digital output values that change with time when a DC input signal is applied (a ripple on the output similar to what was seen in a noise-shaping data converter output). Mismatches in the ADCs can also result in a reduced (from ideal) signal-to-noise plus distortion ratio (*SNDR*).

The pipelined ADC can be thought of as an amplitude-interleaved topology where errors from one stage are correlated with errors from previous stages. The basic block diagram implementation of an N -bit pipelined ADC using the cyclic stage (see Fig. 34.42 for example) is seen in Fig. 34.49. Instead of cycling the analog output of the 1 bit/stage section back to its input, we feed the output into the next stage. The stages are clocked with opposite phases of the master clock signal. The comparator outputs are labeled *digital* in the figure. The digital comparator outputs are delayed through latches so that the final digital output word corresponds to the input signal sampled N clock cycles earlier. The first stage in Fig. 34.49 must be N -bit accurate. It must amplify its analog output voltage, V_{N-1} , to within 1 LSB of the ideal value (after the subtraction of 0 or V_{CM} from the input signal and the multiplication by two). The second stage output, V_{N-2} , must

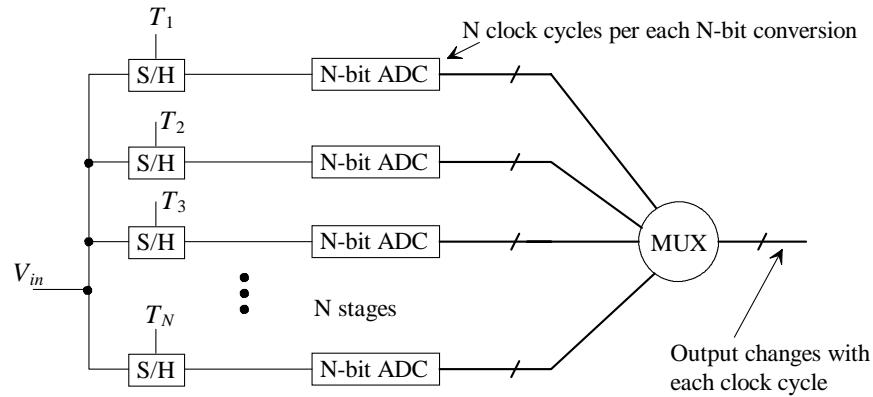


Figure 34.48 Time-interleaved operation of an ADC [10].

be an analog voltage within 2 LSBs of its ideal value. The third stage output, V_{N-3} , must be an analog voltage within 4 LSBs of its ideal value and so forth. The important point here is that because the required accuracy of each stage decreases as we move down the line, the settling time, gain accuracy, and offsets all become less important. Smaller (and thus lower power) stages can be used for the later stages having, possibly, a dramatic effect on both layout size and power dissipation. While we're showing 1 bit/stage in Fig. 34.49, most commercially available pipeline ADCs use the *digital error correction* present in the 1.5 bit/stage topology discussed later.

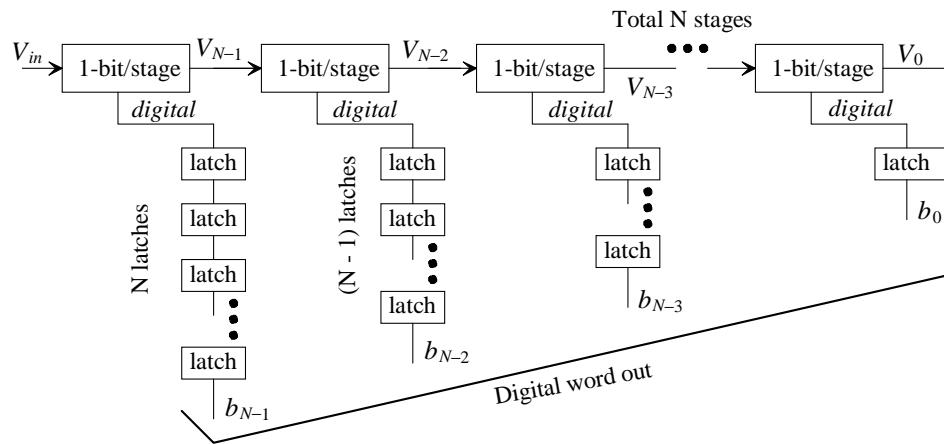


Figure 34.49 Pipelined ADC based on the cyclic stages discussed in the last section.

We know from our analysis of pipeline ADC errors in Ch. 29 that they can be the result of comparator offsets, gain or linearity errors, and amplifier offsets. In the remainder of this section we discuss how to reduce the effects of these errors. We begin with a discussion of using the 1.5 bit/stage topology to make the comparator offsets (and the reference voltages used with the comparators) a "don't care." While we might think that using six stages with 1.5 bits/stage would result in an ADC with 9-bit resolution, we find that the extra 0.5 bit/stage is used to correct for errors introduced by the comparators. Using six stages will still result in a 6-bit ADC. We then cover the use of capacitor error averaging [12] to set the gains of the amplifiers to precisely two. The cost of this technique is the increase in the number of clock cycles required for each stage's operation and a slightly more complex switching scheme. Finally, we cover some other topologies useful in amplifier offset removal and discuss offsets in general.

Using 1.5 Bits/Stage

As we saw in Ex. 34.11, the gain of the comparator (and the offset) can present a practical limitation to the operation of the ADC at high resolutions. The transfer curve of Fig. 34.45 relates the analog input of the cyclic ADC to its analog output voltage. The important point in this figure, besides the gain and linearity, is where the output of the comparator, b , transitions from a 0 to a 1. One-bit corresponds to two levels, i.e., a 0 or a 1. Two bits corresponds to four levels, i.e., 00, 01, 10, and 11. If we were to use three levels then we would get 1.5 bits of resolution. Using a thermometer code or decimal numbers, the three levels would then be

Thermometer, ab	Decimal
11	3
01	1
00	0

Next consider the transfer curves shown in Fig. 34.50 where three levels are used (1.5 bits). We can rewrite Eq. (34.54) for the 1.5 bit case as

$$V_{out} = 2 \cdot (V_{in} - \bar{a}\bar{b} \cdot 0 - \bar{a}b \cdot V_{CM} - ab \cdot 2V_{CM}) + V_{CM} \quad (34.59)$$

or

$$V_{out} = 2 \cdot (V_{in} + \bar{a}\bar{b} \cdot \frac{V_{CM}}{2} - \bar{a}b \cdot \frac{V_{CM}}{2} - ab \cdot \frac{3V_{CM}}{2}) \quad (34.60)$$

or if $ab = 00$ ($V_{in} < V_{CM}/2$), then $V_{out} = 2 \cdot (V_{in} + V_{CM}/2)$, if $ab = 01$ ($V_{CM} < V_{in} < 3V_{CM}/2$) then $V_{out} = 2 \cdot (V_{in} - 0.5V_{CM})$, and if $ab = 11$ then $V_{out} = 2 \cdot (V_{in} - 1.5V_{CM})$. For the fully-differential situation Eq. (34.59) can be rewritten as

$$V_{out+} - V_{out-} = 2 \cdot (V_{in+} - V_{in-} + \bar{a}\bar{b} \cdot V_{CM} - \bar{a}b \cdot 0 - ab \cdot V_{CM}) \quad (34.61)$$

where all we did was reference, to V_{CM} , the voltages added/subtracted to V_{in} as seen in Eq. (34.58).

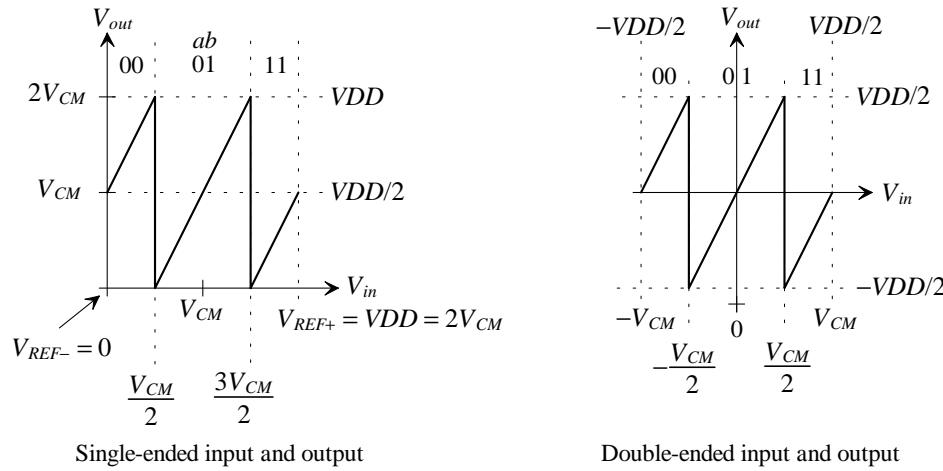


Figure 34.50 Transfer curves for using 1.5-bits per clock cycle.

Figure 34.51a shows how the comparators would be set up to determine ab and how we would provide the outputs. The outputs of the comparators are used as address inputs to a MUX. The MUX is used to set the bottom plate voltage of the C_l capacitor in Fig. 34.42. Figure 34.51b shows how we would implement the comparators if the input and output signals are double-ended.

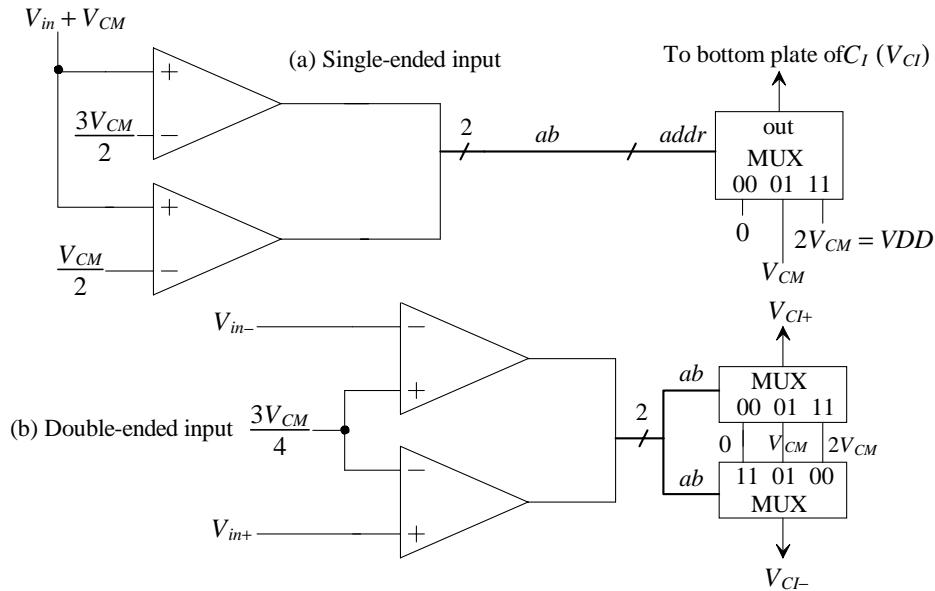


Figure 34.51 Implementing comparators and MUX for 1.5 bits.

Example 34.15

Repeat Ex. 34.14 if 1.5 bits/stage are used.

The simulation results are shown in Fig. 34.52. The same signals were used for the inputs (two ramps) here as used in Ex. 34.14. The voltage V_{Cl+} is 0 V when V_{in} is less than -375 mV and is 0.75 when V_{in} is between -375 mV and $+375$ mV, while it is 1.5 V ($= VDD = 2V_{CM}$) when V_{in} is greater than $+375$ mV. Notice how we only need three precision voltages, unlike the 1-bit/stage case, that is, VDD , V_{CM} , and 0. ■

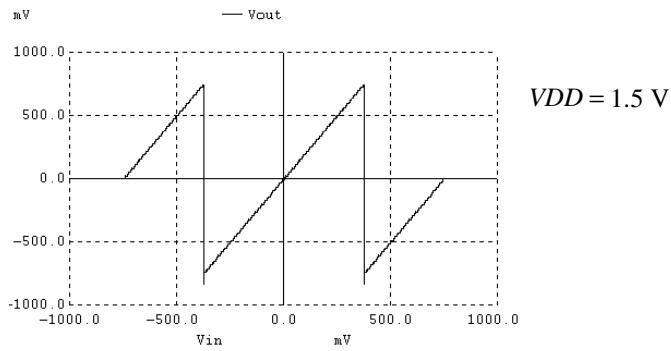


Figure 34.52 Simulating the transfer curves for 1.5 bits/stage.

Before going any further, let's answer, *"How can using 1.5 bits/stage eliminate the need for precision comparators?"* After reviewing Fig. 34.50, we see that an output of 11 cannot be followed by another output of 11 because we subtract V_{CM} from the input prior to multiplication by two. Even if the comparator has a reasonable offset (say less than 100 mV) or low gain (say less than 50) resulting in a wrong decision, it's impossible for a 11 to be followed by another 11 or for a 00 to be followed by another 00. It is possible, however, to get a continuous string of 01 outputs (a simple example is when V_{CM} is applied to the ADC).

Let's now discuss how the outputs of the 1.5-bit stage are combined into the final ADC output word. Let's assume, again, that $V_{REF+} = VDD = 1.5$ V, $V_{REF-} = 0$, and $V_{CM} = 0.75$ V. We know that the final N -bit output word can be converted back into an output voltage (with the unwanted quantization noise) using a DAC with the following weighting, Eq. (30.27),

$$V_{out} \text{ (if a DAC or } V_{in} \text{ if an ADC)} = b_{N-1} \cdot V_{CM} + b_{N-2} \cdot \frac{V_{CM}}{2} + b_{N-3} \cdot \frac{V_{CM}}{4} + \dots + b_0 \cdot \overbrace{\frac{V_{CM}}{2^{N-1}}}^{1 \text{ LSB}}$$

(34.62)

Reviewing Eq. (34.54) note how if, on the first cycle, $b = 1$, we subtract V_{CM} from the input and then multiply the result by two. After a little thought, we should be able to see how we derive Eq. (34.62) from Eq. (34.54) after N clock cycles.

If the first thermometer code output of the 1.5-bit stage is labeled $a_{1.5N-1}b_{1.5N-1}$ and the second output is $a_{1.5N-2}b_{1.5N-2}$, etc., then we can write, with the help of Eq. (34.59), the relation between the ADC input (analog) and the ADC outputs (digital) as

$$\begin{aligned} V_{in} &= \overline{a_{1.5N-1}}b_{1.5N-1} \cdot V_{CM} + a_{1.5N-1}b_{1.5N-1} \cdot 2V_{CM} - \frac{V_{CM}}{2} \\ &\quad + \overline{a_{1.5N-2}}b_{1.5N-2} \cdot \frac{V_{CM}}{2} + a_{1.5N-2}b_{1.5N-2} \cdot \frac{2V_{CM}}{2} - \frac{V_{CM}}{4} \\ &\quad + \overline{a_{1.5N-3}}b_{1.5N-3} \cdot \frac{V_{CM}}{4} + a_{1.5N-3}b_{1.5N-3} \cdot \frac{2V_{CM}}{4} - \frac{V_{CM}}{8} + \dots \end{aligned} \quad (34.63)$$

noting that we can group

$$-\frac{V_{CM}}{2} - \frac{V_{CM}}{4} - \frac{V_{CM}}{8} - \dots = -V_{CM} \cdot \underbrace{\frac{2^N - 1}{2^N}}_{=0 \text{ here}} = -V_{CM} + 0.5 \text{ LSB} + \frac{V_{REF-}}{2^N} \quad (34.64)$$

which is nothing more than a level shift. Clearly we can combine outputs in the following manner to arrive at an equation similar to Eq. (34.62)

$$\begin{aligned} V_{in} &= (\overline{a_{1.5N-1}}b_{1.5N-1} + a_{1.5N-2}b_{1.5N-2}) \cdot V_{CM} + (\overline{a_{1.5N-2}}b_{1.5N-2} + a_{1.5N-3}b_{1.5N-3}) \cdot \frac{V_{CM}}{2} \\ &\quad + (\overline{a_{1.5N-3}}b_{1.5N-3} + a_{1.5N-4}b_{1.5N-4}) \cdot \frac{V_{CM}}{4} + \dots \\ &\quad a_{1.5N-1}b_{1.5N-1} \cdot 2V_{CM} - V_{CM} + 0.5 \text{ LSB} \end{aligned} \quad (34.65)$$

Next notice that, when using a thermometer code, the only time $a_{1.5N-X}b_{1.5N-X}$ (the logical AND of ab) can be high is when both are high. The term $a_{1.5N-X}$ cannot be high while $b_{1.5N-X}$ is low (there is no such output code as 10, see Fig. 34.50). This means that we can replace $a_{1.5N-X}b_{1.5N-X}$ with simply $a_{1.5N-X}$. We can rewrite Eq. (34.65) as

$$\begin{aligned} V_{in} &= a_{1.5N-1} \cdot 2V_{CM} + (\overline{a_{1.5N-1}}b_{1.5N-1} + a_{1.5N-2}) \cdot V_{CM} + (\overline{a_{1.5N-2}}b_{1.5N-2} + a_{1.5N-3}) \cdot \frac{V_{CM}}{2} \\ &\quad + (\overline{a_{1.5N-3}}b_{1.5N-3} + a_{1.5N-4}) \cdot \frac{V_{CM}}{4} + \dots + \overline{a_{1.50}}b_{1.50} \cdot \frac{V_{CM}}{2^{N-1}} - (V_{CM} - 0.5 \text{ LSB}) \end{aligned} \quad (34.66)$$

The $+$ symbol in Eq. (34.66) indicates addition rather than a logical OR. If $\overline{a_{1.5N-1}}b_{1.5N-1} = 1$ and $a_{1.5N-2} = 1$, then the second term in this equation is $2V_{CM}$ and the first term must be 0. When this occurs, the addition of the two terms is 0 and a carry is generated. We can now use this information to write the relationship between Eq. (34.62) and Eq. (34.63) knowing \oplus indicates an exclusive OR

$$b_0 = \overline{a_{1.50}}b_{1.50} \text{ with carry} = c_0 = a_{1.50} \quad (34.67)$$

$$b_1 = \overline{a_{1.51}}b_{1.51} \oplus c_0 \text{ with } c_1 = \overline{a_{1.51}}b_{1.51}c_0 \quad (34.68)$$

$$b_2 = \overline{a_{1.52}}b_{1.52} \oplus a_{1.51} \oplus c_1 \text{ with } c_2 = \overline{a_{1.52}}b_{1.52}a_{1.51} + c_1(\overline{a_{1.52}}b_{1.52} + a_{1.51}) \quad (34.69)$$

noting each bit and carry are the outputs of a full adder. To simplify the carry equation, and the subsequent equations, we can substitute c_1 to get

$$c_2 = \overline{a_{1.52}}b_{1.52}a_{1.51} + c_1\overline{a_{1.52}}b_{1.52} \quad (34.70)$$

We can write the general form of b_2 through b_{N-1} , using full adders, as

$$b_{N-1} = \overline{a_{1.5N-1}}b_{1.5N-1} \oplus a_{1.5N-2} \oplus c_{N-2} \text{ and}$$

$$c_{N-1} = \overline{a_{1.5N-1}}b_{1.5N-1}a_{1.5N-2} + c_{N-2}(\overline{a_{1.5N-1}}b_{1.5N-1} + a_{1.5N-2}) \quad (34.71)$$

The bit b_N has a weighting of $2V_{CM}$ and thus the final output word size is $N + 1$

$$b_N = a_{1.5N-1} \oplus c_{N-1} \quad (34.72)$$

Before we sketch the implementation of this digital circuit, let's make a few comments. To begin, notice that the word size is one larger than N (the resolution). In the 1 bit/stage circuit our maximum output is (assuming $N = 8$)

$$1111\ 1111 = VDD - 1 \text{ LSB} = 2V_{CM} - 1 \text{ LSB} \quad (1 \text{ bit/stage})$$

Now our maximum output is

$$1\ 0000\ 0000 = VDD = 2V_{CM} \quad (1.5 \text{ bit/stage})$$

The resolution is still essentially eight bits; we just have a slightly larger (1 LSB) output range. To make the words exactly match, we can throw out the MSB and assume 1 0000 0000 is an out-of-range condition. Note that an input of $VDD - 1$ LSB using 1.5 bits/stage gives an output code of 0 1111 1111.

One more comment: if we go through the logic in Eq. (34.66), we don't get the correct outputs unless we subtract $V_{CM} - 0.5$ LSB. The binary offset representation can be written as

$$V_{CM} - 0.5 \text{ LSB} = 00111111... \quad (34.73)$$

For example, applying V_{CM} (single-ended, see Fig. 34.51) to the ADC input results in a continuous output (ab) of 01. The output prior to subtraction, from Eqs. (34.67) to (34.72), is then 01111111... ($b_N = 0$, $b_{N-1} = 1$, $b_{N-2} = 1$, etc.). After subtracting 00111111..., we get 01000000 or V_{CM} knowing the weighting of the second bit is V_{CM} .

Figure 34.53 shows one possible implementation of Eqs. (34.67)-(34.72) for a cyclic ADC. The state shown, i.e., the ab values, is valid at the end of the conversion (after N clock cycles). When starting the algorithm (on the first rising edge of clock), all latches are reset to zeroes and the first comparator outputs, ab , are applied. On the second rising edge of clock the output b_2 corresponds to the final b_N . After N clock cycles, the hold register is clocked (and the latches are reset). The final ADC output is the contents of the hold register after subtracting 00111111.... Changing the numbers to two's complement may be useful when implementing this stage (assuming the MSB has a weighting of V_{CM} , i.e., throw out b_N). Note the subtraction can precede the hold register.

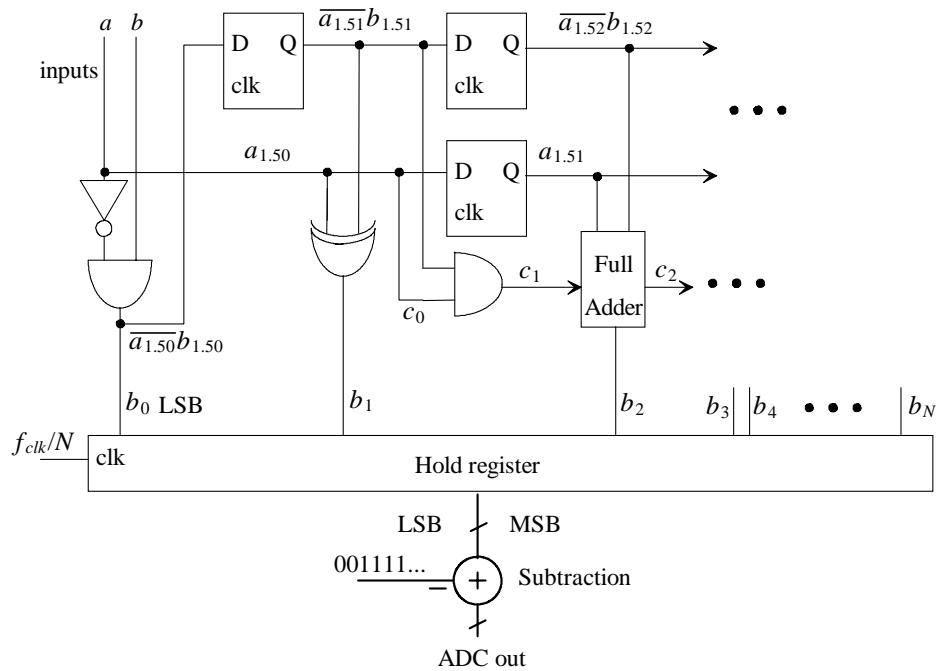


Figure 34.53 Combining the outputs of the cyclic ADC when 1.5 bits/stage is used.

Example 34.16

Repeat Ex. 34.10 if 1.5 bits/stage are used. Assume the converter is ideal and the comparators switch precisely at $V_{CM}/2$ ($= 0.375$ V here) and $3V_{CM}/2$ ($= 1.125$ V here). Assume all latches initially contain zeroes.

V_{in}	Comparator outputs, ab	V_{out}	Digital out
1.1 ($N - 1 = 7$)	01	1.45	$b_7 = 0 \ c_7 = 1 \ b_8 = 1$
1.45 ($N - 2 = 6$)	11	0.65	$b_6 = 0 \ c_6 = 0$
0.65 ($N - 3 = 5$)	01	0.55	$b_5 = 1 \ c_5 = 0$
0.55 ($N - 4 = 4$)	01	0.35	$b_4 = 1 \ c_4 = 0$
0.35 ($N - 5 = 3$)	00	1.45	$b_3 = 1 \ c_3 = 0$
1.45 (2)	11	0.65	$b_2 = 0 \ c_2 = 0$
0.65 (1)	01	0.55	$b_1 = 1 \ c_1 = 0$
0.55 (0)	01	0.35	$b_0 = 1 \ c_0 = 0$

We can reorder the bits so the MSB is on the left, the LSB is on the right, and subtract 0 0111 1111 yielding

$$\begin{array}{r} 1\ 0011\ 1011 \text{ (315 decimal)} \\ - 0\ 0111\ 1111 \text{ (127)} \\ \hline 0\ 1011\ 1100 \text{ (188)} \end{array}$$

This is the result given in Ex. 34.10 (0 1011 1011) plus 1 LSB. The 1 LSB discrepancy can be traced to Eq. (34.66) where we used 0.5 LSBs. Because our resolution is at best 1 LSB, sometimes the result will experience a round-off error. To understand this in the subtraction above, the more correct decimal representation of $V_{CM} - 0.5$ LSBs is 127.5 and the more correct decimal output is 187.5 (the answer given in Ex. 34.10 was decimal 187). When we discussed the output of the ADC with an input voltage of V_{CM} , on the bottom of page 365, our output prior to subtraction was 01111... For eight bits this is decimal 255 (nine bits out). When we subtract 127, we get the correct output of 128 for the final output. In this situation the round-off worked in our favor (we round 127.5 up to 128). Because of this 0.5-bit offset, one might simply subtract V_{CM} ($= 0\ 1000\ 000\dots$) to simplify the subtraction circuitry. While this appears complicated, the significant benefit is relaxed comparator and reference voltage (used with the comparators) requirements. ■

Example 34.17

Repeat Ex. 34.16 if the comparators switch at 0.4 V (a 25 mV offset) and 1.05 V (a 75 mV offset).

V_{in}	Comparator outputs, ab	V_{out}	Digital out
1.1 ($N - 1 = 7$)	11	-0.05	$b_7 = 0\ c_7 = 0\ b_8 = 1$
-0.05 ($N - 2 = 6$)	00	0.65	$b_6 = 0\ c_6 = 0$
0.65 ($N - 3 = 5$)	01	0.55	$b_5 = 1\ c_5 = 0$
0.55 ($N - 4 = 4$)	01	0.35	$b_4 = 1\ c_4 = 0$
0.35 ($N - 5 = 3$)	00	1.45	$b_3 = 1\ c_3 = 0$
1.45 (2)	11	0.65	$b_2 = 0\ c_2 = 0$
0.65 (1)	01	0.55	$b_1 = 1\ c_1 = 0$
0.55 (0)	01	0.35	$b_0 = 1\ c_0 = 0$

Which is the exact same result! While the comparator performance can be extremely poor, the circuit of Fig. 34.42 must subtract and amplify to an accuracy set by the least significant bit of the converter. When we calculated values in Ex. 34.16 and here we assumed subtractions of exactly 0, V_{CM} , and $2V_{CM}$ followed by a multiplication of exactly two. Finally, notice that the negative output of -0.05 V (single-ended) is easily accommodated when using fully-differential op-amps. ■

Capacitor Error Averaging

While using 1.5 bits/stage has made the comparator offset unimportant, we still have to amplify the signal by a precise factor of 2. Toward the goal of precision gain consider the redrawn version of our basic S/H, Fig. 34.42, shown in Fig. 34.54. In this figure we've shown the clock phases (but not the slightly delayed clock signals also used) and the capacitors with mismatch. Ideally, $\Delta C_{+,-} = 0$ and the gain of the S/H is precisely 2 (assuming sufficiently high op-amp open-loop gain, see Eq. [34.38]). From Eq. (34.52) we can write

$$V_{out+} = \left(2 + \frac{\Delta C_+}{C_+}\right) \cdot V_{in+} - \left(1 + \frac{\Delta C_+}{C_+}\right) V_{CI+} \quad (34.74)$$

where $\Delta C/C$ is the capacitor mismatch (say, $\pm 1\%$ or ± 0.01 , noting ΔC may be positive or negative). For the negative output

$$V_{out-} = \left(2 + \frac{\Delta C_-}{C_-}\right) \cdot V_{in-} - \left(1 + \frac{\Delta C_-}{C_-}\right) V_{CI-} \quad (34.75)$$

where $V_{out} = V_{out+} - V_{out-}$. Note that when using this topology in a pipeline converter one stage can be in the hold state while the next stage can be in the sampling state effectively sharing the time. The result of this sharing is the need for only one clock cycle for each stage in the conversion.

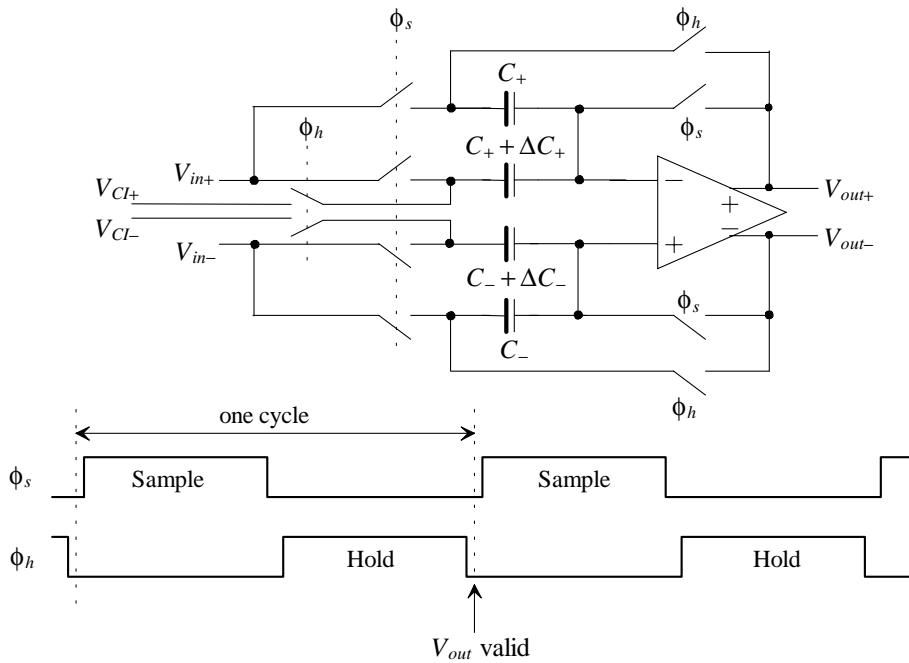


Figure 34.54 S/H of Fig. 34.42 with mismatched capacitors.

Next consider what happens if, instead of sampling the input voltage again, we simply switch the positions of the C and $C + \Delta C$ capacitors, that is, we connect C to V_{CI} and $C + \Delta C$ to V_{out} . Figure 34.55 shows this switch and the modified S/H using an extra half-clock cycle. The sample and amplify phases of the clock are exactly the same as before, and so Eqs. (34.74) and (34.75) are still valid. We denote the outputs at the end (falling edge) of the amplify phase as V_{outa+} and V_{outa-}

$$V_{outa+} = 2 \cdot V_{in+} - V_{CI+} + \frac{\Delta C_+}{C_+} \cdot (V_{in+} - V_{CI+}) \quad (34.76)$$

and

$$V_{outa-} = 2 \cdot V_{in-} - V_{CI-} + \frac{\Delta C_-}{C_-} \cdot (V_{in-} - V_{CI-}) \quad (34.77)$$

where the ideal situation is $\Delta C = 0$.

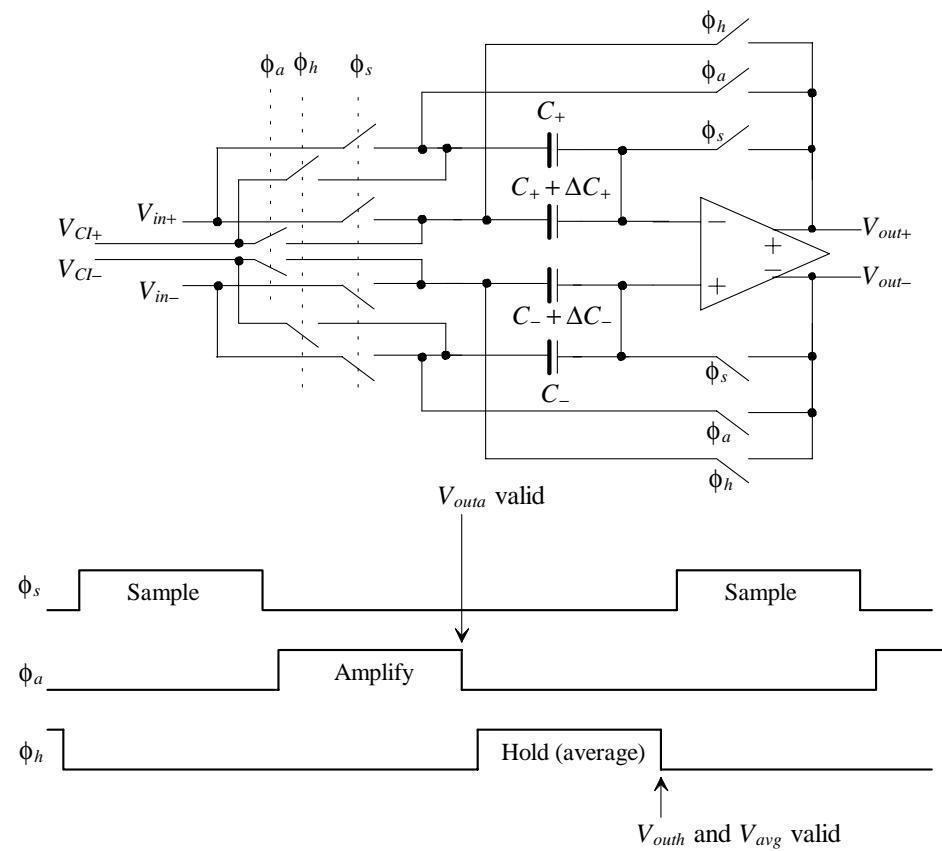


Figure 34.55 S/H using capacitor error averaging.

At the end of the amplify phase the charge on the capacitors is (assuming the op-amp input voltages are 0 and only looking at the + path to simplify the equations)

$$Q_{a+} = (C_+ + \Delta C_+) \cdot (V_{Cl+}) + C_+ \cdot (V_{outa+}) \quad (34.78)$$

and at the end of the hold phase

$$Q_{h+} = (C_+ + \Delta C_+) \cdot (V_{outh+}) + C_+ \cdot (V_{Cl+}) \quad (34.79)$$

Because charge must be conserved, $Q_{a+} = Q_{h+}$, and therefore

$$V_{outh+} = V_{Cl+} + \frac{C_+}{C_+ + \Delta C_+} \cdot V_{outa+} - \frac{C_+}{C_+ + \Delta C_+} \cdot V_{Cl+} \quad (34.80)$$

It will be useful to use

$$\frac{C_+}{C_+ + \Delta C_+} = \frac{1}{1 + \Delta C_+/C_+} \approx 1 - \frac{\Delta C_+}{C_+} \quad (34.81)$$

Rewriting Eq. (34.80) gives

$$V_{outh+} = \left(1 - \frac{\Delta C_+}{C_+}\right) \cdot V_{outa+} + \frac{\Delta C_+}{C_+} \cdot V_{Cl+} \quad (34.82)$$

Substituting Eq. (34.76) for V_{outa+} results in

$$V_{outh+} = 2V_{in+} - V_{Cl+} - \frac{\Delta C_+}{C_+} \cdot (V_{in+} - V_{Cl+}) \quad (34.83)$$

where the terms containing $(\Delta C_+/C_+)^2$ are assumed negligible. If the matching is 1%, then $(\Delta C_+/C_+)^2 = 10^{-4}$.

Clearly averaging V_{outa+} (Eq. [34.76]) and V_{outh+} (Eq. [34.83]) results in the precise desired gain. The question now becomes: "How do we perform the averaging without introducing more error?" Ideally we want

$$\begin{aligned} V_{avg+} &= \frac{V_{outa+} + V_{outh+}}{2} \\ &= \frac{1}{2} \cdot \left(2V_{in+} - V_{Cl+} + \frac{\Delta C_+}{C_+} \cdot (V_{in+} - V_{Cl+}) + 2V_{in+} - V_{Cl+} - \frac{\Delta C_+}{C_+} \cdot (V_{in+} - V_{Cl+}) \right) \\ &= 2V_{in+} - V_{Cl+} \end{aligned} \quad (34.84)$$

or more generally

$$V_{avg} = V_{avg+} - V_{avg-} = 2(V_{in+} - V_{in-}) - (V_{Cl+} - V_{Cl-}) \quad (34.85)$$

This equation should be compared to Eq. (34.61).

Consider the averaging amplifier shown in Fig. 34.56. We can write the charge on the four capacitors when the ϕ_a switches are closed (actually at the falling edge of the ϕ_a clock, assuming complete settling) as

$$Q_+^{\phi_a} = (V_{outa+} - V_{CM} \pm V_{OS}) \cdot C_F + (V_{outa-} - V_{CM} \pm V_{OS}) \cdot C_I \quad (34.86)$$

and

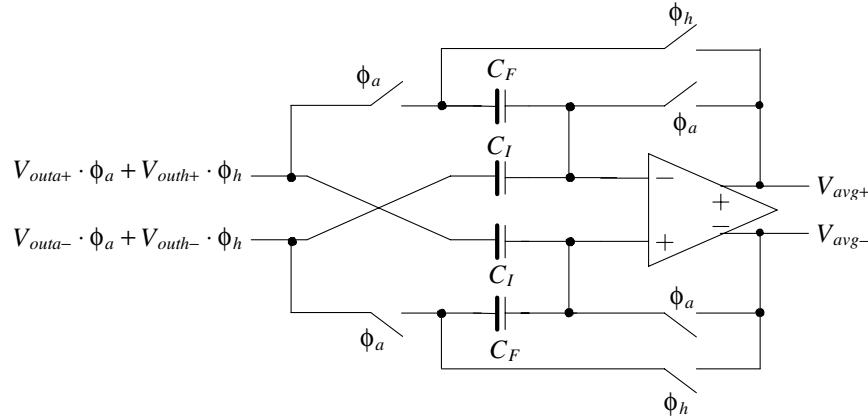


Figure 34.56 Averaging amplifier for use on the output of Fig. 34.55.

$$Q_-^{\phi_a} = (V_{outa-} - V_{CM} \pm V_{OS}) \cdot C_F + (V_{outa+} - V_{CM} \pm V_{OS}) \cdot C_I \quad (34.87)$$

Note that the common-mode voltage and op-amp offset subtract out when we take the difference between the balanced signals and so we do not include V_{CM} or V_{OS} in the remaining analysis. (The offset from the common-mode feedback circuit results in an unharful variation in V_{CM} and is discussed later in this section.) On the falling edge of ϕ_h , and following the same procedure as used in Eq. (34.48), we can write

$$C_F \cdot V_{avg+} = C_F \cdot V_{outa+} + C_I \cdot V_{outa-} - C_I \cdot V_{outh-} \quad (34.88)$$

and

$$C_F \cdot V_{avg-} = C_F \cdot V_{outa-} + C_I \cdot V_{outa+} - C_I \cdot V_{outh+} \quad (34.89)$$

We can then write

$$V_{avg+} = V_{outa+} + (V_{outa-} - V_{outh-}) \cdot \frac{C_I}{C_F} \quad (34.90)$$

$$V_{avg-} = V_{outa-} + (V_{outa+} - V_{outh+}) \cdot \frac{C_I}{C_F} \quad (34.91)$$

noting that if $V_{outa-} = V_{outh-}$ and $V_{outa+} = V_{outh+}$, the matching of the capacitors in Fig. 34.55 is perfect and $V_{avg+} - V_{avg-} = V_{outa+} - V_{outa-}$ (the output perfectly follows Eq. [34.61]). Taking the difference of these two equations

$$V_{avg+} - V_{avg-} = V_{outa+} - V_{outa-} - \underbrace{\frac{(V_{outa+} - V_{outh+}) - (V_{outa-} - V_{outh-})}{2}}_{\text{Error adjustment term}} \quad (34.92)$$

If we substitute Eqs. (34.76), (34.77), and (34.83) into this equation, we get

$$\begin{aligned}
 V_{avg+} - V_{avg-} = & 2(V_{in+} - V_{in-}) - (V_{Cl+} - V_{Cl-}) + (V_{in+} - V_{Cl+}) \cdot \frac{\Delta C_+}{C_+} - (V_{in-} - V_{Cl-}) \cdot \frac{\Delta C_-}{C_-} - \\
 & \frac{2(V_{in+} - V_{Cl+})}{C_F/C_I} \cdot \frac{\Delta C_+}{C_+} + \frac{2(V_{in-} - V_{Cl-})}{C_F/C_I} \cdot \frac{\Delta C_-}{C_-} \quad (34.93)
 \end{aligned}$$

which reduces to Eq. (34.85) or (34.61) assuming C_I/C_F is 1/2 (C_F is twice as large as C_I). Note how the selection of (and matching) of the capacitor ratios, C_F/C_I , is not that important. The capacitors do not have to match to the final ADC resolution. A matching of 1% will result in an error term that is one-hundredth of the error that would be present if the error averaging were not used. Also note, as indicated at the beginning of the section, that only the first stages (say the first five in a 14-bit ADC) need use capacitor error averaging because of the reduced accuracy requirements placed on the later stages as we move through the converter.

The penalty for this precision technique is the increase in conversion time used in each stage (and increased noise because two op-amps are used in each 1.5-bit section). As seen in Fig. 34.55, an extra half-clock cycle is required. For 20 Msamples/s a 30 MHz clock would be required. Again while one stage is in the hold (average) state, the next stage in the pipeline can be in the sample state so that the conversion time is shared (but still requiring 1.5 clock cycles). Using the capacitor error averaging technique for precision gain (and subtraction) and using 1.5-bit/stage sections a 14-bit ADC, using fully-differential input signals, has been attained at 20 Msamples/s without trimming or calibration [13].

Example 34.18

Simulate the operation of the circuit shown in Fig. 34.57. Comment on the ideal outputs and the simulation results.

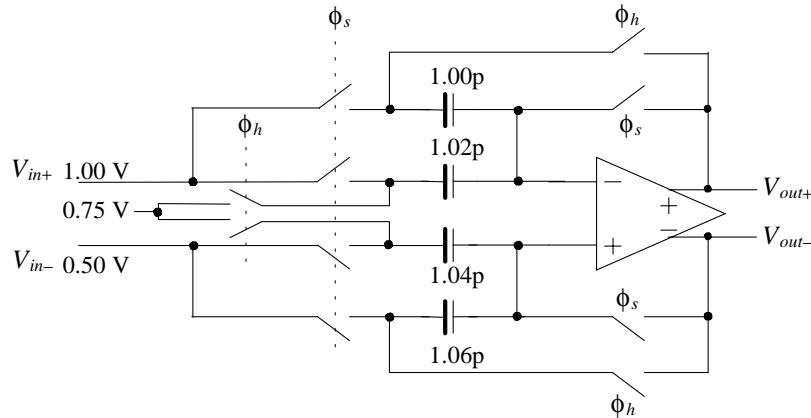


Figure 34.57 Matching errors in capacitors (see Ex. 34.18).

The capacitor values were chosen arbitrarily. The input voltage, V_{in} , is 1 – 0.5 or 0.5 V. The ideal output voltage, V_{out} , is then 1 V. V_{out+} should ideally be 1.25 V, and V_{out-} should ideally be 0.25 V. The simulation results are shown in Fig. 34.58. The error plotted in this figure is the result of taking the difference between the ideal output and the actual output. The error for the capacitor values shown in Fig. 34.57 is approximately 5 mV. Clearly, at the risk of stating the obvious, capacitor mismatch is a fundamental limitation to ADC accuracy. ■

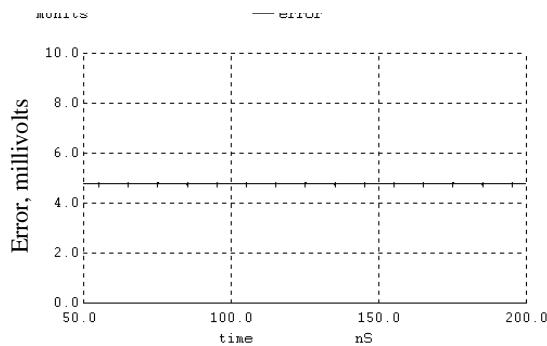


Figure 34.58 Simulation results from Ex. 34.18 showing amplification error.

Example 34.19

Simulate the operation of the circuit shown in Fig. 34.59 (a cascade of Figs. 34.55 and 34.56). Comment on the ideal outputs and the simulation results.

This circuit shows the same mismatched capacitors in the multiply-by-two stage as we saw in Ex. 34.18. The averaging stage also shows mismatched capacitors with

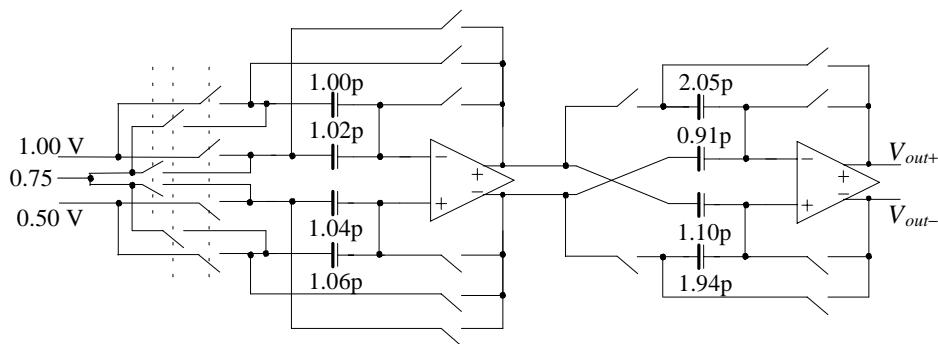


Figure 34.59 Implementation of error averaging (see Ex. 34.19).

arbitrary values. Again, as in Ex. 34.18, the ideal output voltage is 1.0 V ($V_{out+} = 1.25$ V and $V_{out-} = 0.25$ V). The simulation results are shown in Fig. 34.60. The error has dropped from 5 mV to $-70 \mu\text{V}$. It may be instructive to resimulate the capacitor error averaging topology of Fig. 34.59 using different values of capacitors in order to get a feeling for just how forgiving the topology is to mismatches.

Note that in the simulation netlist, where we are using near ideal op-amps with open loop gains of 100 million, we added switches in series with the C_i capacitors in the averaging circuit to avoid the situation of the op-amp operating open-loop with its outputs going to millions of volts when both ϕ_a and ϕ_h are low. (The op-amp operates open loop when ϕ_s is high, which is usually not a problem in a practical circuit). ■

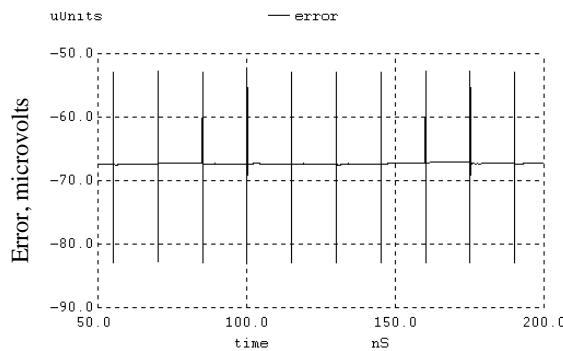


Figure 34.60 Simulation results from Ex. 34.19 showing amplification error.

Example 34.20

Repeat Ex. 34.19 if the op-amps used have open-loop gains of 1,000.

The simulation results are shown in Fig. 34.61. The error has increased by a factor greater than 10. As indicated earlier in Eq. (34.39), the minimum op-amp open-loop gain is set by the resolution of the data converter. If we were designing a 14-bit ADC, the op-amp used would require open-loop gains greater than 2^{16} or 64k (96 dB). ■

Comparator Placement

The implementation of a pipeline ADC showing how the clock signals are used in each error averaging stage is shown in Fig. 34.62. The important thing to note is the use of only three phases of an input clock signal. The connection of the clock phases to the switches changes in each stage allowing a stage to sample an input signal while the previous stage is in the hold mode. We'll discuss the generation of these clock signals in the next section; here we discuss comparator placement and performance.

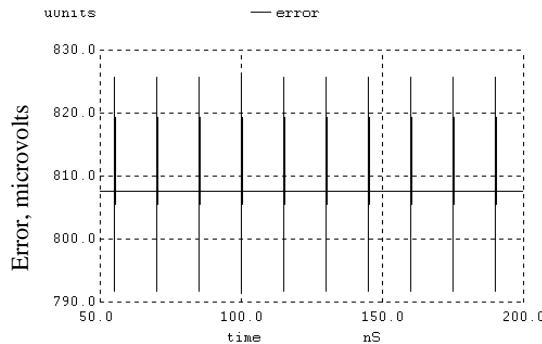


Figure 34.61 Regeneration of Fig. 34.60 using op-amps with open-loop gains of 1,000.

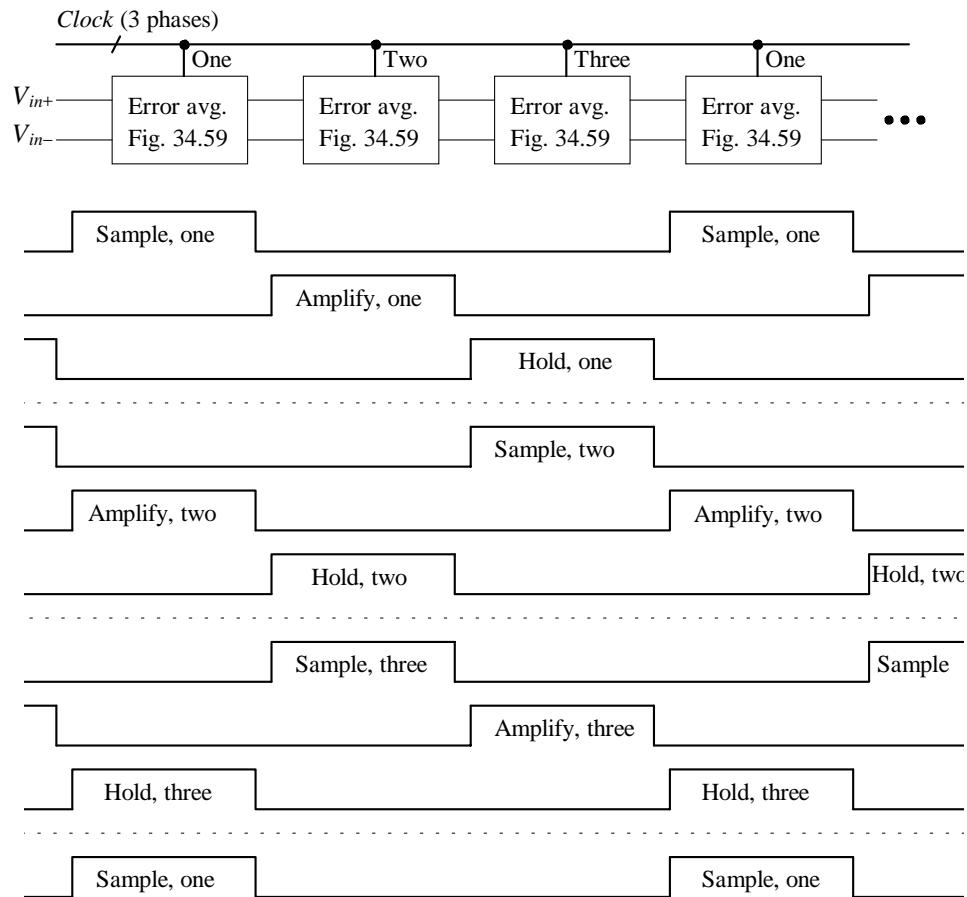


Figure 34.62 Implementation of a pipeline ADC using error averaging (Fig. 34.59).

As seen in Fig. 34.55 the voltages V_{Cl+} and V_{Cl-} must be valid and stable when the amplify-and-hold clock phases are high. Reviewing Fig. 34.51 we can implement the two comparators using a clock signal followed by a latch, Fig. 34.63. The comparators can be clocked on the rising edge of the amplifying clock (which is a different clock phase in each of the three possible clocking schemes). It's important to place a separate clocked latch on the outputs of each comparator (*clocked with a slightly delayed clock signal which isn't shown in the figure*) to ensure that comparator metastability isn't a factor when generating the MUX addressing (select inputs). We can get away with this type of clocking scheme because we are using three levels/stage (1.5 bits/stage). The comparators don't have to be N-bit accurate, as discussed earlier, but can withstand offsets/errors approaching $V_{CM}/4$. The signals V_{Cl+} and V_{Cl-} do have to be N-bit accurate, though, and must settle and stay settled to this accuracy by the end of the amplify phase.

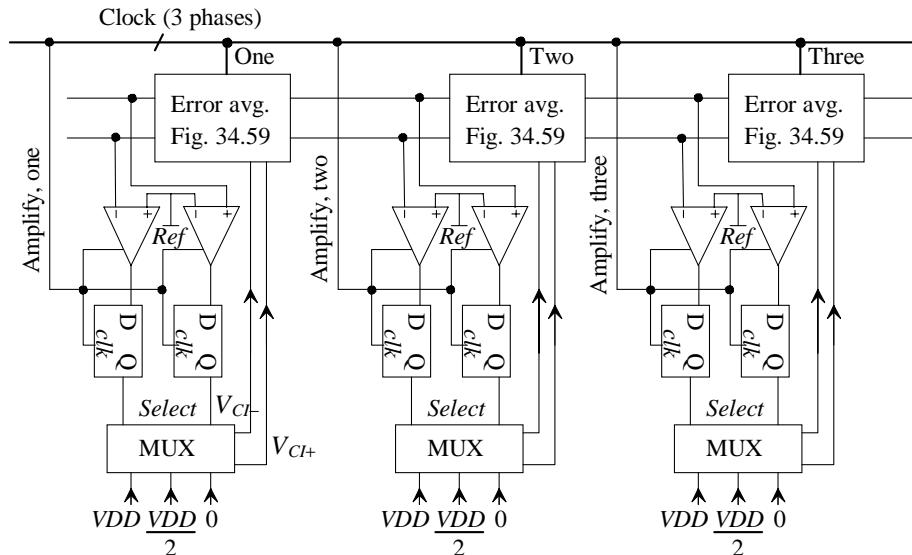


Figure 34.63 Placement of comparators in a pipeline ADC.

The two bits coming out of the comparators (actually the latches connected to the comparator outputs) can be thought of as the first delay element shown in Fig. 34.49 (except now it is a 2-bit delay element). Because each of these first delays are clocked on the rising edge of one phase of the clock signal, we have a problem with synchronizing the bits together prior to application to the digital correction logic (similar to Fig. 34.53). Reviewing the clock signals in Fig. 34.62, we see that if we clock all other delay elements in the pipeline of Fig. 34.49 on the rising edge of "amplify, one" we can synchronize all of the digital data together.

Clock Generation

Generating the nonoverlapping clock signals used in, for example, Fig. 34.55 can be accomplished by using the basic circuit shown in Fig. 34.64. It might be helpful to review Fig. 33.32 before we discuss the operation of this circuit. To understand the operation of this circuit (Fig. 34.64), note that when one of the input signals goes low the corresponding output phase goes high. The feedback ensures that the output doesn't go high until the previous phase goes back low (both inputs of the NOR gate must be low before its output goes high). The amount of nonoverlap time, again, is set by the delay in series with the output of the NOR gates.

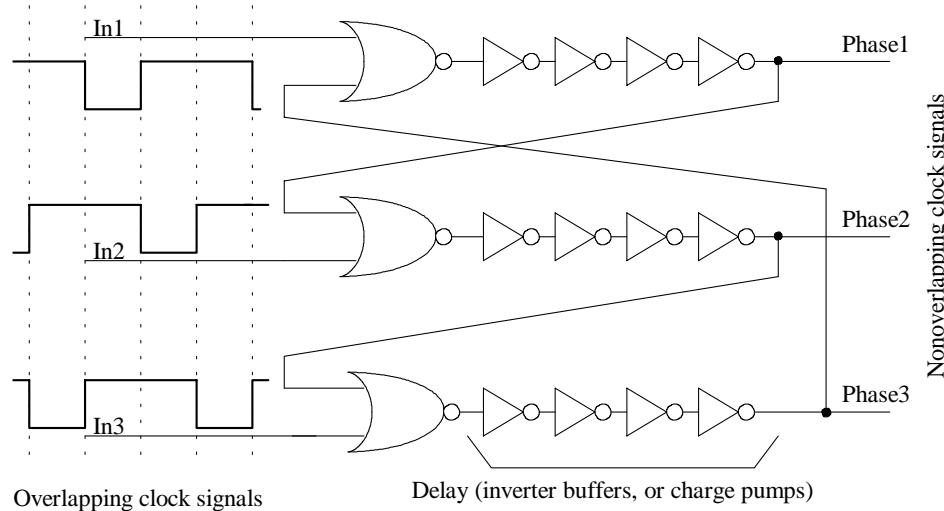


Figure 34.64 Circuit used for generating three phases of a nonoverlapping clock.

The input signals (overlapping clock signals) are generated with the circuit shown in Fig. 34.65. The outputs of this circuit change states on both the rising and the falling edges of the input clock signal. The latches are clocked on both the rising and falling edges of the input clock. They can be implemented with a parallel connection of the dynamic latches given in the last chapter (one latch clocked on the rising edge and one clocked on the falling edge with both latches sharing the common output MOSFETs). If the input clock signal is 25 MHz with a period of 40 ns, then the width of each of the output nonoverlapping clock phases in Fig. 34.64 is less than (but approximately) 20 ns. Note that there are other ways to generate the input clock signals for the circuit of Fig. 34.64. (A shift register with one bit set low is one example.) The method shown starts with both latches' outputs set low, 00 ($= Q_1Q_2$). On the rising edge of the clock signal the output changes to 01. On the following falling edge the output changes to 11. On the next rising edge the output changes to 01 and the sequence repeats itself.

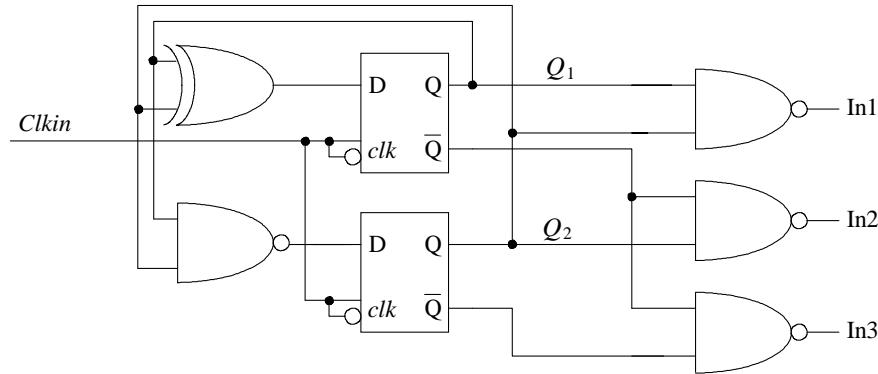


Figure 34.65 Generating the overlapping input clock signals for Fig. 34.64.

Offsets and Alternative Design Topologies

When discussing offsets, we've concentrated on the op-amp's offset. The offset associated with the common-mode voltage hasn't been discussed in detail. What happens if the output signals are moving around a voltage of $V_{CM} + V_{os}$? Taking the difference in the output signals will eliminate both the common-mode voltage and the offset. A problem could occur if the CMFB circuit doesn't affect each output the same. (The result is a difference-mode signal.)

Example 34.21

Simulate the operation of the circuit shown in Fig. 34.66. Show the input and output signals as the difference between the two differential input signals. Assume the common-mode voltage coming out of the op-amp is precisely 0.75 V.

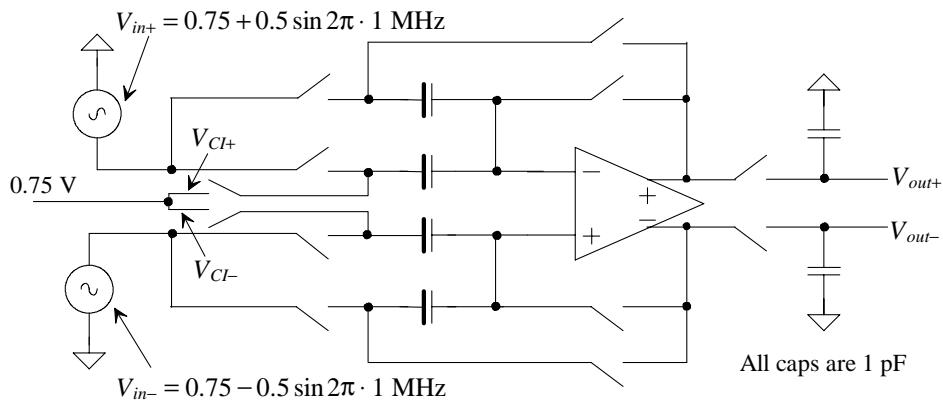


Figure 34.66 ADC building block discussed in Ex. 34.21.

The simulation results are shown in Fig. 34.67. Note how, as we would expect, the gain of the circuit is two. The signals shown are ideal. ■

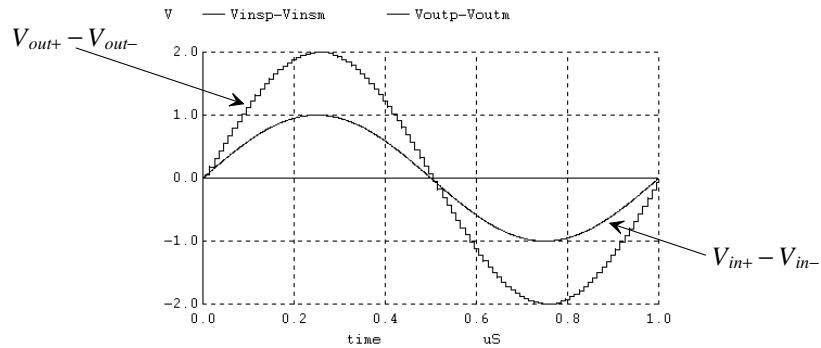


Figure 34.67 Input and output for the circuit of Fig. 34.66.

Example 34.22

Repeat Ex. 34.21 if the common-mode voltage coming out of the op-amp sees an offset of 100 mV, i.e., it is 850 mV.

The differential input and output signals with this output common-mode offset look exactly like what is seen in Fig. 34.67. The single-ended output signals, Fig. 34.68, show the offset (the signals swing around 850 mV in Fig. 34.68). Clearly a common-mode offset in the op-amp output signals isn't a concern (unless it's so large that it limits the op-amp output swing range). Likewise an offset in the common-mode voltage of the input signals isn't a concern (this comment is easy to verify with simulations). What is a concern, though, is the value of the voltages used for V_{Cl+} and V_{Cl-} . ■

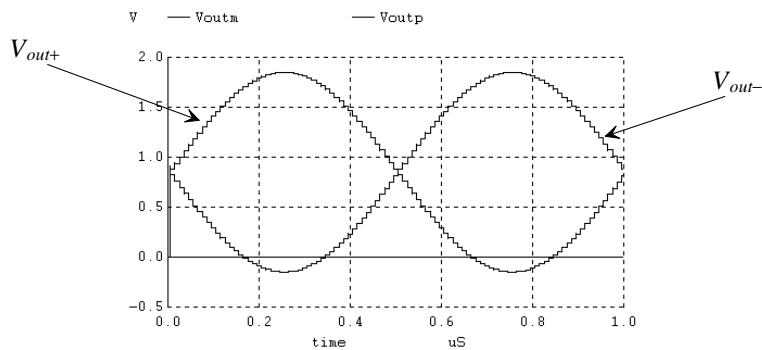


Figure 34.68 Output signals for the circuit of Fig. 34.66 if common-mode voltage is 0.85 V.

In Fig. 34.66 V_{Cl+} and V_{Cl-} are shorted together and connected, through a switch, to 0.75 V (V_{CM}). It can be shown that when this occurs, the absolute value of the voltage is irrelevant because it is common to both input signal paths, see Eq. (34.53). This means that if we used ground instead of V_{CM} in Fig. 34.66, we would get the same outputs. A problem does occur if the difference in the voltages (when adding or subtracting a voltage from the input signal) isn't exactly what is desired, see Eq. (34.61).

While an offset in the common-mode voltage isn't important, the op-amp's offset is a concern. The op-amp offset voltage is zeroed out when using the topology shown in Fig. 34.66. Equation (34.50) was derived assuming the op-amp had a nonzero offset voltage and shows the offset will not (ideally) affect the building block's output signals. To show the removal of the offset using simulations, consider the following example.

Example 34.23

Simulate the operation of the circuit shown in Fig. 34.69. This figure is Fig. 34.66 redrawn with a 100 mV op-amp offset.

The simulation results are shown in Fig. 34.70. Figure 34.70 should be compared to Fig. 34.67. Note how the unrealistically large offset has no effect on the building block's output signals. ■

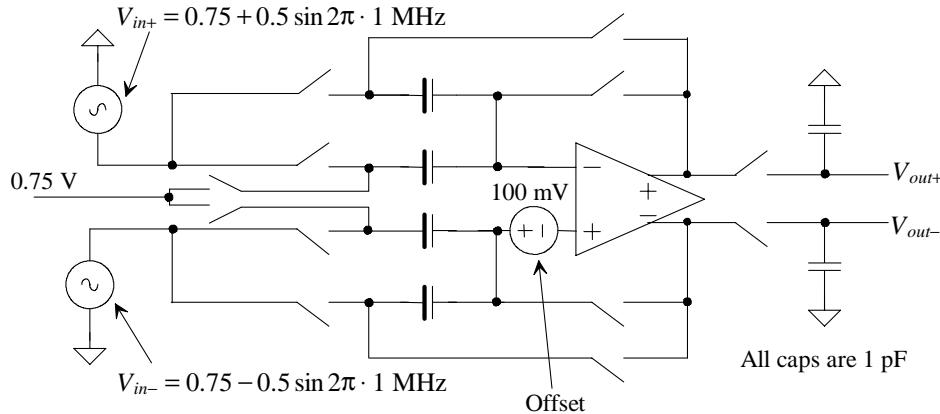


Figure 34.69 ADC building block discussed in Ex. 34.23.

If the op-amp offset is zeroed out and doesn't affect the circuit's outputs when using the basic topology shown in Fig. 34.30, why would we want to consider some other topology? The answer to this question comes from the realization that the op-amp's outputs must settle to the final accuracy of the ADC, referring to Fig. 34.55, by the falling edge of all three phases of the clock. It would be nice to make the settling during one of these three (or two) phases irrelevant. Also, and perhaps more importantly, it would be nice to use other types of CMFB (discussed in the next section).

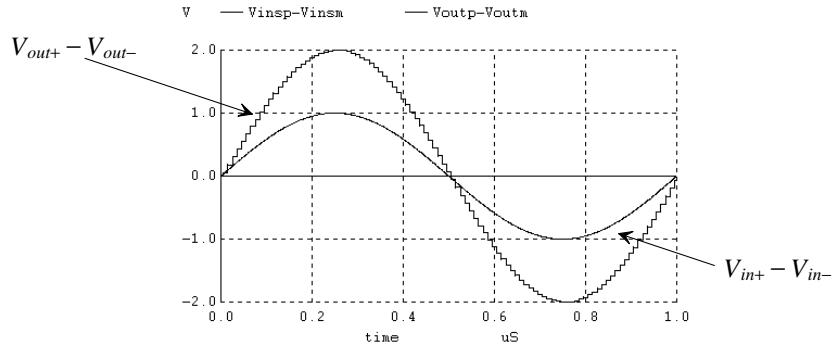


Figure 34.70 Input and output for the circuit of Fig. 34.69 with op-amp offset of 100 mV.

Consider the building-block topology shown in Fig. 34.71. During the sampling phase, the inputs of the op-amp are shorted to the common-mode voltage. During this time the op-amp operates open loop, and settling time is meaningless. The status of the op-amp outputs during this time is discussed in the next section. The charge stored on the capacitors during the storage phase is

$$Q_{I,F}^{\phi_s} = C_{I,F} \cdot (V_{in} - V_{CM}) \quad (34.94)$$

while during the hold phase

$$Q_I^{\phi_h} = C_I \cdot (V_{CI} - V_{CM} \pm V_{OS}) \text{ and } Q_F^{\phi_h} = C_F \cdot (V_{out} - V_{CM} \pm V_{OS}) \quad (34.95)$$

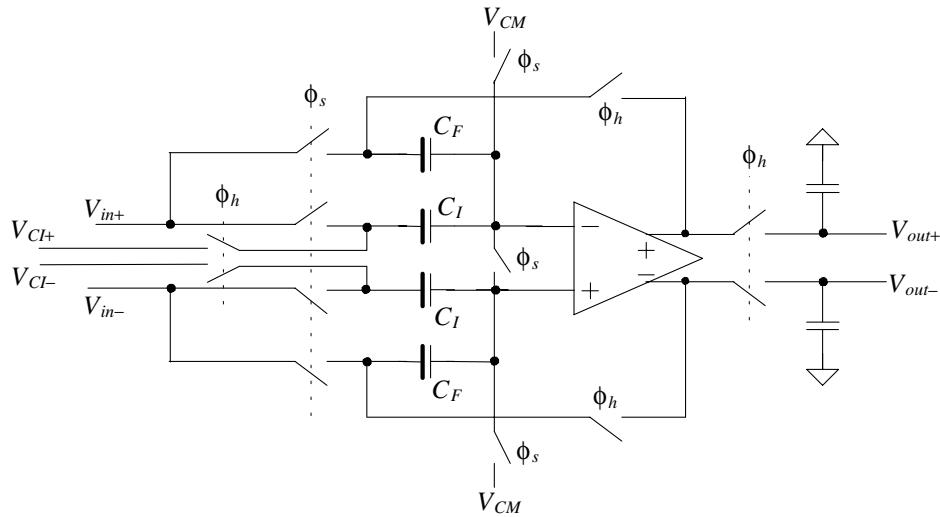


Figure 34.71 Alternative ADC building block.

Once again knowing charge must be conserved, the output voltage can be written as

$$V_{out} = V_{out+} - V_{out-} = \left(1 + \frac{C_I}{C_F}\right) \cdot (V_{in+} - V_{in-}) - \frac{C_I}{C_F} \cdot (V_{CI+} - V_{CI-}) \pm \underbrace{\left(1 + \frac{C_I}{C_F}\right) \cdot V_{os}}_{\text{unwanted term}} \quad (34.96)$$

This equation should be compared to Eq. (34.53) where the offset was zeroed out during the sample phase of the clock (the phases ϕ_1 and ϕ_2). The topology would be useful in an ADC where we can guarantee that the offset contribution is well below the data converter's LSB or in a later stage in the ADC where the accuracy requirements are relaxed.

Example 34.24

Repeat Ex. 34.23 using the topology shown in Fig. 34.71.

The simulation results are shown in Fig. 34.72. The ratio of C_I to C_F is one and so, ideally, the output is just twice as large as the input. The output should swing from 2 V down to -2 V. However, because of the 100 mV offset, which is also multiplied by a factor of 2, the output is shifted downwards so that it swings from 1.8 V to -2.2 V. ■

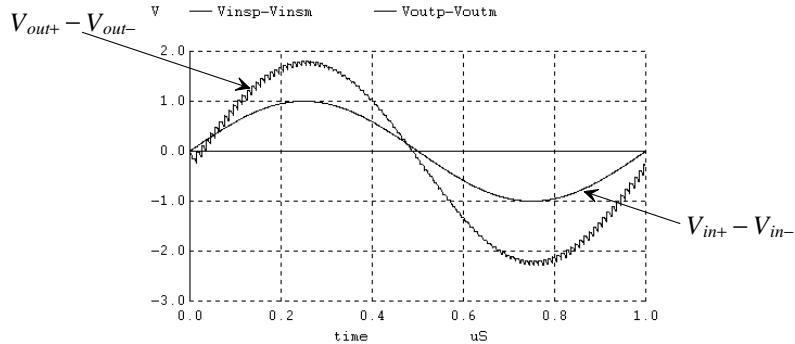


Figure 34.72 Input and output for the circuit of Fig. 34.71 with op-amp offset of 100 mV.

Dynamic CMFB

The CMFB circuits discussed earlier employ an amplifier to sense the average of the outputs and feedback a correction to center the signals around V_{CM} . In Ch. 27 we discussed a CMFB technique used in an op-amp that was dynamic and doesn't employ an amplifier. A simplification of this dynamic CMFB can be realized by noting that during the sampling phase in Fig. 34.71 the op-amp inputs are forced to V_{CM} . The scheme we are about to present won't force the inputs to $V_{CM} \pm V_{os}$ during the sample phase as in the other topologies based on Fig. 34.30.

The basic dynamic CMFB circuit is shown in Fig. 34.73. During the sample phase of the clock the inputs and outputs of the op-amp are shorted to the common-mode voltage. Also during this time the common-mode feedback voltage, V_{CMFB} , is set to a bias voltage (V_{bias4} if the op-amp of Fig. 34.36 is used [see Fig. 33.60]). During the hold phase, the CMFB capacitors on the output of the circuit are disconnected from both V_{CM} and V_{bias4} and are used to sense the average value of the outputs. If the outputs move in a balanced fashion, then V_{CMFB} remains equal to V_{bias4} . If the average of the outputs moves upwards above V_{CM} , then V_{CMFB} increases, pulling the output common-mode voltage downwards. Again, because the CMFB loop utilizes negative feedback, an increase in V_{CMFB} must result in a decrease in $(v_{o+} + v_{o-})/2$.

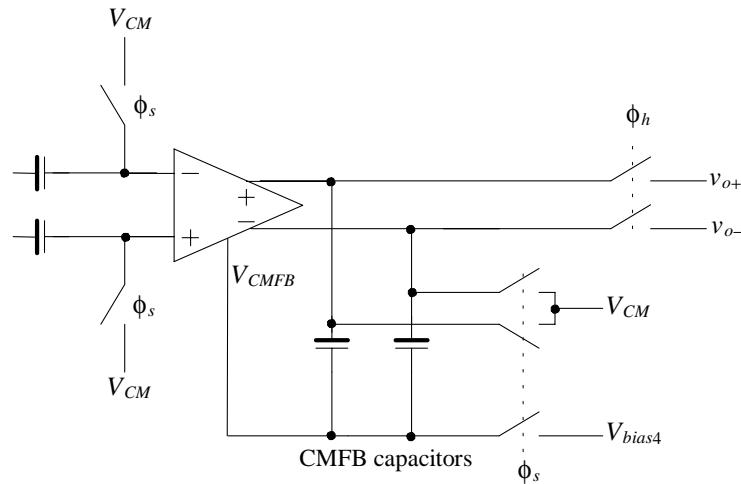


Figure 34.73 Dynamic CMFB.

Looking at Fig. 34.73 we see that because of the op-amp's offset voltage the outputs of the op-amp will source/sink a current into V_{CM} during the sample phase of the circuit's operation. By adding an extra pair of switches to disconnect the CMFB capacitors from the op-amp outputs we can avoid this situation. Adding the switches causes the op-amp outputs to approach the power supply rails during the sample phase (because of the offset voltage). This output rail isn't a problem if an OTA (single-stage) topology like the one in Fig. 34.36 is used. The outputs have no capacity to hold charge and so when the CMFB capacitors are reconnected to the op-amp outputs, the outputs immediately go to V_{CM} (neglecting the connection of the feedback capacitors which are not shown in Fig. 34.73). If a two-stage op-amp is used then care must be taken to ensure that the compensation capacitor (which, of course, does have the capacity to hold charge) doesn't charge to the rail resulting in a large recovery time when the op-amp circuit is put in the hold mode.

One solution to the problem of offset (when using the topology of Fig. 34.71) is to AC couple the output of the first op-amp stage to the input of the second op-amp stage, as in Fig. 34.74 (add a capacitor between the two stages to remove the DC component). The first stage's offset is stored on the capacitors during the sample phase, while the second stage's inputs are shorted to V_{bias1} . When the op-amp is used in the hold phase of operation, the second stage's offset is referred back to the input of the op-amp after dividing by the first stage's gain. For example, if the first stage of the op-amp has a gain of 50 and the second stage's offset is 10 mV, then the input-referred offset of the op-amp in the hold mode is only 0.2 mV. Note that the gain of the first stage can't be too large. If, for example, the first stage's gain were 1,000 and its offset were 10 mV, the outputs would saturate and the offset would not be stored on the capacitors. If the gain of the first stage were 50 with a 10 mV offset, then the (differential) outputs of the first stage would change by 500 mV. As long as the MOSFETs remain in the saturation region, the offset storage works as desired.

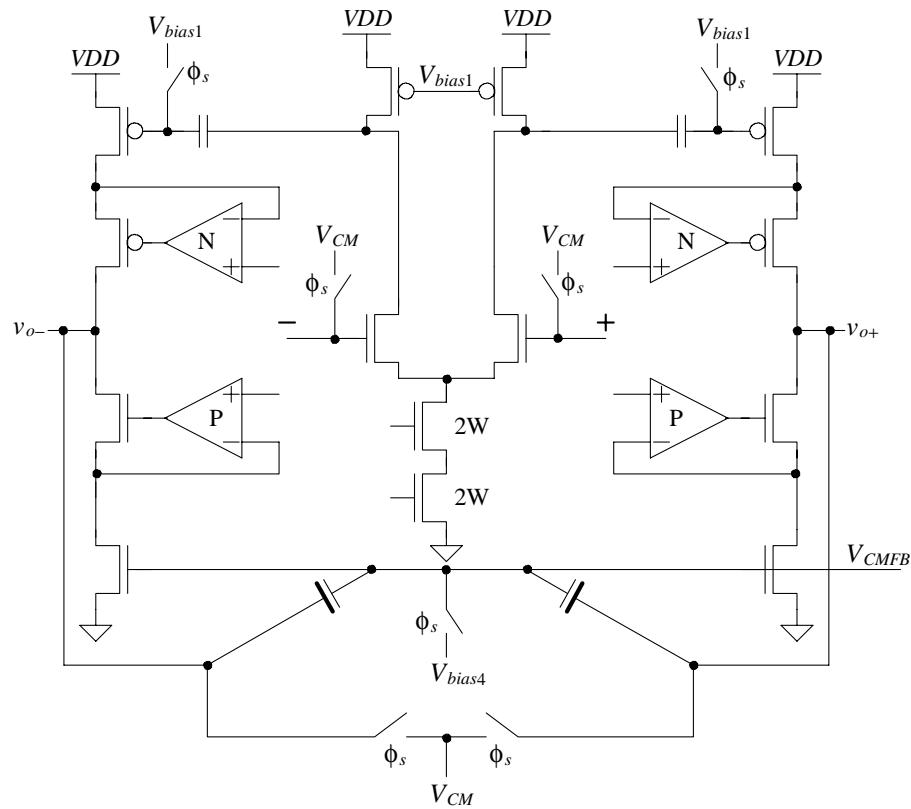


Figure 34.74 Implementation of dynamic CMFB.

Depending on the design parameters the op-amp shown in Fig. 34.74 would be compensated using either a load capacitance (as if the op-amp were a single stage) or using Miller compensation by adding a capacitor from the output of the op-amp to the output of the diff-amp (first stage) with a series resistance to cancel the right-hand plane zero (as discussed in Ch. 25).

Layout of Pipelined ADCs

Before leaving this chapter, let's comment on the layout of pipeline ADCs. Using a fixed-height layout structure, with the automatically routed power and ground busses (when the cells are laid end-to-end), is a good place to start when laying out the op-amp. As seen in Fig. 34.75, a height (with example values shown) is selected with the width variable. It's important, as discussed in Ch. 28, to keep the analog signals separate, both physically and by distance, from the digital signals.

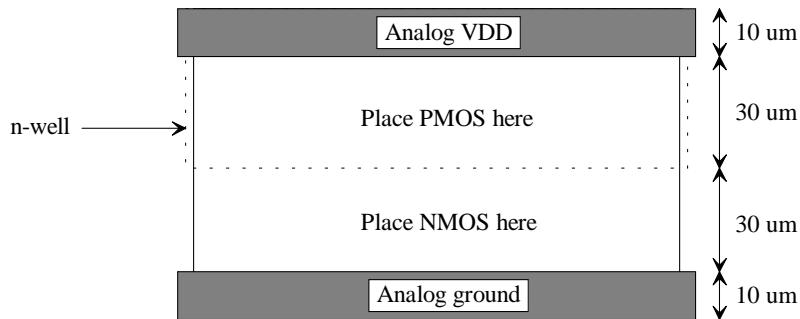


Figure 34.75 Fixed-height layout structure.

A possible block diagram of the placement of the fixed-height cells together with the capacitors and switches used to implement a stage of the ADC is seen in Fig. 34.76. Looking at the input signal, we notice that they are laid out next to each other and routed as close as possible to the input of the first stage of the ADC. All of the differential analog signals in the ADC should follow this practice to help make any coupled noise truly common-mode. Notice how we have placed ground pads adjacent to the input signals. These pads are not used for ground connections on-chip. The pads are used to help reduce noise coupling into the input signals. Ideally, these ground pins provide a termination for the noise keeping the input signals "clean." This is especially important when bonding wires connect the integrated circuit to a padframe in the final packaged part. The bonding wires used for digital signals tend to radiate more than enough signal to corrupt the input signal and ruin the ADC's SNR when placed close to an analog low-level signal.

Next notice that we must have a clock signal in our analog domain. This signal, as we have seen, is used for clocking the switches in the S/H stage. Although in the figure we show the placement of the clock adjacent to the input signals, it may be better to move the pad and, if possible, the routing of the clock signals away from the inputs. The main goal

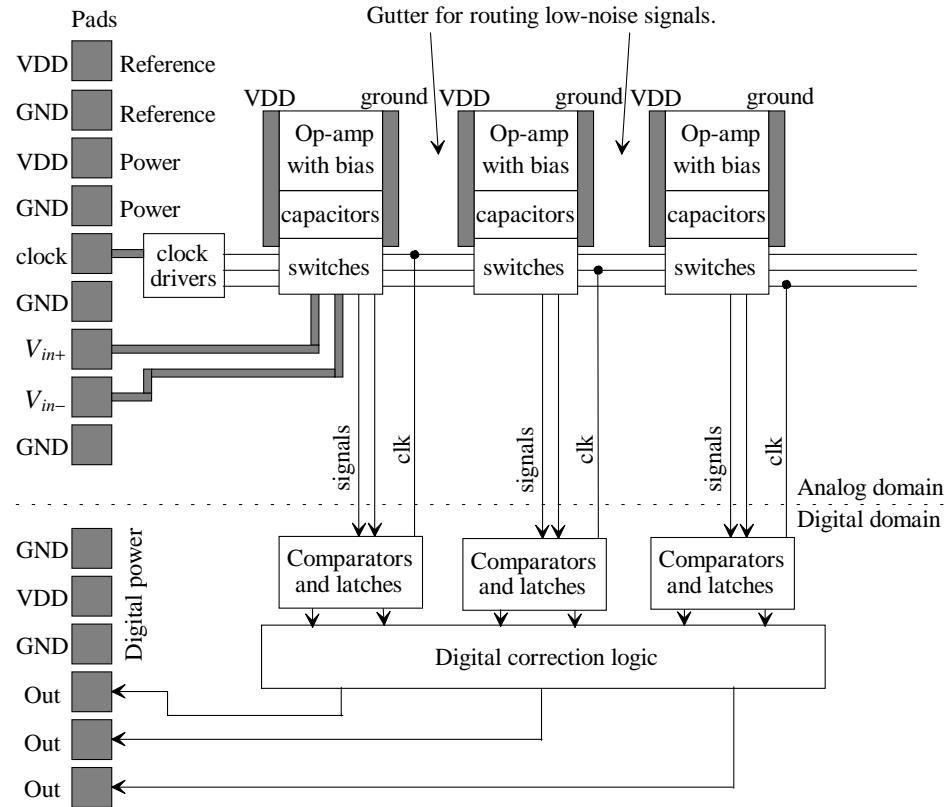


Figure 34.76 Block diagram layout of a pipeline stage.

when routing the clock signals is to keep the layout regular. Routing clock signals all over the layout is asking for problems.

Two sets of power and ground pads are used (more if possible) in the analog domain for the implementation of the ADC. One set of pads is used for supplying power to the op-amps while the other set is used for supplying the reference voltages to each stage. The power and ground supplies are common to both digital and analog sections off-chip. Off-chip the supplies are connected together and decoupled (bypassed) using large capacitors (actually a wide range of capacitor values are connected in parallel between the VDD and ground to avoid the increase in a single large capacitor's effective series resistance with frequency). On-chip decoupling capacitors can be used as well. The analog and digital power and ground connections are not shared, and so care must be taken not to decouple the analog VDD to digital ground or digital VDD to analog ground. Figure 34.77 shows one example of how the decoupling capacitors can be connected.

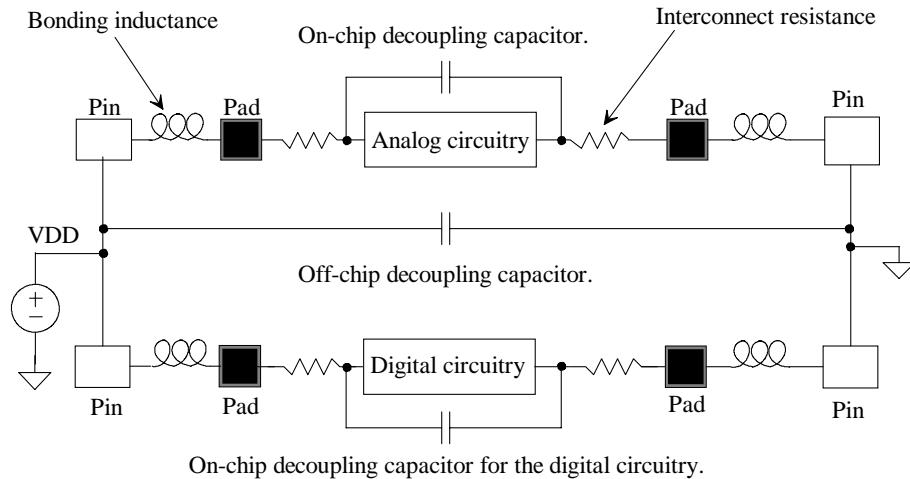


Figure 34.77 Showing how decoupling capacitors are used in a mixed-signal chip.

In general we don't want any DC current flowing on our reference voltages (those voltages used for V_{Cl} in our op-amp) because of the possible voltage drop along the supplying line. There may also be a voltage drop along the metal lines supplying power to the op-amps. However, small (DC) changes in these voltages are usually not a significant factor in the precision of the ADC. These changes could be a factor if the output signal approaches the power-supply voltages where the op-amp runs out of head room. AC changes in the power-supply voltages can be a significant factor limiting the ADC's performance.

Ground planes and wide conductors should be used where possible. Using power and ground planes not only provides good distribution of power and ground but also increases the capacitance between the supplies. Areas, labeled "gutter" in Fig. 34.76 should be provided for low-level analog signals. These areas are free to allow the quiet routing of the switch inputs and outputs to the op-amp outputs.

It's also a good idea to use guard rings around the sensitive analog circuits (e.g., the switched capacitors) to avoid coupled substrate noise. Figure 34.78 shows the basic idea. In this figure the capacitors are laid out over an n-well. The n-well is tied to analog VDD through an n+ implant in the n-well and metal. Surrounding the n-well is a ring of p+. This ring is tied to analog ground. The idea is that the p+ will provide a sink for any current injection from the surrounding circuitry. Because ground is the lowest potential in the circuit, the noise will terminate on this p+ and not penetrate the area under the capacitors (and then hopefully not couple into the capacitors). While this works well by itself, it may not be enough. Noise currents may still move deep in the substrate and work their way up under the capacitors. Because the n-well under the capacitor is held at the most positive potential in the circuit, any noise that does get into the n-well will hopefully be swept out through VDD and not couple into the capacitors.

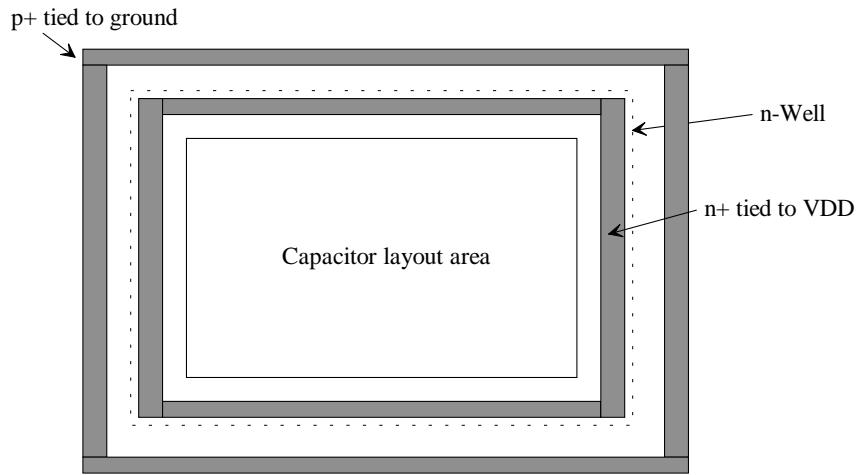


Figure 34.78 Using guard rings for protection in sensitive analog blocks.

Finally, if a sensitive analog signal does need to cross a digital signal (an example being the potential need to feed the switch inputs and outputs across the three phases of the switch clock signals in Fig. 34.76) a shield should be used, Fig. 34.79. In this figure the sensitive analog signal is assumed to exist on Metal1, while Metal2 is used for an analog ground shield from the digital signal on Metal3. This shield is used for isolation providing a terminating plane for the electric fields resulting from the voltages on both the digital and analog signals.

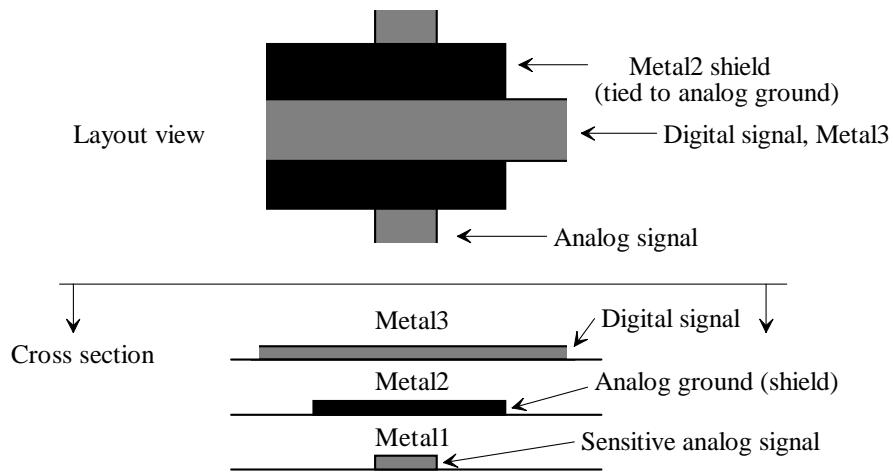


Figure 34.79 Shielding a sensitive analog signal from a digital signal.

REFERENCES

- [1] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*, Cambridge University Press, 1998.
- [2] R. E. Suarez, P. R. Gray, and D. A. Hodges, "All-MOS Charge Redistribution Analog-to-Digital Conversion Techniques - Part II," *IEEE Journal of Solid-State Circuits*, Vol. 10, No. 6, pp. 379-385, December 1975.
- [3] R. J. Baker, H. W. Li, and D. E. Boyce, *CMOS Circuit Design, Layout, and Simulation*, Wiley-IEEE, 1998.
- [4] S. Franco, *Design with Operational Amplifiers and Analog Integrated Circuits*, Second Edition, McGraw-Hill, 1998.
- [5] J. A. Schoeff, An Inherently Monotonic 12-bit DAC, *IEEE Journal of Solid-State Circuits*, Vol. SC-14, No. 6 December 1979, pp. 904-911.
- [6] C. G. Yu and R. L. Geiger, "An Automatic Offset Compensation Scheme with Ping-Pong Control for CMOS Operational Amplifiers," *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 5 May 1994, pp. 601-610.
- [7] C. C. Enz and G. C. Temes, "Circuit Techniques for Reducing the Effects of Op-Amp Imperfections: Autozeroing, Correlated Double Sampling, and Chopper Stabilization," *Proceedings of the IEEE*, Vol. 84, No. 11, pp. 1584-1614, November 1996.
- [8] I. E. Opris, L. D. Lewicki, and B. C. Wong, "A Single-Ended 12-bit 20 Msample/s Self-Calibrating Pipeline A/D Converter," *IEEE Journal of Solid-State Circuits*, Vol. 33, No. 12, December 1998.
- [9] E. Fong, private communication, May 2001.
- [10] B. Ginetti, P. G. A. Jespers, and A. Vandemeulebroecke, "A CMOS 13-b Cyclic RSD A/D Converter," *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 7, July 1992.
- [11] W. C. Black and D. A. Hodges, "Time Interleaved Converter Arrays," *IEEE Journal of Solid-State Circuits*, Vol. SC-15, No. 6, December 1980.
- [12] B. S. Song, M. F. Tompsett, and K. R. Lakshmikumar, "A 12-bit, 1-MSample/s Capacitor Error-Averaging Pipelined A/D Converter," *IEEE Journal of Solid State Circuits*, Vol. 23, No. 6, pp. 1324-1333, December 1988.
- [13] H. S. Chen, B. S. Song, and K. Bacrania, "A 14-b 20-Msample/s CMOS Pipelined ADC," *IEEE Journal of Solid State Circuits*, Vol. 36, No. 6, pp. 997-1001, June 2001.

QUESTIONS

- 34.1** Assuming the DAC shown in Fig. 34.1 is 8 bits and $V_{REF+} = 1.5$ V and $V_{REF-} = 0$, what are the voltages on each of the $R-2R$ taps?

- 34.2** Give an example of how the traditional current-mode DAC will have limited output swing.
- 34.3** Repeat question 34.1 for the DAC shown in Fig. 34.2.
- 34.4** For the wide-swing current mode DAC shown in Fig. 34.3, what are the voltages at the taps along the $R-2R$ string assuming 8 bits, $V_{REF+} = 1.5$ V, $V_{REF-} = 0$, and a digital input code of 0000 0000?
- 34.5** Can the op-amp shown in Fig. 34.36 be used in fully-differential implementations of the DACs shown in Figs. 34.1 - 34.3? Why or why not?
- 34.6** Show the detailed derivation of Eqs. (34.12)-(34.14).
- 34.7** Why would we want to use both current segments and binary-weighted currents to implement a current-mode DAC? (Why use segmentation?)
- 34.8** Why do we subtract ΔA in Eq. (34.36)? Why not add the gain variation?
- 34.9** Does the matching of the capacitors matter in the S/H of Fig. 34.31? Why or why not?
- 34.10** Neglecting offsets and assuming the ϕ_1 switches are closed in Fig. 34.34, can the two inverting terminals of the error amplifier (the input terminals of the fully-differential op-amp) be at different potentials?
- 34.11** When the ϕ_3 switches are closed in Fig. 34.34, is it possible for the inverting inputs of the error amplifier to be at different potentials? Again, neglect offsets.
- 34.12** Repeat Ex. 34.10 if the cyclic ADC's input is 0.41 V.
- 34.13** Is kick-back noise from the comparator a concern for the circuit of Fig. 34.39?
- 34.14** Derive the transfer function for the circuit shown in Fig. 34.80.

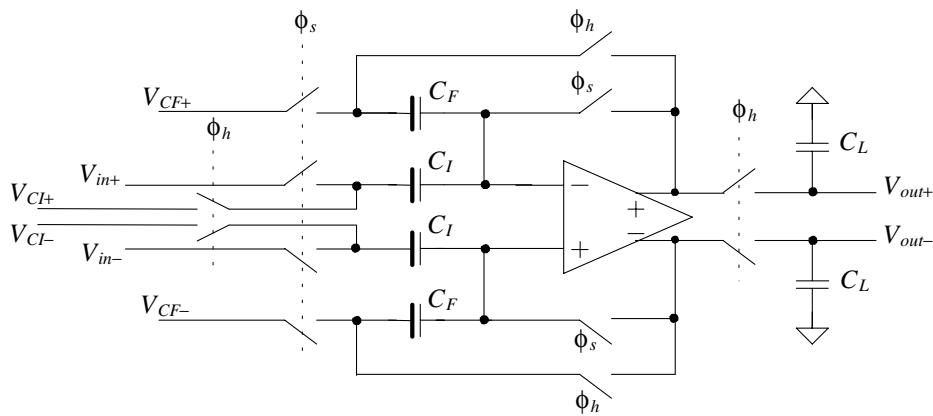


Figure 34.80 Circuit used in question 34.14.

- 34.15** Repeat Ex. 34.16 if the input voltage is 0.41 V.
- 34.16** Repeat Ex. 34.17 if the input voltage is 0.41 V.
- 34.17** Resketch the clock waveforms for Fig. 34.54 if bottom plate sampling is used.
- 34.18** Show the derivation leading up to Eq. (34.83). Show, using practical values for mismatch, how the squared mismatch terms are negligible.
- 34.19** What happens to the error adjustment term in Eq. (34.92) if the capacitors in the S/H are perfectly matched?
- 34.20** Repeat Ex. 34.18 if all capacitors are 1 pF (the ideal situation) and verify that the error out of the stage is zero.
- 34.21** Sketch a circuit to provide the inputs for the four-phase, nonoverlapping clock generator shown in Fig. 34.81.

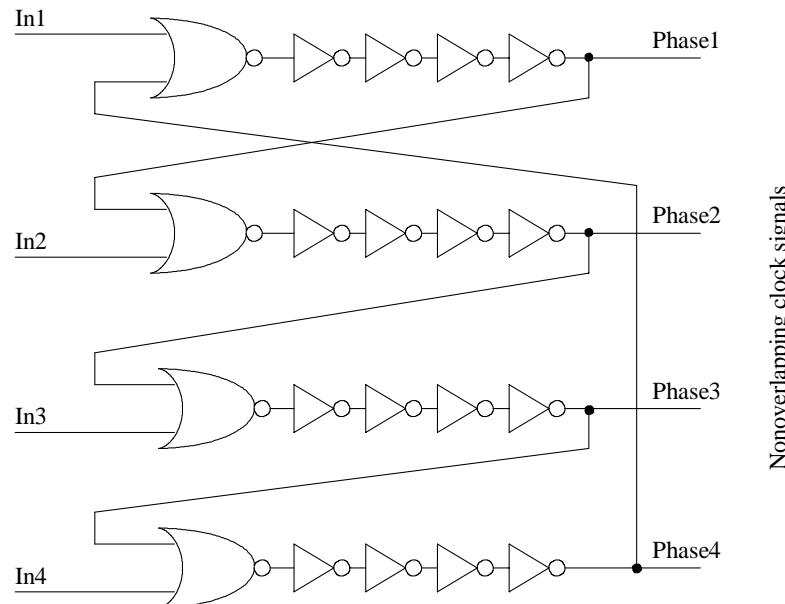


Figure 34.81 Four-phase, nonoverlapping clock generator.

- 34.22** What is the main advantage of using dynamic CMFB over other CMFB circuits? What is the main disadvantage?

- 34.23** Can MOSFETs be used to implement the on-chip decoupling capacitors in Fig. 34.77?
- 34.24** Sketch the cross-sectional view of the layout in Fig. 34.78.
- 34.25** Figure 34.82 shows the implementation of a pipeline DAC. How would we implement this DAC using a topology similar to Fig. 34.42? Sketch the DAC's implementation and the timing signals (clock phases) used.

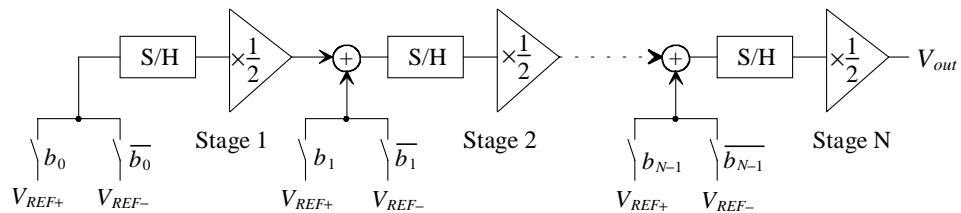


Figure 34.82 A pipeline digital-to-analog converter.

Chapter

35

Integrator-Based CMOS Filters

We've covered the detailed design of analog interfaces used for analog-to-digital and digital-to-analog conversion in the past five chapters. While we can perform signal processing and filtering in the digital domain, as seen in Fig. 30.2, analog antialiasing and reconstruction filters are still required in our system. Analog continuous-time filters (a simple example being the RC circuit shown in Fig. 30.9) can be faster (have wider bandwidths) and take up less area than their discrete-time counterparts. However, unlike discrete-time filters, continuous-time filters cannot be fabricated with precise transfer functions and must be tuned. This is especially true if passive resistors and capacitors are used where each can have a variation of $\pm 20\%$. By using active CMOS integrators in the filter implementations instead of passive elements, we can electrically tune the filters. Also, we can more easily implement higher order filters while minimizing the effects of loading.

In this chapter we discuss analog filters (both continuous- and discrete-time filters where the input and output are analog) made using continuous-time analog integrators (CAIs or active-RC integrators), MOSFET-C integrators, transconductor-capacitor (g_m -C) integrators, and discrete-time analog integrators (DAIs). We also discuss digital filter design (which, of course, is also a discrete-time filter) based on the topologies developed using the digital integrator (e.g., see Fig. 31.34) where the input and output are digital.

35.1 Integrator Building Blocks

35.1.1 Lowpass Filters

To methodically develop our understanding of CMOS filters, consider the lowpass filter shown in Fig. 35.1. The transfer function of this filter (see also Ex. 30.3) is

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{1}{1 + j\omega RC} \quad (35.1)$$

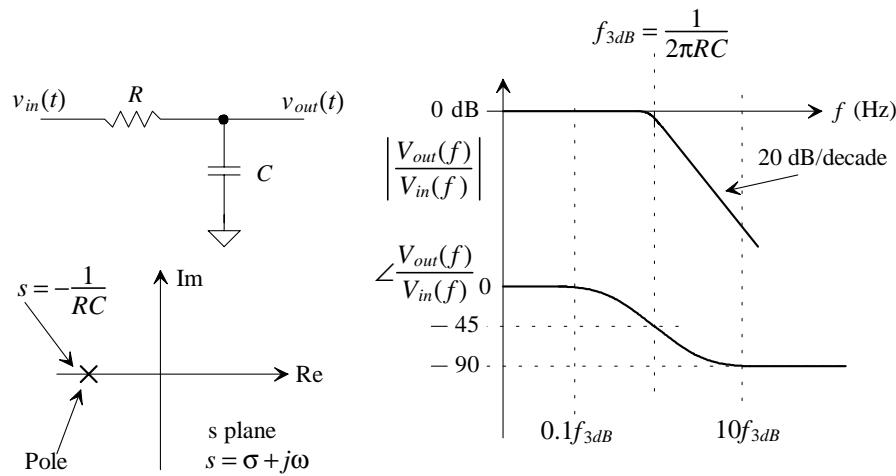


Figure 35.1 First-order lowpass filter.

where $\omega = 2\pi \cdot f$ and f is the frequency of the input (and thus the output). Next consider the block diagram in Fig. 35.2. This figure shows an integrator and a summing block (which by now we know can be implemented with a single op-amp). The output of the block diagram can be determined by solving

$$V_{out}(f) = \frac{G}{s} \cdot (V_{in}(f) - V_{out}(f)) \quad (35.2)$$

or

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{1}{1 + s/G} \quad (35.3)$$

where for a sinewave input $s = j\omega$. Comparing this equation to Eq. (35.1), we see that if we set the integrator's gain, G , using

$$G = \frac{1}{RC} \text{ where } f_{3dB} = \frac{G}{2\pi} \quad (35.4)$$

we can use an integrator to implement a lowpass first-order filter (the filter has a single pole).

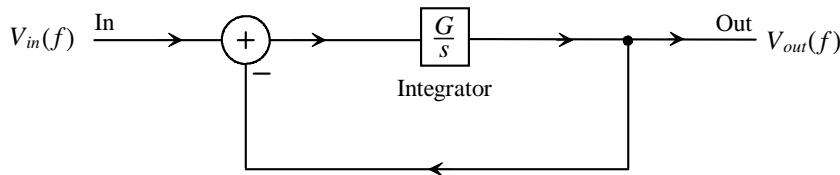


Figure 35.2 Block diagram of an integrator-based lowpass filter.

35.1.2 Active-RC Integrators

We showed a continuous-time analog integrator (CAI) in Fig. 32.24. The fully-differential version of this integrator is shown in Fig. 35.3. The CAI goes by other names, including the Miller integrator, the active-RC integrator, and when the resistors are replaced with MOSFETs operating in the triode region, the MOSFET-C integrators. The gain of the CAI can be written as

$$\frac{V_{out}}{V_{in}} = \frac{V_{out+} - V_{out-}}{V_{in+} - V_{in-}} = \frac{1}{s} \cdot \underbrace{\frac{1}{RC}}_G \quad (35.5)$$

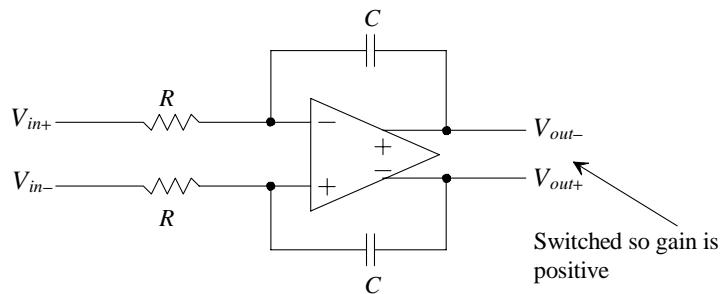


Figure 35.3 A continuous-time analog integrator (CAI).

Reviewing Fig. 35.2, we see that the CAI of Fig. 35.3 alone will implement the needed integration but not the summing (difference) block. By adding an additional feedback path, as we did in Fig. 32.24, the entire block diagram of Fig. 35.2 can be implemented. Figure 35.4 shows the integrator-based implementation of the circuits in

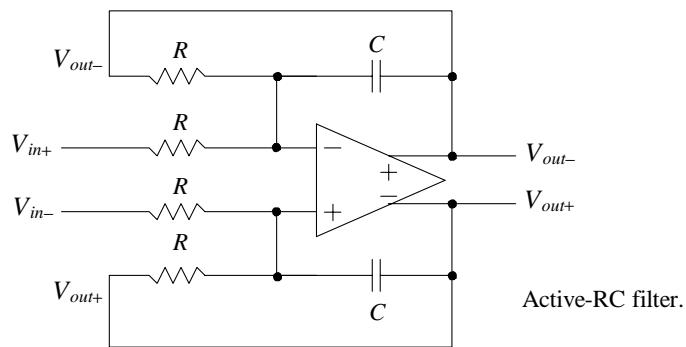


Figure 35.4 Implementation of a first-order lowpass filter using a CAI.

Figs. 35.1 and 35.2 (noting the op-amp must be able to drive a resistive load). This filter is called an *active RC* filter because the RC is used with an active element (the op-amp). At this point there are several practical and useful modifications that we can make to this filter. However, let's work an example before moving on.

Example 35.1

Simulate the operation of the filter in Fig. 35.4 from DC to 100 MHz if $R = 10\text{k}$ and $C = 10 \text{ pF}$. Show both the magnitude and phase responses of the filter. Assume the op-amp is ideal.

From Fig. 35.1 we know the 3 dB frequency of the filter is 1.59 MHz. The simulation results are shown in Fig. 35.5. The magnitude and phase response follow, as expected, the responses for the simple RC filter shown in Fig. 35.1. ■

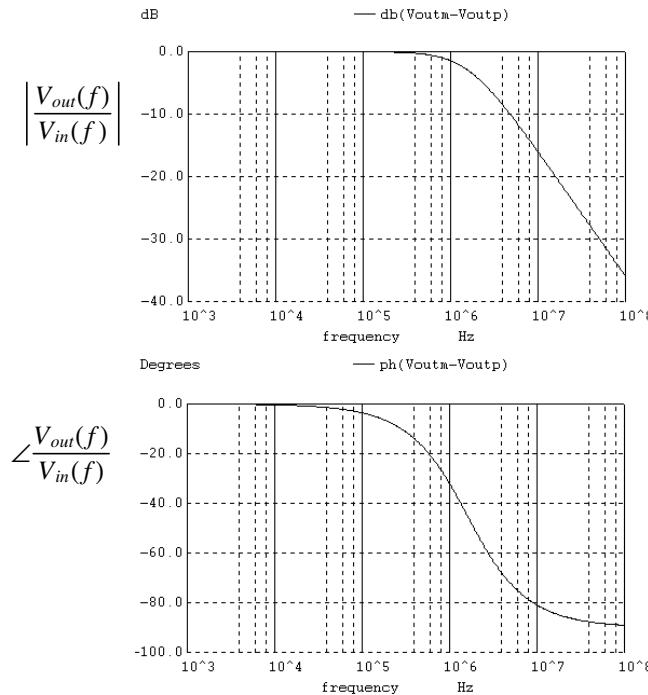


Figure 35.5 Magnitude and phase responses for the first-order filter in Fig. 35.4 if $R = 10\text{k}$ and $C = 10 \text{ pF}$.

What would happen if we switched what we define as V_{out+} and V_{out-} in the filter described in Ex. 35.1 without changing any other connections? Perhaps it is trivial, but the answer is that the output will be inverted. We can modify the block diagram of Fig. 35.2 by simply multiplying the output by -1 , as seen in Fig. 35.6. The phase shift in Fig. 35.5 would shift up or down by 180 degrees. It would vary from 180 to 90 degrees, or from

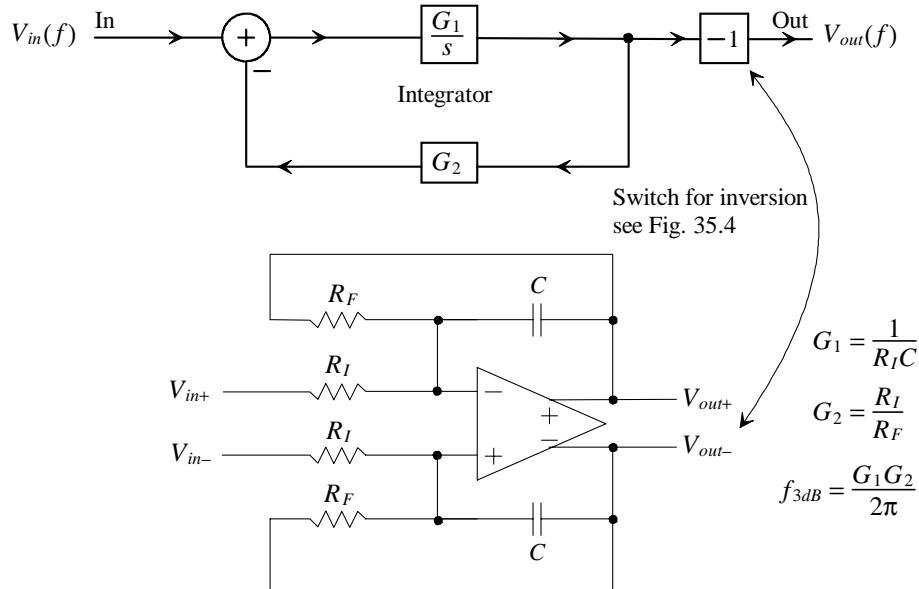


Figure 35.6 Integrator-based first-order filter.

-180 to -270 (because +180 degrees is the same as -180 degrees) instead of from 0 to -90 degrees.

If we allow the resistors used in the filter to have different values, as seen in Fig. 35.6, we can add a feedback gain to our block diagram. Assuming the outputs are labeled so that we don't have an inversion in the output of the filter (i.e., they are labeled as seen in Fig. 35.4), we can write

$$\frac{V_{in}}{R_I} - \frac{V_{out}}{R_F} = \frac{V_{out}}{1/sC} \quad (35.6)$$

It's important to note (for use later) that in order to subtract the output from the input, the voltages are first changed to currents. This equation can be rewritten as

$$\frac{V_{out}}{V_{in}} = \frac{\frac{R_F}{R_I}}{1 + sR_F C} \quad (35.7)$$

Using the block diagram in Fig. 35.6, we can write

$$\frac{V_{out}}{V_{in}} = \frac{\frac{1}{G_2}}{1 + \frac{s}{G_1 G_2}} \text{ and } f_{3dB} = \frac{G_1 G_2}{2\pi} \quad (35.8)$$

Equating coefficients in these equations results in

$$G_2 = \frac{R_I}{R_F} \text{ and } G_1 = \frac{1}{R_I C} \quad (35.9)$$

Note that at DC where $s \rightarrow 0$, the block diagram in Fig. 35.6 becomes the classic feedback diagram with the forward gain approaching infinity and a feedback factor of G_2 . Then from classic feedback theory, the closed-loop gain becomes $1/G_2$ or, for the filter in Fig. 35.6, R_F/R_I . Of course, analyzing this circuit (using loop equations) at DC when the capacitor is an open results in the same gain.

Example 35.2

Modify the filter in Ex. 35.1 so that the low-frequency gain is 20 dB.

Using Eq. (35.7) or Eq. (35.8), we leave $C = 10 \text{ pF}$ and $R_F = 10\text{k}$. To get the gain of 10, we make $R_I = 1\text{k}$. The simulation results are shown in Fig. 35.7. ■

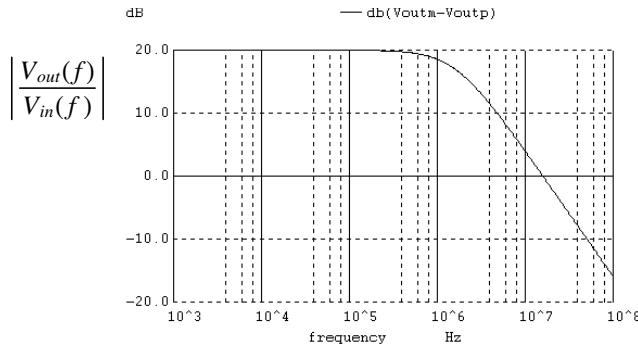


Figure 35.7 A first-order filter with gain, see Ex. 35.2.

Effects of Finite Op-Amp Gain Bandwidth Product, f_u

In the previous two examples we assumed a near-ideal op-amp. We know from Fig. 34.21 that the open-loop gain of the op-amp can be written, assuming a dominant-pole compensated op-amp, by

$$A_{OL}(f) = \frac{V_{out}}{v_+ - v_-} = \frac{A_{OLDC}}{1 + j \cdot \frac{f}{f_{3dB}}} \quad (35.10)$$

where v_+ and v_- are the voltages on the noninverting and inverting op-amp input terminals, respectively. (Note how we are using f_{3dB} in both Figs. 35.6 and 34.21 to indicate the 3 dB frequency of a frequency response.) When a practical op-amp is operating at frequencies above a few kHz, we can approximate the open-loop response (knowing the imaginary part of the denominator is much larger than the real part) as

$$A_{OL}(f) \approx \frac{A_{OLDC} \cdot f_{3dB}}{j \cdot f} = \frac{f_u}{j \cdot f} = \frac{2\pi f_u}{s} = \frac{\omega_u}{s} \quad (35.11)$$

where, again from Fig. 34.21, f_u is the frequency where the op-amp's open loop gain is unity (0 dB). Rewriting Eq. (35.6) to include the op-amp's finite gain bandwidth product (that is, f_u) and assuming, without the loss of generality that the op-amp is operating with a single-ended output (v_+ tied to V_{CM} [AC ground]), results in

$$\frac{V_{in} - v_-}{R_I} - \frac{V_{out} - v_-}{R_F} = sC \cdot (V_{out} - v_-) \quad (35.12)$$

After some algebraic manipulation with $v_- = -V_{out}/A_{OL}(f)$, we get

$$\frac{V_{out}}{V_{in}} = \frac{\frac{R_F}{R_I}}{\underbrace{1 + sCR_F}_{\text{Desired response}} + \frac{sCR_F}{A_{OL}(f)} + \frac{1}{A_{OL}(f)} \left(1 - \frac{R_F}{R_I}\right)} \quad (35.13)$$

or, with $A_{OL}(f) = \omega_u/s$

$$\frac{V_{out}}{V_{in}} = \frac{\frac{R_F}{R_I}}{s^2 \frac{CR_F}{\omega_u} + s \cdot \left[CR_F + \frac{1}{\omega_u} \left(1 - \frac{R_F}{R_I}\right)\right] + 1} \quad (35.14)$$

This equation is very revealing and shows just how significant a limitation the op-amp can be in a filter. For the moment, to simplify things, let's assume $\omega^2 \ll \omega_u/CR_F$ so that the s^2 term in Eq. (35.14) is negligible. We can then write the magnitude and phase responses as

$$\left| \frac{V_{out}}{V_{in}} \right| = \frac{\frac{R_F}{R_I}}{\sqrt{1 + \left[\omega CR_F + \frac{\omega}{\omega_u} \left(1 - \frac{R_F}{R_I}\right)\right]^2}} \text{ and } \angle \frac{V_{out}}{V_{in}} = -\tan^{-1} \left[\omega CR_F + \frac{\omega}{\omega_u} \left(1 - \frac{R_F}{R_I}\right) \right] \quad (35.15)$$

Example 35.3

Suppose a first-order filter is designed based on the topology seen in Fig. 35.6, where $\omega_u = 10/R_F C$ and $R_F/R_I = 10$. Assuming $\omega^2 \ll \omega_u/CR_F$, comment on how the magnitude and phase responses of the filter will be affected by the finite op-amp, f_u .

The op-amp's unity gain frequency is only 10 times larger than the bandwidth of the filter. This means the op-amp's closed-loop bandwidth (with a gain of 10) is equal to the desired bandwidth of the filter. The bandwidth of a gain of 10 op-amp circuit is $f_u/10$, which here is equal to the ideal filter 3 dB frequency of $1/2\pi R_F C$. The magnitude of the filter's response can be written as

$$\left| \frac{V_{out}}{V_{in}} \right| = \frac{\frac{R_F}{R_I}}{\sqrt{1 + \left[\omega \cdot \frac{CR_F}{10} \right]^2}} \text{ and } \angle \frac{V_{out}}{V_{in}} = -\tan^{-1} \left[\omega \cdot \frac{CR_F}{10} \right]$$

which shows the filter's 3 dB frequency is off by a factor of 10. ■

The point of the preceding example is, in general lowpass filter design, to minimize the effects of the op-amp's finite f_u a low value of closed-loop DC gain should be used. In the remaining discussion let's assume $R_F/R_I = 1$, so Eq. (34.14) simplifies to

$$\frac{V_{out}}{V_{in}} = \frac{1}{\frac{CR_F}{\omega_u} \cdot s^2 + s \cdot CR_F + 1} \quad (35.16)$$

The poles of this transfer function are located at

$$s_{p1,p2} = \frac{-CR_F \pm \sqrt{(CR_F)^2 - 4\frac{CR_F}{\omega_u}}}{2 \cdot \frac{CR_F}{\omega_u}} \quad (35.17)$$

noting that if $\omega_u \rightarrow \infty$, then $s_{p1} = \infty$, and $s_{p2} = -1/CR_F$ (the ideal position of the pole, see Fig. 35.1).

To get some idea of the required op-amp f_u ($\omega_u = 2\pi f_u$), let's assume that we want the pole to vary no more than 1% from the ideal location due to finite op-amp bandwidth

$$\frac{-1}{CR_F} = \frac{99}{100} \cdot \frac{-CR_F - \sqrt{(CR_F)^2 - 4\frac{CR_F}{\omega_u}}}{2 \cdot \frac{CR_F}{\omega_u}} \quad (35.18)$$

This can be rewritten as

$$1.01 = \frac{\omega_u \cdot CR_F}{2} - \frac{1}{2} \sqrt{(\omega_u \cdot CR_F)^2 - 4(\omega_u \cdot CR_F)} \quad (35.19)$$

If we let $x = \omega_u \cdot CR_F$, then we need to solve

$$x - \sqrt{x^2 - 4x} = 2.02 \quad (35.20)$$

knowing x is positive and much larger than one ($\omega_u \gg 1/CR_F$). Solving Eq. (35.20) for x , results in $x = 100$. This means the op-amp's unity gain frequency must be 100 times larger than the filter's f_{3dB} in order for the variation of this frequency (the pole) to deviate less than 1% from the ideal. If we can withstand a 10% decrease in the filter's cutoff frequency, then f_u need only be 10 times larger than the filter's f_{3dB} . Clearly, from Eq. (35.14), the first-order filter's frequency response is actually second-order when the op-amp's gain bandwidth product f_u is a factor. Therefore, the shapes of the magnitude and phase responses will deviate from the ideal first-order shapes seen in Fig. 35.1 so we can draw two very practical conclusions. First, even if it were possible to fabricate precise resistor and capacitor values, the limitations of the op-amp's finite bandwidth may still require the use of tuning when filtering with active-RC integrator-based filters. Tuning would consist of adding or removing resistors and capacitors to adjust the precise filter cutoff frequency (adding/removing the elements using either fuses or, if possible, MOSFET switches). Second, the op-amp's f_u should be at least 10 times larger than the cutoff frequency (f_{3dB}) of the filter (again assuming a closed-loop DC gain of unity, i.e., $R_F/R_I = 1$). This is a general "rule-of-thumb." Precision filters (well-defined magnitude and phase responses) would require wider bandwidth op-amps. Consider the following example.

Example 35.4

Repeat Ex. 35.1 if an op-amp is used with a DC gain of 10,000 and an f_u of 10 MHz.

Because the op-amp's A_{OLDC} is 10,000 and $f_u = 10$ MHz, the op-amp's open loop f_{3dB} is 1 kHz (see Fig. 34.21). We can use the circuit shown in Fig. 35.8 (see also Fig. 32.45) in our SPICE simulation to model an op-amp with finite f_u . The RC in Fig. 35.8 is selected to give an op-amp open loop f_{3dB} of 1 kHz.

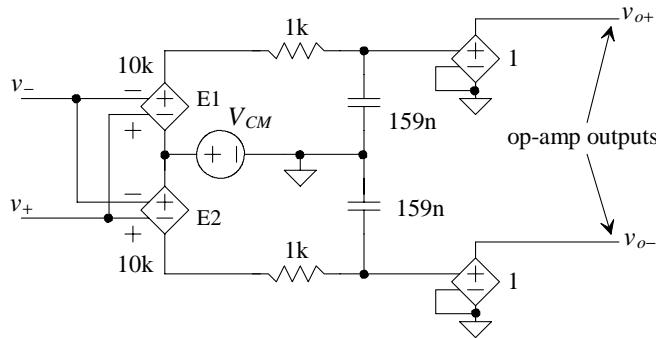


Figure 35.8 SPICE modeling a differential input/output op-amp with finite bandwidth.

The 3-dB frequency of the filter described in Ex. 35.1 is, under ideal conditions, 1.59 MHz. Because our op-amp's unity gain frequency is only 10 MHz, we would expect, from Eq. (35.16), the op-amp to affect the frequency response of the filter. Figure 35.9 shows the simulation results using the op-amp model of Fig. 35.8. Comparing Fig. 35.9 to Fig. 35.5, we see differences in both the magnitude and phase responses of the filters. The magnitude response of Fig. 35.9 initially rolls off at 20 dB/decade below the ideal 1.59 MHz. Around 10 MHz the response transitions to 40 dB/decade. Clearly this faster roll off is the result of the op-amp's closed-loop pole coming into play. The limiting behavior of the op-amp, when looking only at the magnitude response, may be welcome (the filter's response rolls off faster) in a lowpass filter. However, it is not welcome in other filters (a highpass filter, for example). Figure 35.10 shows what happens if we decrease the filter's 3 dB frequency to 159 kHz by increasing the resistors used to 100k. What we are doing here is showing how making the op-amp's bandwidth much larger than the filter's affects the frequency response of the circuit. The magnitude response starts to fall off at -40 dB/decade at the op-amp's unity gain frequency, f_u , of 10 MHz. Also seen in Fig. 35.10 is the phase response of the filter. The op-amp's (closed-loop) phase response, which starts rolling off one decade below f_u , results in the final phase shift of the filter approaching -180 degrees. ■

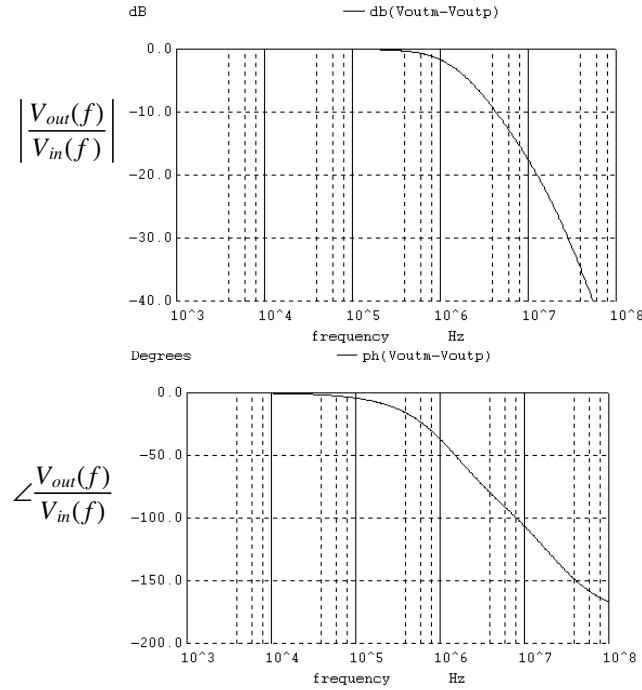


Figure 35.9 Magnitude and phase responses for the first-order filter in Fig. 35.4 if $R = 10\text{k}$ and $C = 10\text{ pF}$ using an op-amp with a 10 MHz unity-gain frequency.

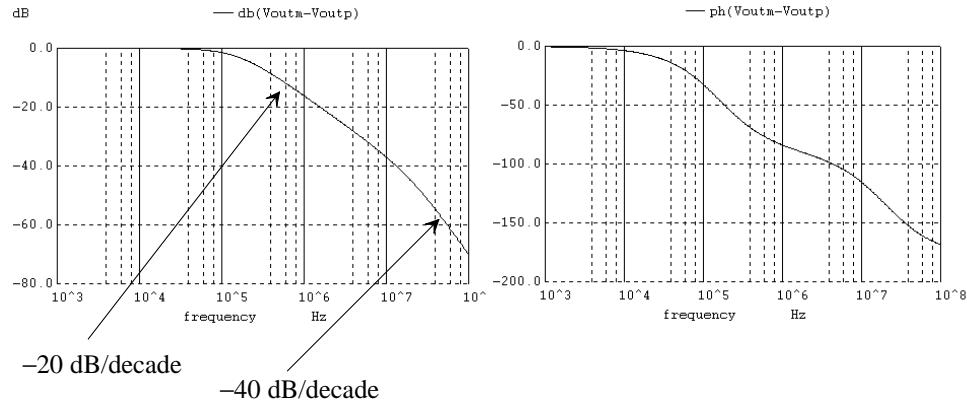


Figure 35.10 Increasing the resistance to 100k and replotting Fig. 35.9.

To model the effects of op-amp finite bandwidth on an active-RC filter's frequency response we can add a pole to the ideal transfer function. Assuming unity gain in the passband (see Eq. [35.3]) results in

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{1}{1 + s/G} \cdot \underbrace{\frac{1}{1 + j\frac{f}{f_u}}}_{\text{Undesired}} \quad (35.21)$$

This result could have been used in the previous example to predict how the op-amp affects the filter's behavior. If the filter has gain (> 1) in the passband, see Eq. (35.8), we can modify this equation to read

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{\frac{1}{G_2}}{1 + \frac{s}{G_1 G_2}} \cdot \underbrace{\frac{1}{1 + j\frac{f}{G_2 f_u}}}_{\text{Undesired}} \quad (35.22)$$

For a higher order filter we would multiply the desired frequency response by the undesired term's (the op-amp's) response for each op-amp used in the circuit. Clearly, this limits the order of the filter (limits the number of op-amps used in a circuit; a first-order filter uses one op-amp, a second-order filter uses two op-amps, etc.). This is especially true if the filter has a passband approaching the f_u of the op-amps used.

Active-RC SNR

Consider the single-ended active-RC filter shown in Fig. 35.11. Let's assume an ideal op-amp with a maximum RMS output voltage of $VDD/(2\sqrt{2})$. The RMS input-referred noise of the filter, assuming thermal noise dominates over the bandwidth of interest, is simply $\sqrt{kT/C}$. The filter's SNR can then be written as

$$\text{SNR} = 20 \cdot \log \frac{VDD/(2\sqrt{2})}{\sqrt{kT/C}} = 10 \cdot \log \frac{VDD^2/8}{kT/C} \quad (35.23)$$

This result shouldn't be too interesting at this point. As we've already seen, the size of the integrating capacitor fundamentally sets the SNR in integrator-based data converters or

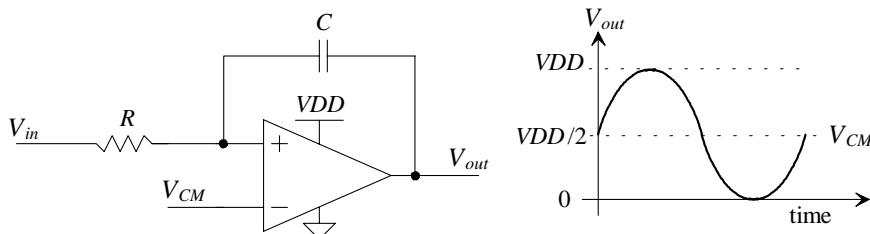


Figure 35.11 Estimating maximum possible SNR of an active-RC filter.

modulators. But consider the following: the maximum electrical energy stored in the capacitor used in an integrator is

$$\text{Maximum electrical energy} = \frac{1}{2}C \cdot \left(\frac{VDD}{2}\right)^2 \quad (35.24)$$

Equation (35.23) can then be rewritten as

$$SNR = 10 \cdot \log \frac{VDD^2/8}{kT/C} = 10 \cdot \log \frac{\frac{1}{2}C\left(\frac{VDD}{2}\right)^2}{kT} = 10 \cdot \log \frac{\text{Electrical energy}}{\text{Thermal energy}} \quad (35.25)$$

This equation can also be used to estimate the fundamental dynamic range, DR, of a filter. That is, we can estimate DR by assuming $DR = SNR$. Of course, as discussed in Ch. 31, a more accurate estimate of DR is the measured signal-to-noise plus distortion ratio, SNDR. Practically, DRs approaching 90 dB (15 bits) can be attained using active-RC filters with good polysilicon resistors (to avoid the large nonlinear voltage coefficient associated with diffused or implanted resistors) and linear capacitors. Bandwidths approaching 50 MHz, assuming 500 MHz f_u op-amps are used, can be attained (at, of course, lower DRs).

35.1.3 MOSFET-C Integrators

Let's now look at a variation of the active-RC filter where the resistors are replaced with MOSFETs. Figure 35.12 shows a MOSFET-C filter. In order for the MOSFETs to behave as resistors they must remain in the triode region. Using long length devices helps ensure triode operation. Because the MOSFETs are operating as resistors, their speed is not governed by their gate-source voltage or channel length, as indicated in Eq. (33.20). However, the linearity of the MOSFET resistors is still very important as is the possibility that the MOSFETs will introduce a parasitic pole into the filter's frequency response because of the distributed resistance/capacitance of the channel (Fig. 35.12). Figures 33.23 and 33.29 show how the channel resistance of an NMOS device changes with V_{DS} . For large input signals, the active MOSFET resistors become nonlinear, resulting in filters with SNDRs of around only 40 dB. The bandwidth of the MOSFET-C filters parallels that of the active-RC filters.

We might be questioning the usefulness of the MOSFET-C filter with an SNDR of only 40 dB. Clearly this filter will only find use in data conversion systems using six bits of resolution or less (36 dB DR) or in systems that process continuous-time signals. The big benefit of this filter over the active-RC filter is its ability to be tuned. Tuning the active-RC filter required adding or removing, via switches or fuses, resistors or capacitors in parallel or series with the existing resistors and capacitors. Tuning the MOSFET-C filter shown in Fig. 35.12 can be accomplished by adjusting V_{tune} upwards or downwards. If we assume long-channel behavior, we can write the resistance (see Ch. 9) of the MOSFETs in terms of V_{tune} (assuming the input common-mode voltage of the op-amp is 0) as

$$R_n = \frac{1}{KP \cdot \frac{W}{L} \cdot \left(V_{tune} - V_{THN} - \underbrace{V_{DS}}_{=V_{in}} \right)} \quad (35.26)$$

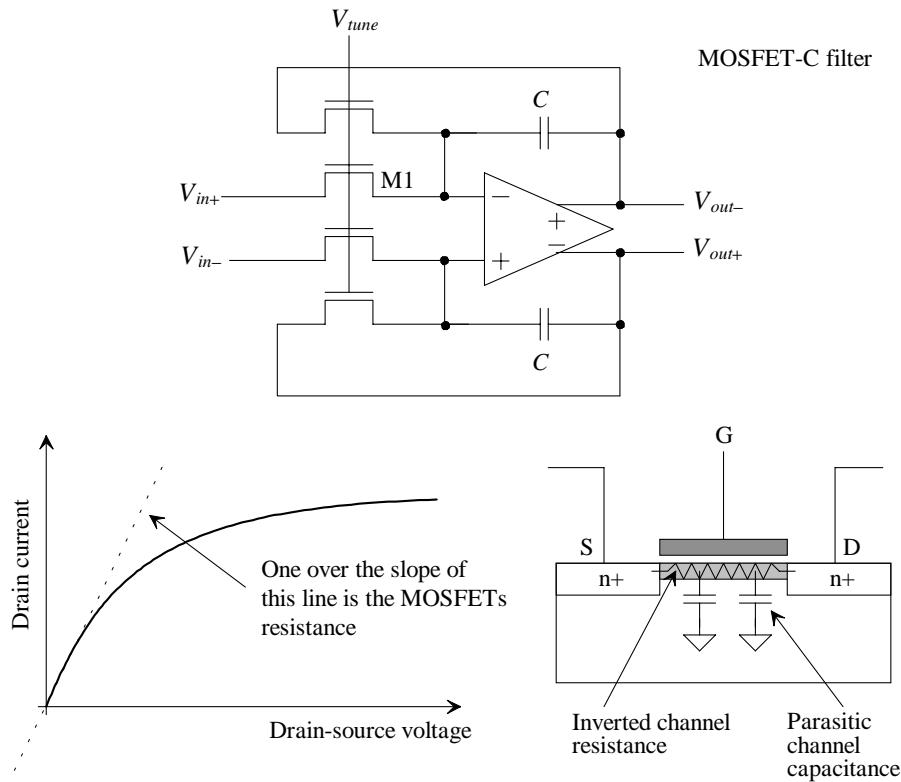


Figure 35.12 A first-order MOSFET-C filter.

The current through M1 in Fig. 35.12 is

$$\frac{V_{in+}}{R_{n1}} = V_{in+} \cdot KP \cdot \frac{W}{L} (V_{tune} - V_{THN} - V_{in+}) \quad (35.27)$$

Some improvement in the linearity of the MOSFET resistors, say 10 dB (resulting in an SNDR of 50 dB), can be achieved by utilizing the fully-balanced signals available in the circuit. Consider replacing M1 in Fig. 35.12 with the pair of MOSFETs, M1A and M1B, shown in Fig. 35.13. The resulting current is now

$$\frac{V_{in+}}{R_{n1A}} + \frac{V_{in-}}{R_{n1B}} = KP \cdot \frac{W}{L} [V_{in+} \cdot (V_{tune+} - V_{THN} - V_{in+}) + V_{in-} \cdot (V_{tune-} - V_{THN} - V_{in-})] \quad (35.28)$$

Knowing $V_{in+} = -V_{in-}$ and letting $V_{tune} = V_{tune+} - V_{tune-}$, we can write the equivalent current through M1 as

$$\frac{V_{in+}}{R_{n1}} = V_{in+} \cdot KP \cdot \frac{W}{L} \cdot V_{tune} \quad (35.29)$$

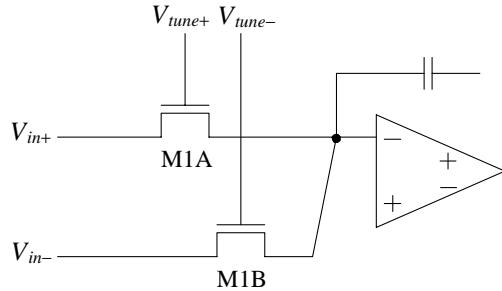


Figure 35.13 Linearizing MOSFET resistors.

The result is that the nonlinear behavior of the MOSFET's channel resistance due to the changing drain-source voltage cancels to a first order. Figure 35.14 shows the implementation of a first-order MOSFET-C filter using linearized MOSFETs.

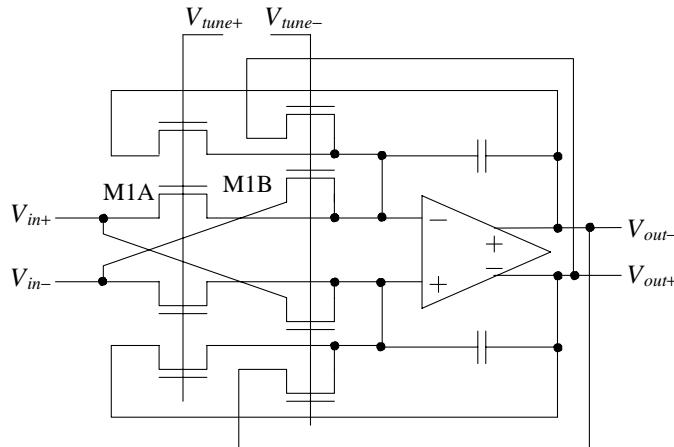


Figure 35.14 First-order MOSFET-C filter using linearized MOSFET resistors.

Why Use an Active Circuit (an Op-Amp)?

Before going any further, let's realize that we can get the exact same frequency performance using a simple resistor/capacitor or MOSFET/capacitor as we get when we use these elements with an op-amp. So, "Why use the op-amp?" The answer to this question comes when we realize that when a capacitive or resistive load is connected to the output of the filter (without an active element), the frequency behavior changes. Using the op-amp allows us to drive an arbitrary (within reason) capacitive or resistive load. Using an active element will also allow us to cascade first-order sections to implement higher order filters.

35.1.4 g_m -C (Transconductor-C) Integrators

We first described operational-transconductance amplifiers, OTAs, back in Ch. 25. Figure 35.15 shows a schematic symbol, transfer curves, and a possible implementation for an OTA (based on a fully-differential diff-amp). Transconductor-C, or g_m -C, filters use a circuit, a transconductor, that provides a linear voltage-current transfer curve. Our OTA in Fig. 35.15 does behave like a transconductor over a portion of the input voltage range but becomes nonlinear for large input voltage differences, $v_{in+} - v_{in-}$. By increasing the lengths of the NMOS diff-pairs used in the OTA, we can increase the linear common-mode range of the OTA, making it appear as though it were a transconductor. The fundamental problems with increasing the lengths of the diff-pair MOSFETs are the increase in the OTA's input capacitance (affecting the location of the filter's poles and zeroes) and, perhaps more fundamentally, the inherent reduction in the MOSFET's f_T (see Eq. [33.20]). The MOSFET parasitic capacitances introduce parasitic poles into the filter's transfer function.

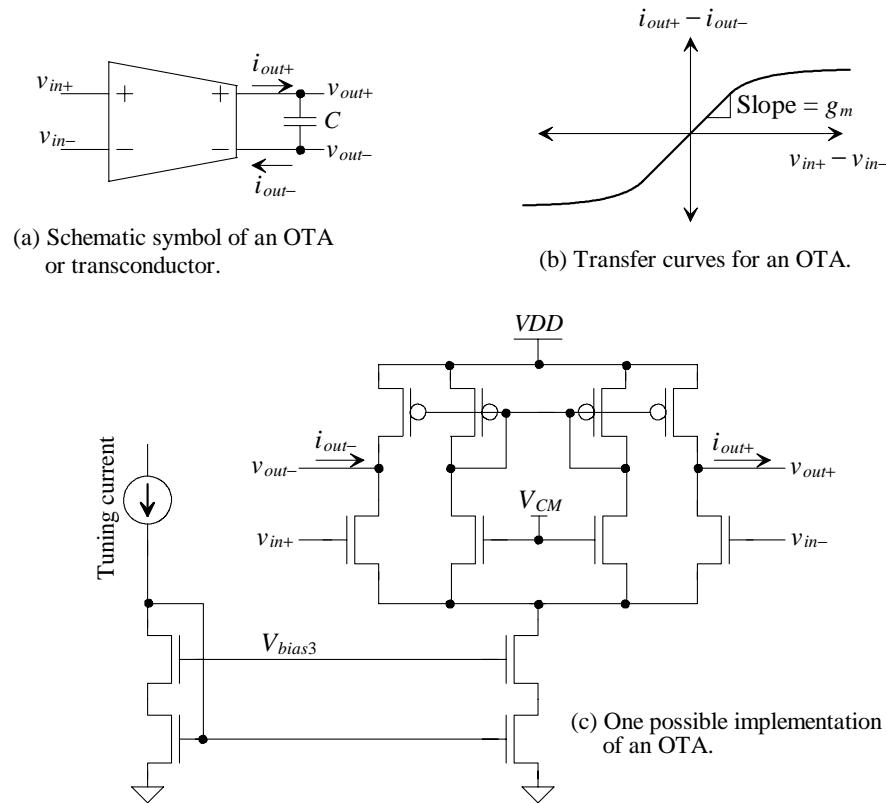


Figure 35.15 Showing an implementation of an OTA and transfer curves.

Before discussing these issues in more detail let's look at an important limitation of g_m -C filters; namely, the fact that the transconductor's input voltage must vary. As we discussed in Sec. 34.1.2, in any precision application the input voltage must remain constant because of the roll-off associated with the amplifier's CMRR (unless, of course, the common-mode voltage can be held at a precise value). This *limits the SNDR of g_m -C filters to around 50 dB*. Again, not too useful if used as an antialiasing or reconstruction filter unless the system's resolution is less than eight bits (48 dB SNR). The g_m -C filter finds extensive use in continuous-time signal processing.

We can relate the input voltage difference to the output voltage difference for the circuit in Fig. 35.15a using

$$V_{out+} - V_{out-} = \frac{I_{out+}}{j\omega C} = \frac{I_{out-}}{j\omega C} = \frac{g_m(V_{in+} - V_{in-})}{j\omega C} \quad (35.30)$$

where, for example, $V_{out+} = V_{out+}(f)$, is the frequency domain representation of the output voltage $v_{out+}(t)$. Comparing this result to Eq. (35.5), we can use the same design techniques if we require

$$G = \frac{1}{RC} = \frac{g_m}{C} \text{ or } f_{3dB} = \frac{G}{2\pi} \quad (35.31)$$

The big benefit of the g_m -C filter over the active-RC filter is the ability to tune the filter by adjusting the transconductor's g_m .

The circuit of Fig. 35.15a implements an integrator, as does the active-RC circuit of Fig. 35.3. However, as seen in Fig. 35.2, we also need to implement a summing block in a first-order filter. Toward the goal of implementing the summing block, consider the transconductor circuit shown in Fig. 35.16a. The output current of a single transconductor

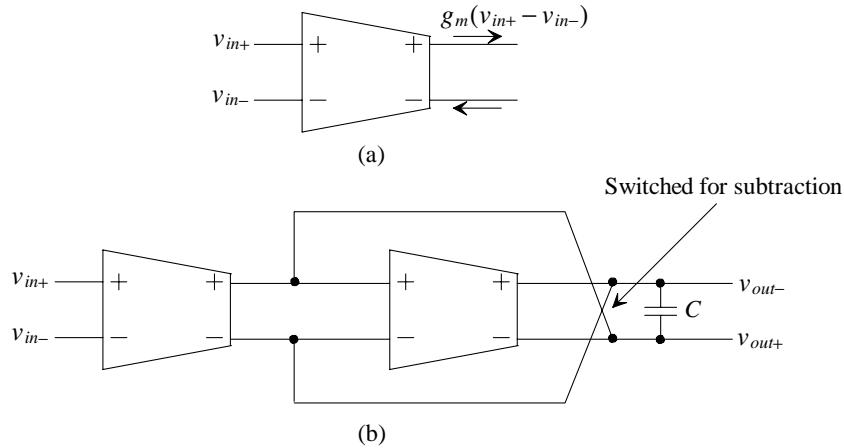


Figure 35.16 Implementing a first-order filter using transconductors.

is, as shown in the figure, $g_m(V_{in+} - V_{in-})$. We can sum this current with the output current from the second transconductor to implement the summing block in Fig. 35.2. As seen in Fig. 35.16b, the outputs of the two OTAs are combined, so the currents output from each transconductor subtract. Assuming each transconductor has the same transconductance, we can write

$$g_m(V_{in+} - V_{in-}) - g_m(V_{out+} - V_{out-}) - j\omega C(V_{out+} - V_{out-}) = 0 \quad (35.32)$$

or

$$\frac{V_{out+} - V_{out-}}{V_{in+} - V_{in-}} = \frac{1}{1 + j\omega C \cdot \frac{1}{g_m}} \quad (35.33)$$

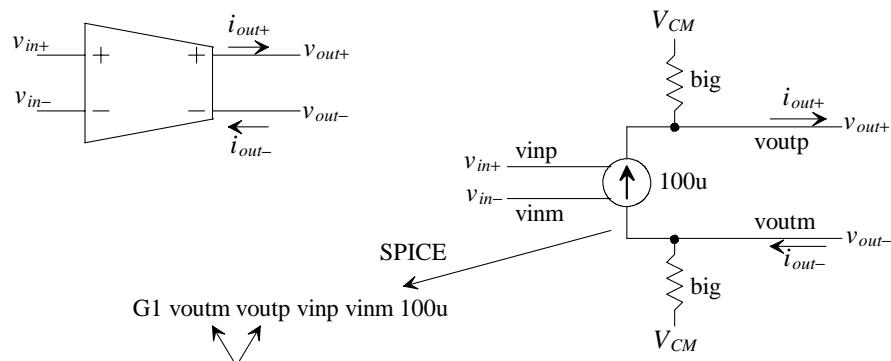
If the transconductors have different g_m s, we can design a filter with a DC gain (see Problem 35.10), which can be characterized using Eq. (35.8).

Example 35.5

Repeat Ex. 35.1 using a g_m -C filter with a g_m of 100 $\mu\text{A/V}$.

To simulate a transconductor using SPICE, a voltage-controlled current source can be used as seen in Fig. 35.17. In order to set the output common-mode voltage to V_{CM} in the simulation, we add the large resistors (whose values can be changed to simulate the finite, nonideal, output resistance of the OTA) connected to V_{CM} . Not using these resistors after reviewing Fig. 35.16 would result in an unknown common-mode voltage on the second transconductor input.

In order to have the same time constant, and thus pole location, as in Ex. 35.1, let's set the capacitor value, in the schematic of Fig. 35.16 to 10 pF. The value of the transconductance, $1/g_m$, is 10k. The simulation results are shown in Fig. 35.18. As we would expect, the shape follows, exactly, that of the active-RC filter in Fig. 35.5. Also, although not shown, the phase response matches as well. ■



Notice how SPICE defines positive current flow as current flowing from the + terminal to the - terminal

Figure 35.17 Modeling an ideal transconductor in SPICE using a voltage-controlled current source.

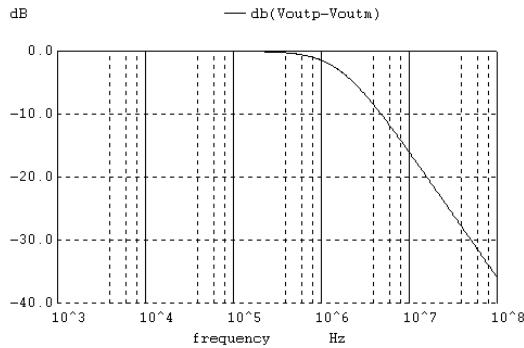


Figure 35.18 Simulation results for Ex. 35.5.

Common-Mode Feedback Considerations

In most of our high-gain OTAs, such as the one shown in Fig. 34.36, we used the load capacitances to compensate the amplifier. These capacitances compensated both the normal, forward, differential signal path as well as the CMFB path. Reviewing Fig. 35.15, we see that the capacitance in part (a) indeed does provide a load for differential signals. However, any signal that is common to both outputs (a common-mode signal) doesn't cause a displacement current to flow through the capacitor. Because both sides of the capacitor change at the same rate for common-mode signals, the change in voltage across the capacitor is zero. This can result in unstable CMFB loops. Figure 35.19 shows how we would break the capacitor in Fig. 35.15 up into two components to provide the same loading for differential and common-mode signals. We can write

$$\frac{V_{out+}}{1/j\omega 2C} = I_{out+} = I_{out-} = \frac{-V_{out-}}{1/j\omega 2C} \quad (35.34)$$

$$I_{out+} = g_m(V_{in+} - V_{in-}) = I_{out-} \quad (35.35)$$

$$\frac{V_{out+} - V_{out-}}{V_{in+} - V_{in-}} = \frac{g_m}{j\omega C} \quad (35.36)$$

which is the same result as Eq. (35.30).

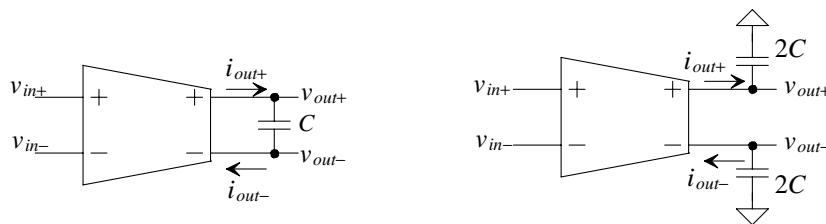


Figure 35.19 Showing how we break the capacitor up to provide a load for the CMFB circuit.

A High-Frequency Transconductor

A transconductor, Fig. 35.20, can be implemented using the common-mode noise elimination discussed in Ch. 33 [6]. To increase the input common-mode range, the lengths of INV1 and INV2 can be increased as in the diff-amp of Fig. 35.15. This, again, lowers the location of the parasitic poles introduced into the transconductor's response. Note, in this circuit, that other than the power supplies and the tuning voltage, there are only two sets of nodes: the input and the output nodes. This allows the transconductor's capacitances to sum with the load capacitances and be tuned out.

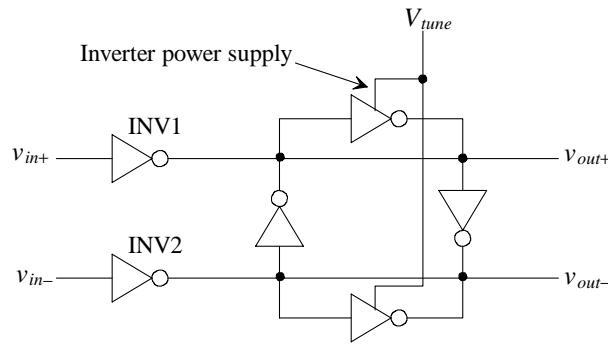


Figure 35.20 High-frequency transconductor (see also Figs. 33.36 - 33.39).

35.1.5 Discrete-Time Integrators

Let's consider using the DAIs in Fig. 31.80 to implement a first-order filter. The gain of a DAI is

$$\frac{V_{out}(z)}{V_{in}(z) - V_{out}(z)} = \frac{C_L}{C_F} \cdot \frac{z^{-1}}{1 - z^{-1}} \quad (35.37)$$

Knowing

$$z = e^{j2\pi \frac{f}{f_s}} \text{ and } e^{j\theta} = \cos \theta + j \sin \theta \quad (35.38)$$

we can write

$$\frac{V_{out}(z)}{V_{in}(z) - V_{out}(z)} = \frac{C_L}{C_F} \cdot \frac{1}{(\cos 2\pi \frac{f}{f_s} - 1) + j \sin 2\pi \frac{f}{f_s}} \quad (35.39)$$

The sampling frequency (the frequency the discrete-time filter is clocked at) is f_s , while the filter's input frequency is labeled f . If we require $f \ll f_s$ (say at least sixteen times less) where $\cos 2\pi \frac{f}{f_s} \approx 1$ and $\sin 2\pi \frac{f}{f_s} \approx 2\pi \frac{f}{f_s}$, then we can rewrite Eq. (35.39) as

$$\frac{V_{out}(z)}{V_{in}(z) - V_{out}(z)} = \frac{C_I}{C_F} \cdot \frac{z^{-1}}{1 - z^{-1}} \approx \frac{C_I}{C_F} \cdot \frac{f_s}{j2\pi f} = G \cdot \frac{1}{s} \quad (35.40)$$

where

$$G = \frac{C_I}{C_F} \cdot f_s = \frac{1}{C_F R_{sc}} \text{ and } f_{3dB} = \frac{G}{2\pi} \quad (35.41)$$

placing the gain of the integrator in the same form as Eqs. (35.4). From Ch. 27 we should recognize R_{sc} as a switched capacitor resistor

$$R_{sc} = \frac{1}{C_I \cdot f_s} = \frac{T_s}{C_I} \quad (35.42)$$

noting that in Ch. 27 we used f_{clk} to indicate the frequency of the clock waveform used with the filter (not f_s as we are here). The equivalent block diagrams for first-order filters using CAIs and using the DAIs (again assuming $f \ll f_s$) are compared in Fig. 35.21.

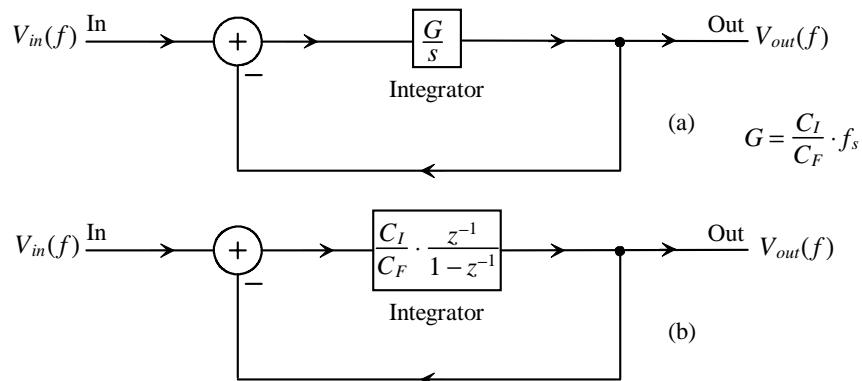


Figure 35.21 Block diagram of an integrator-based lowpass filter.
 (a) Continuous-time and (b) the discrete-time equivalent.

Example 35.6

Sketch the implementation of a DAI-based (switched-capacitor), first-order filter with characteristics like the one in Ex. 35.1. Using SPICE simulate the design.

The schematic of the filter is shown in Fig. 35.22. Here we are assuming the clocking frequency of the filter is 100 MHz. If the feedback capacitance, C_F , is 10 pF, the size of the input capacitor, C_p , is then, from Eq. (35.42) and knowing R_{sc} is 10k, 1 pF. The 3 dB frequency of the filter is, once again, $1/2\pi R_{sc} C_F = 1.59$ MHz. Because of the time-domain (clock) component of the filter, we can't use an AC analysis for the SPICE simulation. Let's apply a 1 V peak-to-peak sinewave to the filter at 1.59 MHz and verify the output of the filter is 3 dB down (0.707 V peak-to-peak). The results are seen in Fig. 35.23.

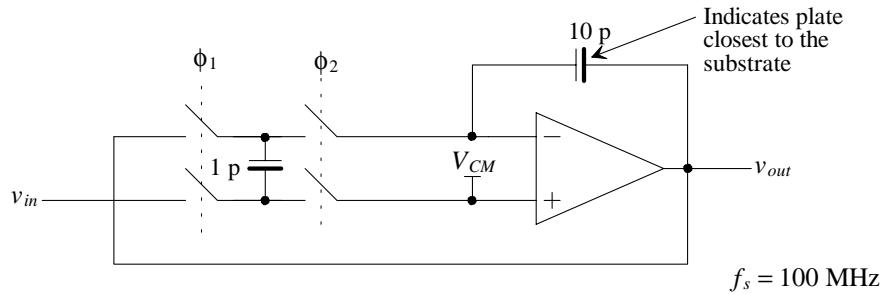


Figure 35.22 A switched capacitor, first-order filter similar to the one described in Ex. 35.1. See Fig. 31.80a for additional information concerning the DAI topology used in this filter.

Let's comment on the exact transfer function of the DAI in Fig. 35.22. It might be helpful, at this point, to review Fig. 31.80a. We see in Fig. 35.22 that the output is indeed fed back to the input through a ϕ_1 controlled switch. The result is that the output, through the feedback loop, sees one clock cycle delay, z^{-1} . The output is assumed settled on the falling edge of ϕ_2 during each clock cycle. Because of this, the input, which is settled on the falling edge of ϕ_1 , sees only a half clock cycle delay, $z^{-1/2}$. This means that the DAI in Fig. 35.22 really has a transfer function of

$$V_{out}(z) = \frac{C_I}{C_F} \cdot \frac{z^{-1}}{1-z^{-1}} \cdot [V_{in}(z) \cdot z^{1/2} - V_{out}(z)] \quad (35.43)$$

If we think of the input signal arriving a half-clock cycle earlier, then the only difference in the transfer function here, when compared to the continuous-time equivalent and assuming $f \ll f_s$, is a small phase difference. We can delay the input signal a full clock cycle by adding a ϕ_1 controlled switch on the output of the filter. However, because this switch may be part of the next filter section, we don't discuss this option further. ■

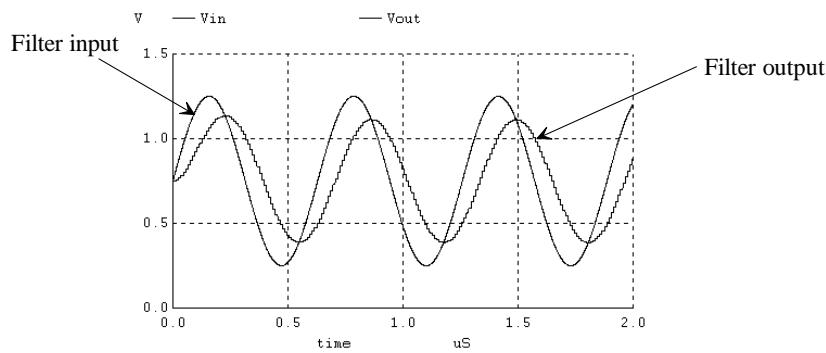


Figure 35.23 Output of the switched-capacitor circuit in Fig. 35.22.

Example 35.7

Sketch the switched-capacitor implementation of the discrete-time lowpass (first-order) filter shown in Fig. 35.24.

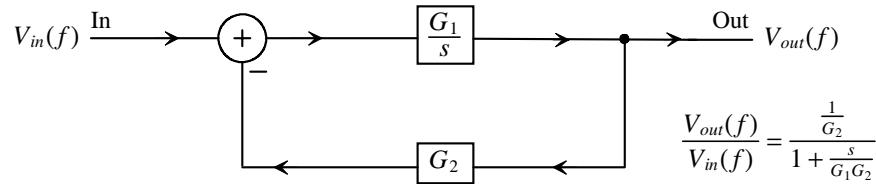


Figure 35.24 General implementation of a lowpass first-order filter.

The implementation is seen in Fig. 35.25. The coefficients, see question 35.12, are

$$G_1 = \frac{C_{I1}}{C_F} \cdot f_s \text{ and } G_2 = \frac{C_{I2}}{C_F} \cdot f_s \cdot \frac{1}{G_1} = \frac{C_{I2}}{C_{I1}} \quad (35.44)$$

The DC gain, as seen in Eq. (35.8) is set by the ratio of C_{I2} to C_{I1} ($1/G_2$), and the filter's 3 dB frequency is

$$f_{3dB} = \frac{G_1 G_2}{2\pi} \quad (35.45)$$

Note that the DAI used in this filter has a transfer function of

$$V_{out}(z) = \frac{z^{-1}}{1 - z^{-1}} \cdot \left[\frac{C_{I1}}{C_F} \cdot V_{in}(z) \cdot z^{1/2} - \frac{C_{I2}}{C_F} \cdot V_{out}(z) \right] \quad (35.46)$$

■

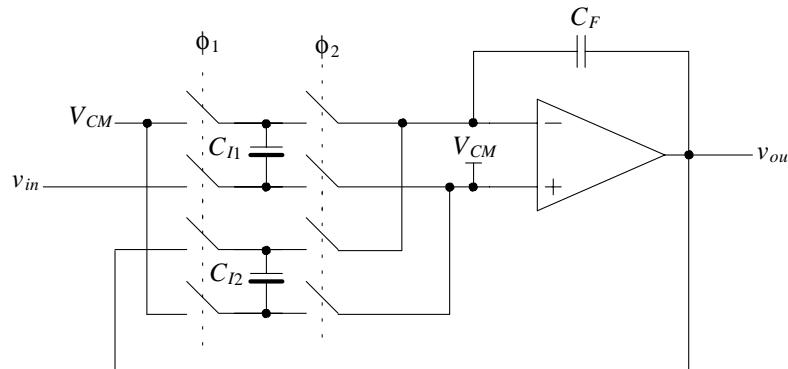


Figure 35.25 Implementation of the block diagram shown in Fig. 35.24.

The big benefit of switched-capacitor-based filters is the fact that the filters' poles and zeroes are determined by a ratio of capacitors and an external clock frequency (which is often a precise frequency set by a crystal oscillator). No tuning is needed. Varying the clocking frequency can precisely set the filter's characteristics for adaptive filtering (changing the filter's characteristics on the fly). Switched-capacitor filters with SNDRs in the 90 dB range have been attained at audio frequencies. SNDRs in the 60-70 dB range have been achieved when the filter is operating in the MHz range.

In the previous two examples we used ideal op-amps and didn't concern ourselves with the potential aliasing resulting from the analog sample and hold operation on the input of the filter. Back in Secs. 34.2.1 and 34.2.2 we derived the requirements placed on the op-amp's open-loop gain and unity-gain frequency for a given data converter resolution (which can easily be converted to SNR for the ideal situation using Eq. [31.4]). An analog antialiasing filter, AAF, (that is, not discrete-time filter) must be used to remove the potential aliased signal prior to sampling. As with the noise-shaping modulator-based data converters the fact that the sampling frequency, f_s , is much larger than the input frequencies of interest allows a relaxed AAF design. Indeed, the resistance of the MOSFET switches used combined with the switched input capacitance (C_i) form a lowpass filter. This filtering may serve as the switched-capacitor filter's AAF.

An Important Note

If we pause for a moment and think about the filters we have covered in this chapter (and the data converter topologies in Ch. 34), we come to the realization that all require precise analog components. High-speed, wide-bandwidth op-amps and/or components with precise matching or absolute values are needed. We might argue that this would be a reason to focus our discussion on digital filtering (filters using only multipliers, delays, and adders) instead of filters using analog components. However, digital filters can't alone filter an analog waveform without first running the signal through an ADC. Further, traditional digital filters that use general-purpose multipliers at reasonable speeds can be very large (take up a significant layout area). So large in fact as to not be practical in a general purpose filtering application. Special-purpose chips have been fabricated specifically for digital filtering (called digital signal processors, DSPs).

At this point we should remember that the reason we spent a whole chapter, Ch. 32, on noise-shaping topologies (see Sec. 31.3.2) was that they reduce the requirements placed on the analog components in the circuit. *Isn't it logical then to attempt to combine noise-shaping with purely digital filtering for the design of future analog interfaces?* The answer is obviously, "yes"; however, as mentioned above, we have some caveats. While the resulting interface will place lower demands on the precision of the analog circuitry, we'll need to (1) develop digital filters that don't rely on complex multipliers. The multiplications we do use should be simple, perhaps requiring an additional adder, or trivial (shift) multiplication. Also we'll need to use the digital filters to not only filter the input signal but to (2) filter out the modulation noise present in the output of the NS modulator. We used a single-bit quantizer in the majority of our examples in Ch. 32. For a general analog interface (where we are using the term "analog interface" to indicate both analog-to-digital conversion and filtering), with up to 60 dB DR a better choice is to use a

multibit quantizer. We say "better" to indicate that we can achieve a higher DR at a lower oversampling ratio when using a multibit modulator. We'll briefly cover this type of interface at the end of this chapter. For now let's show how the filter of Fig. 35.21b can be implemented as a purely digital filter. This will also allow us to derive the exact transfer function of the filter of Fig. 35.22 when the input frequency gets large (where the DAI doesn't behave like a CAI, as indicated by Eq. [35.40]).

Exact Frequency Response of a First-Order Discrete-Time Digital (or Ideal SC) Filter

Figure 35.26 shows the digital only equivalent of Figs. 35.21a and 35.21b. We have replaced the ratio of capacitors, C_I/C_F , with the variable A in the figure. To determine the transfer function we can write

$$V_{out}(f) = A \cdot [V_{in}(f) - V_{out}(f)] \cdot \frac{z^{-1}}{1-z^{-1}} \quad (35.47)$$

or

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{Az^{-1}}{Az^{-1} + 1 - z^{-1}} = \frac{A}{z - (1-A)} \quad (35.48)$$

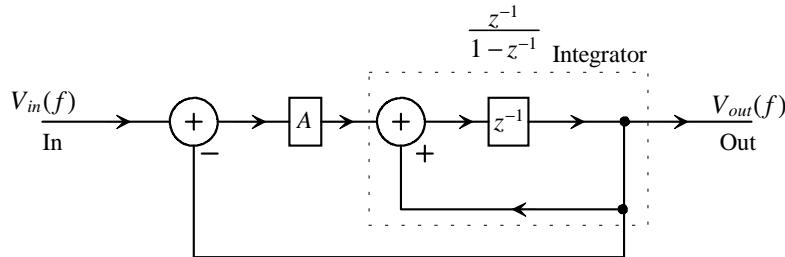


Figure 35.26 Digital block diagram of an integrator-based lowpass filter (digital version of the block diagrams of Fig. 35.21).

Figure 35.27 shows the z -plane plot and magnitude response for this first-order filter. Note the similarity to Fig. 31.62. Remembering all digital filters have a periodic frequency response (a period of f_s or one complete revolution around the unit circle), we can compare this filter's response to the filters in the previous examples. To do so let's assume $f_s = 100$ MHz and write, from Eq. (35.41),

$$f_{3dB} = \frac{Af_s}{2\pi} = 1.59 \text{ MHz} = \frac{0.1 \cdot 100 \text{ MHz}}{2\pi} \text{ that is, } A = 0.1 \quad (35.49)$$

We can write the magnitude response of Eq. (35.48) as

$$\left| \frac{V_{out}}{V_{in}} \right| = \frac{A}{\sqrt{(\cos 2\pi f/f_s - 1 + A)^2 + \sin^2 2\pi f/f_s}} \quad (35.50)$$

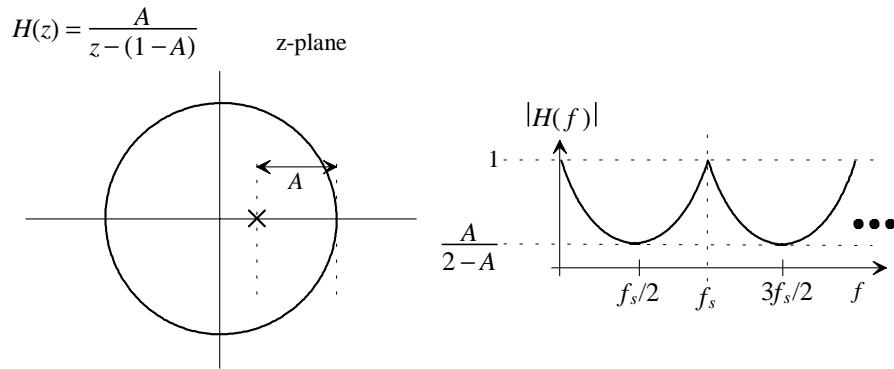


Figure 35.27 The z-plane representation and magnitude response of a first-order digital filter.

Figure 35.28 shows the responses of the first-order discrete-time filter (or the SC filter using an ideal, that is, no dominate pole, op-amp). The maximum attenuation of the filter occurs at $f_s/2$ or 50 MHz here and is, from Fig. 35.27 with $A = 0.1$, 0.0526 or -25.6 dB.

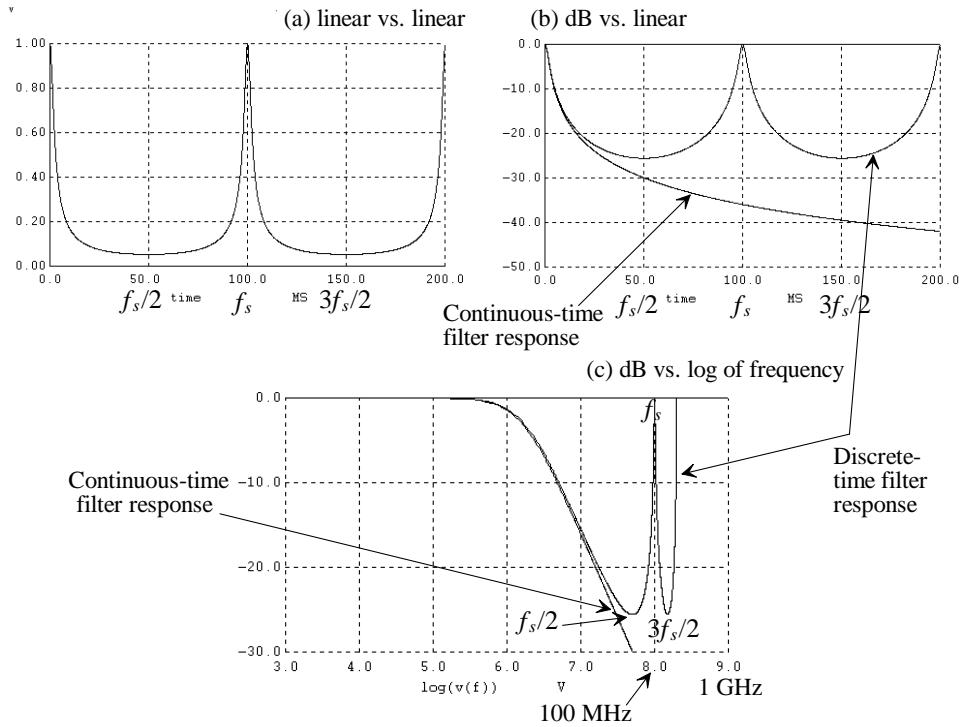


Figure 35.28 The magnitude response of the discrete-time first-order filter of Fig. 35.26 with an A of 0.1.

35.2 Filtering Topologies

In this section we present some basic filter building blocks using integrators. While we show the continuous analog implementations of these filters, our focus, in preparation for the next section covering the combination of NS modulators and digital filters, will be on using the results to design digital filters.

35.2.1 The Bilinear Transfer Function

Consider the block diagram of the general first-order filter shown in Fig. 35.29. We can relate the filter's output to its input using

$$[V_{in}(f) \cdot [1 + sG_3] - G_2 \cdot V_{out}(f)] \cdot \frac{G_1}{s} = V_{out}(f) \quad (35.51)$$

or

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{1}{G_2} \cdot \frac{1 + \frac{s}{1/G_3}}{1 + \frac{s}{G_1 G_2}} \quad (35.52)$$

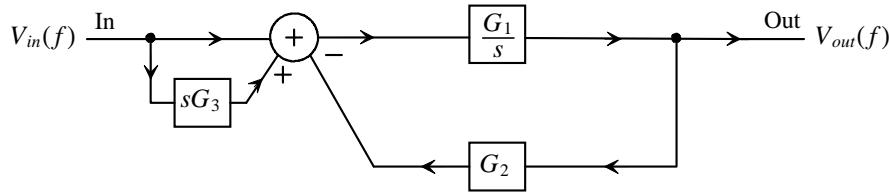


Figure 35.29 Implementation of a bilinear transfer function using an integrator.

This filter's transfer function is termed "bilinear" because it is the ratio of two linear functions. Using this topology we can implement lowpass, allpass (used for phase shifting), and highpass filters (keeping in mind that the highpass filter will ultimately change into a bandpass filter response because of the op-amp's or transconductor's high frequency rolloff). The location of the filter's pole is given by

$$f_{3dB,pole} = \frac{G_1 G_2}{2\pi} \quad (35.53)$$

while the filter's zero is located at

$$f_{3dB,zero} = \frac{1}{2\pi G_3} \quad (35.54)$$

The filter's gain at DC, in all cases, is

$$A_{DC} = \frac{1}{G_2} \quad (35.55)$$

Active-RC Implementation

The active-RC implementation of the bilinear transfer function is seen in Fig. 35.30. Again, as mentioned earlier, the resulting active-RC transfer function suffers from poor repeatability from one process run to the next. The RC time constants must be tuned, on-chip, with fuses (or switches) and adding/removing resistors or capacitors. Note how the summation is implemented by changing the input/output voltages to currents. The currents are summed at the inputs of the op-amp (which remain, ideally, at the common-mode voltage, V_{CM}). This is important to note in both the active-RC and switched-capacitor implementations.

We won't discuss the implementation of the MOSFET-C-based bilinear transfer function. It should be obvious that replacing the resistors in Fig. 35.30 with MOSFETs or linearized MOSFETs (see Figs. 35.12-35.14) provides a MOSFET-C implementation.

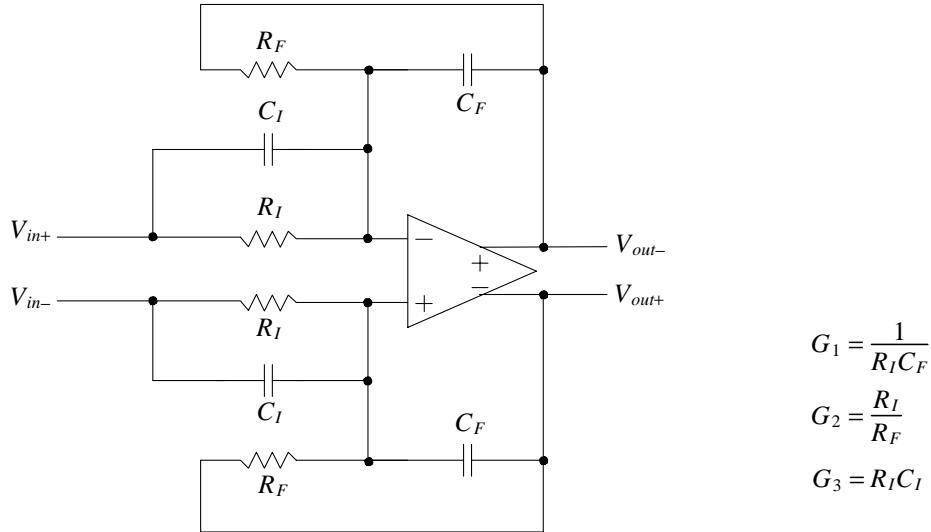


Figure 35.30 Implementation of an active-RC bilinear transfer function filter.

Transconductor-C Implementation

Figure 35.31 shows the implementation of the bilinear transfer function using transconductor stages. Again, as with the active-RC filter, signals are summed using currents. Summing the currents at the output nodes results in

$$g_{m1}(V_{in+} - V_{in-}) - g_{m2}(V_{out+} - V_{out-}) + \frac{V_{in+} - V_{out+}}{1/s2C_1} - \frac{V_{out+}}{1/s2C_2} = 0 \quad (35.56)$$

where we know $V_{out+} = -V_{out-}$ and $V_{in+} = -V_{in-}$. It will be helpful to write

$$\frac{V_{in+}}{1/s2C_1} = \frac{2V_{in+}}{1/sC_1} = \frac{V_{in+} - V_{in-}}{1/sC_1} \quad (35.57)$$

Using this expression, we can write Eq. (35.56) as

$$(V_{out+} - V_{out-}) \cdot (s(C_1 + C_2) + g_{m2}) = (V_{in+} - V_{in-}) \cdot (g_{m1} + sC_1) \quad (35.58)$$

or finally

$$\frac{V_{out+} - V_{out-}}{V_{in+} - V_{in-}} = \frac{g_{m1}}{g_{m2}} \cdot \frac{1 + \frac{s}{g_{m1}/C_1}}{1 + \frac{s}{g_{m2}/(C_1+C_2)}} \quad (35.59)$$

It's important to note that when looking at this equation the location of the pole and zero can be adjusted by changing each transconductor's g_m independently. The ability to adjust one variable in a filter's transfer function and only change the position of a single pole or zero is called *orthogonal tuning*.

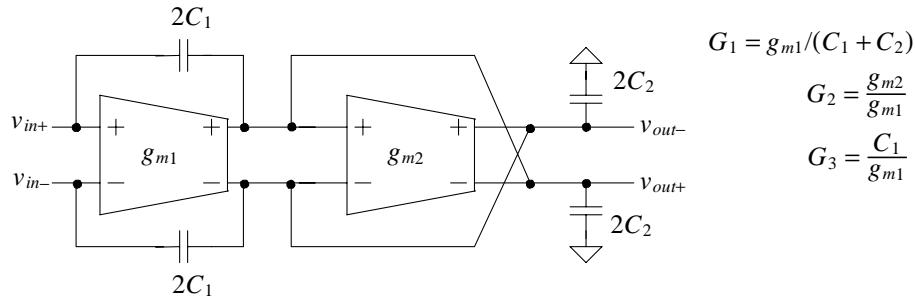


Figure 35.31 Implementation of a bilinear filter using transconductors.

Switched-Capacitor Implementation

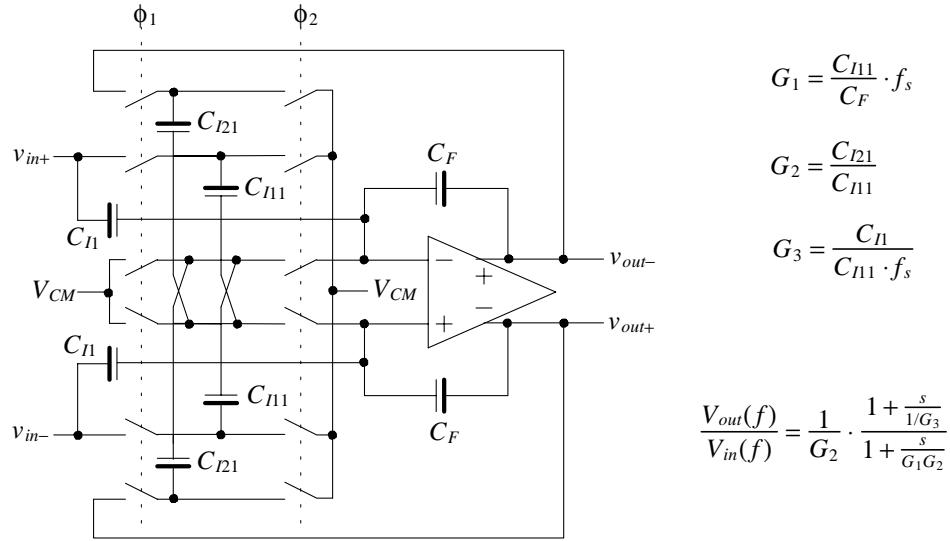
The switched-capacitor, SC, implementation of the bilinear filter is seen in Fig. 35.32. This filter is directly derived from the active-RC filter of Fig. 35.30. From Eq. (35.42) we can write

$$R_I = \frac{1}{C_{I1} \cdot f_s} \text{ and } R_F = \frac{1}{C_{I2} \cdot f_s} \quad (35.60)$$

and so

$$G_1 = \frac{C_{I1}}{C_F} \cdot f_s, G_2 = \frac{C_{I2}}{C_{I1}}, \text{ and } G_3 = \frac{C_{I3}}{C_{I1} \cdot f_s} \quad (35.61)$$

Note how, in this discrete-time filter, the passband gain is C_{I3}/C_F when the filter is designed for a highpass response (and the filter no longer behaves like a discrete-time filter). The gain at DC in all situations is C_{I1}/C_{I2} .

**Figure 35.32** Implementation of a bilinear filter using switched capacitors.*Digital Filter Implementation*

To implement the differentiation in Fig. 35.29, the circuit shown in Fig. 35.33 can be used (see also Figs. 31.50 and 31.51). The transfer function of the filter is

$$1 - z^{-1} = 1 - e^{-j2\pi\frac{f}{f_s}} = 1 - \cos 2\pi \cdot \frac{f}{f_s} + j \sin 2\pi \cdot \frac{f}{f_s} \quad (35.62)$$

Knowing from Fig. 35.28 that the digital filter functions as desired (with characteristics similar to an analog filter) when $f \ll f_s$, we can write (see Fig. 35.33)

$$G_{3D}(1 - z^{-1}) \approx \underbrace{\frac{G_{3D}}{f_s}}_{G_3} \cdot s \quad (35.63)$$

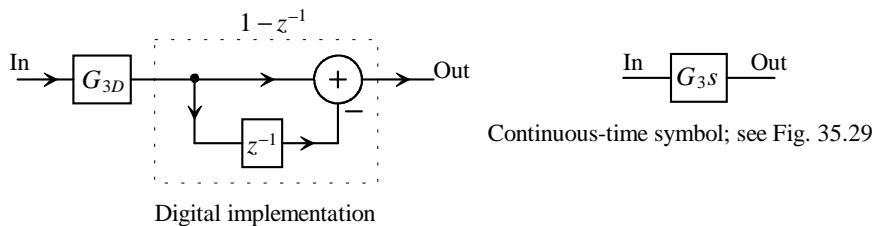
**Figure 35.33** Implementing a digital differentiator.

Figure 35.34 shows the digital implementation of the bilinear filter of Fig. 35.29. It's important, at this point, to see how the continuous-time implementation in Fig. 35.29 is directly implemented in Fig. 35.34. In particular we note that

$$G_1 = G_{1D} \cdot f_s \text{ and } G_3 = \frac{G_{3D}}{f_s} \quad (35.64)$$

and so

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{1}{G_2} \cdot \frac{1 + \frac{s}{1/G_3}}{1 + \frac{s}{G_1 G_2}} = \frac{1}{G_2} \cdot \frac{1 + j \cdot \frac{f}{f_s/(2\pi G_{3D})}}{1 + j \cdot \frac{f}{f_s G_{1D} G_2 / 2\pi}} \quad (35.65)$$

The location of the pole is given by

$$f_{3dB, pole} = \frac{f_s G_{1D} G_2}{2\pi} \quad (35.66)$$

while the location of the zero is at

$$f_{3dB, zero} = \frac{f_s}{2\pi G_{3D}} \quad (35.67)$$

Note in order for our filter to be useful where the frequencies of interest are much less than the filter's clocking frequency, f_s , the pole and zero location must be much less than f_s . This means, assuming $G_2 = 1$, that $G_{1D} \ll 1$ and $G_{3D} \gg 1$.

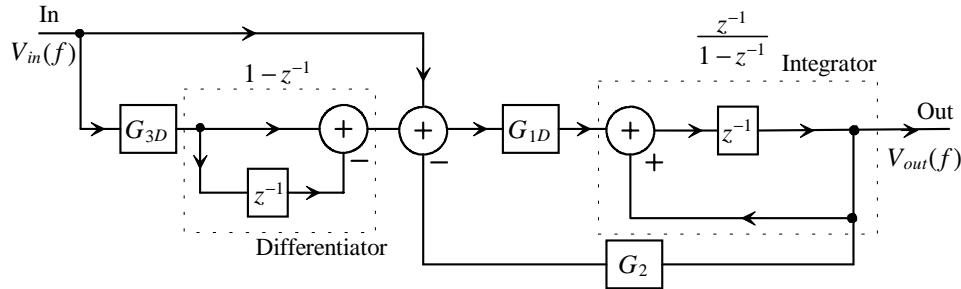


Figure 35.34 Digital implementation of the bilinear transfer function.

Let's attempt to simplify this filter. If we look at the transfer function, Eq. (35.65), we see that the feedback gain, G_2 , simply scales the amplitude of the transfer function and can be used to further adjust the pole of the filter. Because we can scale the amplitude of the signal either before or after the filter, and independent of the filter's operation, and we can precisely set the pole of the filter using G_{1D} , we can, without loss of functionality, set G_2 to 1. We can then rearrange the summing, delaying, and multiplying blocks, as seen in Fig. 35.35.

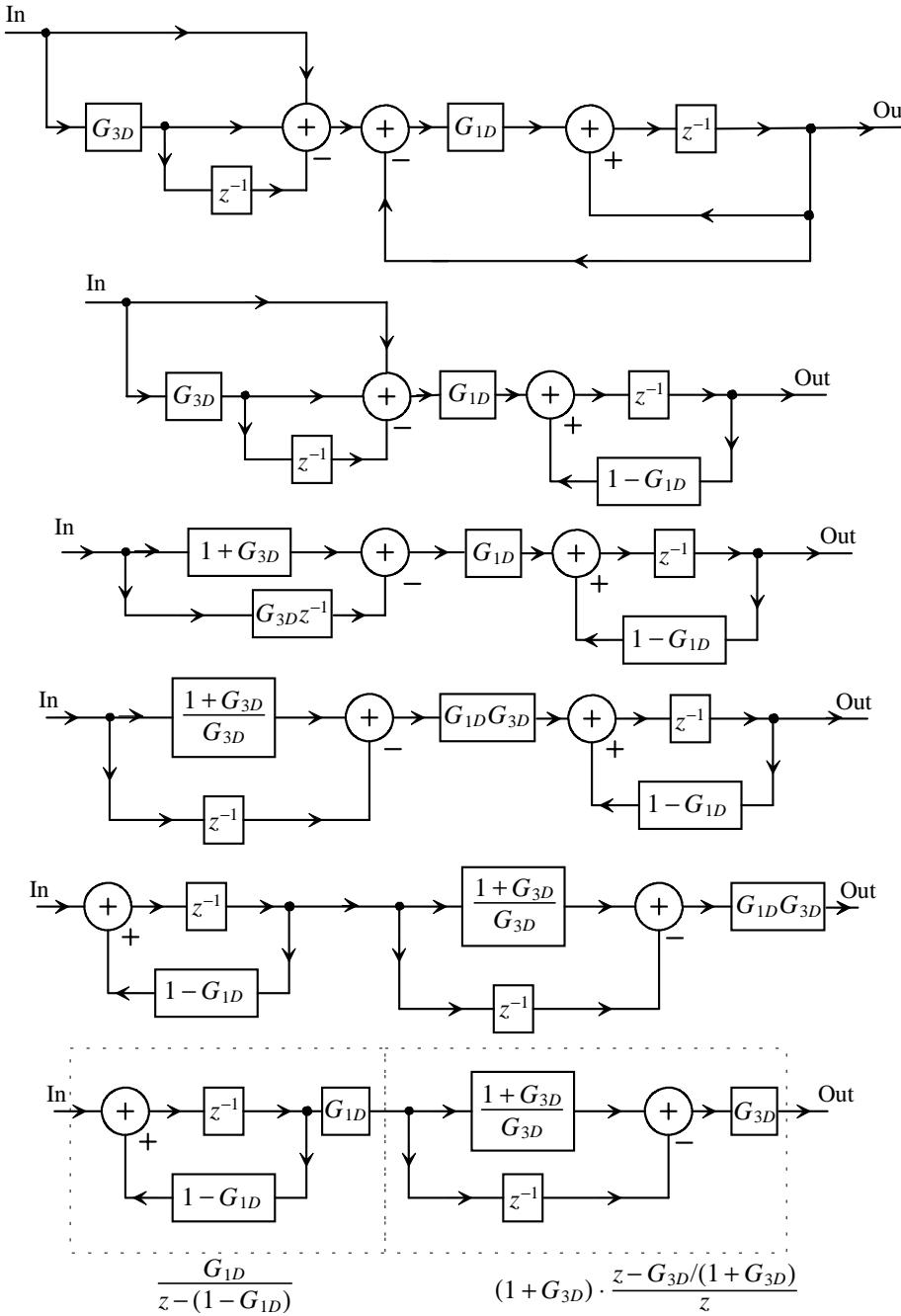


Figure 35.35 Simplifying the digital implementation of the bilinear filter.

Using the results of Fig. 35.35 we can write the z-domain representation of the transfer function, Eq. (35.65), as

$$\frac{V_{out}(z)}{V_{in}(z)} = \frac{G_{1D}(1+G_{3D})}{z-(1-G_{1D})} \cdot \frac{z-G_{3D}/(1+G_{3D})}{z} \quad (35.68)$$

Note how, for a lowpass filter with G_{3D} set to 0, this equation reduces to Eq. (35.48) with $A = G_{1D}$. Before attempting to simplify the filter implementation seen in Fig. 35.35 further, let's show that, indeed, Eq. (35.68) is equivalent to Eq. (35.65) when $f \ll f_s$. It will be helpful to remember that

$$z \approx 1 + \frac{s}{f_s} \text{ and } z^{-1} \approx 1 - \frac{s}{f_s} \text{ if } f \ll f_s \text{ or } z \approx 1 + \frac{s}{f_s} \approx \frac{1}{1 - \frac{s}{f_s}} \approx \frac{1}{z^{-1}} \text{ if } \frac{s^2}{f_s^2} \approx 0 \quad (35.69)$$

where $s = j\omega = j2\pi f$. Rewriting Eq. (35.68) in the frequency-domain gives

$$\frac{V_{out}(f)}{V_{in}(f)} = \frac{G_{1D}(1+G_{3D})}{1 + \frac{s}{f_s} - 1 + G_{1D}} \cdot \left[1 - \left(1 - \frac{s}{f_s} \right) G_{3D}/(1+G_{3D}) \right] \quad (35.70)$$

$$= \frac{1 + \frac{s}{f_s/G_{3D}}}{1 + \frac{s}{G_{1D}f_s}} = \frac{1 + j \cdot \frac{f}{f_s/(2\pi G_{3D})}}{1 + j \cdot \frac{f}{G_{1D}f_s/2\pi}} \quad (35.71)$$

which is clearly the same as Eq. (35.65) when $G_2 = 1$.

Example 35.8

Sketch the digital filter equivalent of the following RC circuit. Assume the digital filter is clocked at 100 MHz.

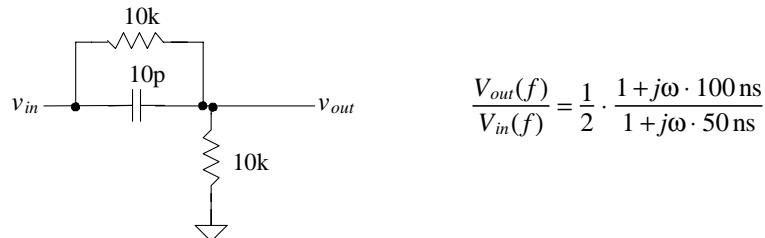
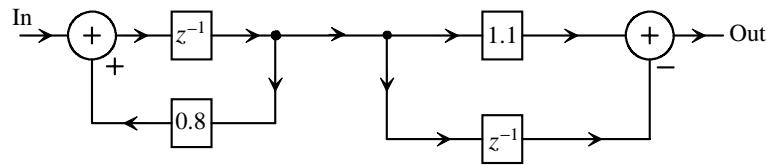
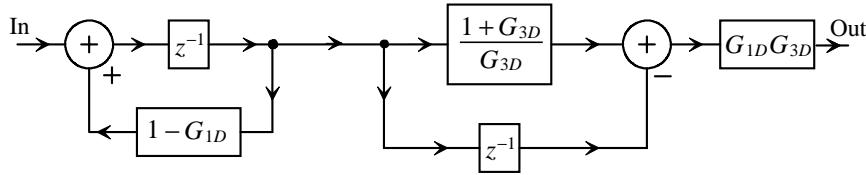


Figure 35.36 A simple first-order RC circuit.

Comparing the transfer function in Fig. 35.36 to Eq. (35.71), we see that if

$$\frac{G_{3D}}{f_s} = 100 \text{ ns} \rightarrow G_{3D} = 10 \text{ and } \frac{1}{G_{1D} \cdot f_s} = 50 \text{ ns} \rightarrow G_{1D} = 0.2$$

then we can use any of the filters in Fig. 35.35. The sketch of the digital filter is seen in Fig. 35.37. The multiplication of the transfer function by 1/2 is nulled by the multiplication by $G_{1D}G_{3D}$ (= 2). To verify that the filter in Fig. 35.37 functions as desired at DC, we see that the output of the first stage is 0.5 when the input is 0.1, and the output of the second stage is 0.05 (with a 0.5 on its input). ■



$$H(z) = 1.1 \cdot \frac{z - 10/11}{z(z - 0.8)} = 1.1 \cdot \frac{z^{-1}}{1 - 0.8z^{-1}} \cdot (1 - z^{-1} \cdot 10/11)$$

Figure 35.37 Digital filter from Ex. 35.8.

The Canonic Form (or Standard Form) of a Digital Filter

Studying Fig. 35.37, we might wonder if we can further reduce the size of the digital filter. We see in this figure that it may be possible to eliminate the second delay element (which, of course, is a register) and use only a single delay. The result of this modification is seen in Fig. 35.38. Intuitively we would think that the phase response of the filter will change because, now, there is less delay in series with input of the second adder. We can write the output as

$$\frac{V_{out}(z)}{V_{in}(z)} = G_{1D}G_{3D} \left[\frac{1 + G_{3D}}{G_{3D}} \cdot \frac{1}{1 - z^{-1}(1 - G_{1D})} - \frac{z^{-1}}{1 - z^{-1}(1 - G_{1D})} \right] \quad (35.72)$$

or

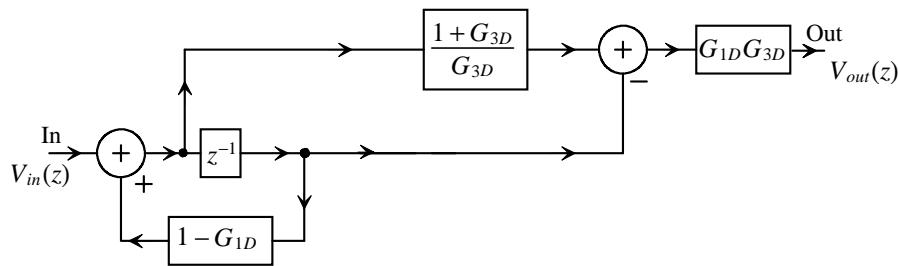


Figure 35.38 Canonic form of a first-order digital filter.

$$\frac{V_{out}(z)}{V_{in}(z)} = G_{1D}(1 + G_{3D}) \cdot \frac{z - G_{3D}/(1 + G_{3D})}{z - (1 - G_{1D})} \quad (35.73)$$

which is clearly the same response as derived in Fig. 35.35 or Eq. (35.68) except that the output is one clock cycle, z , earlier (remembering to delay a signal by one clock cycle we simply multiply it by z^{-1} [latch it into a register for one clock cycle]). This reduced delay, of course, has no effect on the magnitude response of the filter and little effect, assuming $f \ll f_s$, on the phase response of the filter.

The general form of this first-order canonic (or standard form) filter is seen in Fig. 35.39. The filter is termed *canonic* because the minimum number of delays are used. One delay is used for each pole (remembering from Ch. 31 that in order for a digital filter to be realizable in hardware there must be fewer or an equal number of zeroes than poles in a filter's transfer function [see page 101]).

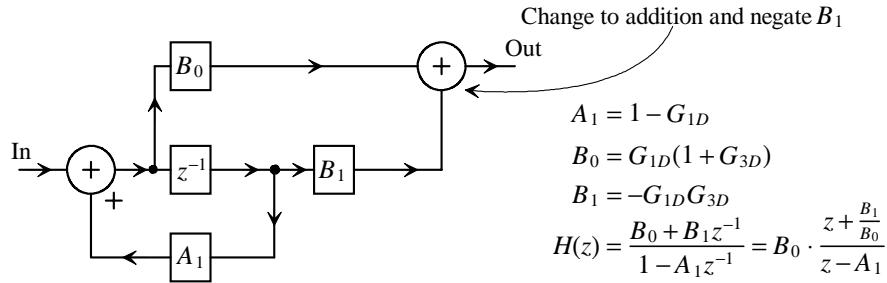


Figure 35.39 General canonic form of a first-order digital filter.

We can, again, derive the transfer function for the first-order bilinear digital filter. This time, however, let's use the variables in Fig. 35.39. Again, assuming $f \ll f_s$, and using Eq. (35.69) results in

$$H(f) = \frac{B_0 + B_1}{1 - A_1} \cdot \frac{1 + \frac{s}{f_s(1 + \frac{B_1}{B_0})}}{1 + \frac{s}{f_s(1 - A_1)}} \quad (35.74)$$

where

$$f_{3dB,pole} = \frac{f_s(1 - A_1)}{2\pi} \quad (35.75)$$

and

$$f_{3dB,zero} = \frac{f_s}{2\pi} \left(1 + \frac{B_1}{B_0} \right) \quad (35.76)$$

and the gain at DC is

$$A_{DC} = \frac{B_0 + B_1}{1 - A_1} \quad (35.77)$$

Example 35.9

Using the canonic form of the first-order digital filter, repeat Ex. 35.8.

Comparing Eq. (35.74) with the transfer function in Fig. 35.36, we can write

$$2\pi f_{3dB,pole} = \frac{1}{50 \text{ ns}} = f_s \cdot (1 - A_1) = 100 \text{ MHz}(1 - A_1) \rightarrow A_1 = 0.8$$

$$A_{DC} = \frac{1}{2} = \frac{B_0 + B_1}{1 - A_1} = \frac{B_0 + B_1}{1 - 0.8} \rightarrow B_0 + B_1 = 0.1$$

$$2\pi f_{3dB,zero} = \frac{1}{100 \text{ ns}} = f_s \cdot \left(1 + \frac{B_1}{B_0}\right) = 100 \text{ MHz} \left(\frac{0.1}{B_0}\right) \rightarrow B_0 = 1$$

and thus $B_1 = -0.9$. The filter's sketch is seen in Fig. 35.40. We will discuss how to implement the multipliers in the final section of the chapter.

Let's do a quick check to see if the filter functions as desired at DC. If we apply 0.1 to the input of the filter then, according to the transfer function in Fig. 35.36, the output of the filter should be 0.05 or one-half the input. Because the input to the filter is a DC signal, both sides of the delay will have the same value. According to Fig. 31.62, this value will be 0.5 (the output of the weighted integrator is $1/[1 - 0.8]$ times the input signal, here 0.1, at DC). The output will then be $0.5 - 0.45$ or 0.05 (as we would expect). ■

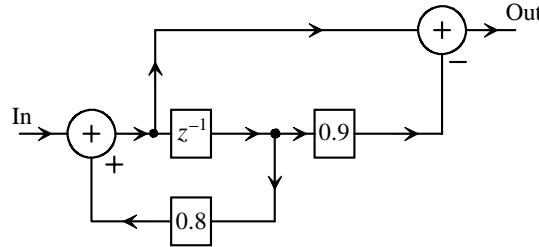


Figure 35.40 Canonic form of the first-order digital filter in Ex. 35.8.

Example 35.10

Sketch the digital filter implementation of the lowpass filter in Ex. 35.1 that has a DC gain of one and a 3 dB frequency of 1.59 MHz. Assume the filter is clocked at 100 MHz.

The filter's continuous-time frequency response is given by

$$H(f) = \frac{1}{1 + j \cdot \frac{f}{1.59 \text{ MHz}}}$$

Using Eqs. (35.74) to (35.77), we begin by calculating A_1

$$f_{3dB, pole} = 1.59 \text{ MHz} = \frac{f_s(1 - A_1)}{2\pi} = \frac{100 \text{ MHz}}{2\pi}(1 - A_1) \rightarrow A_1 = 0.9$$

and then

$$A_{DC} = \frac{B_0 + B_1}{1 - A_1} = 1 = \frac{B_0 + B_1}{1 - 0.9} \rightarrow B_0 + B_1 = 0.1$$

Let's put the zero at infinity so it doesn't affect the transfer function

$$f_{3dB, zero} = \infty = \frac{f_s}{2\pi} \left(1 + \frac{B_1}{B_0} \right) \rightarrow B_0 = 0 \text{ and } B_1 = 0.1$$

A sketch of the filter is seen in Fig. 35.41. Note that this is the exact same filter as the one seen in Fig. 35.26 except that we have combined multipliers so $B_1 = A$ where the A is seen in Fig. 35.26. ■

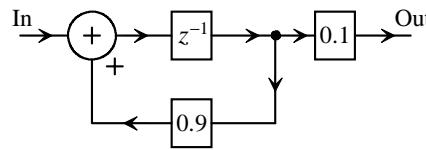


Figure 35.41 First-order digital filter in Ex. 35.10.

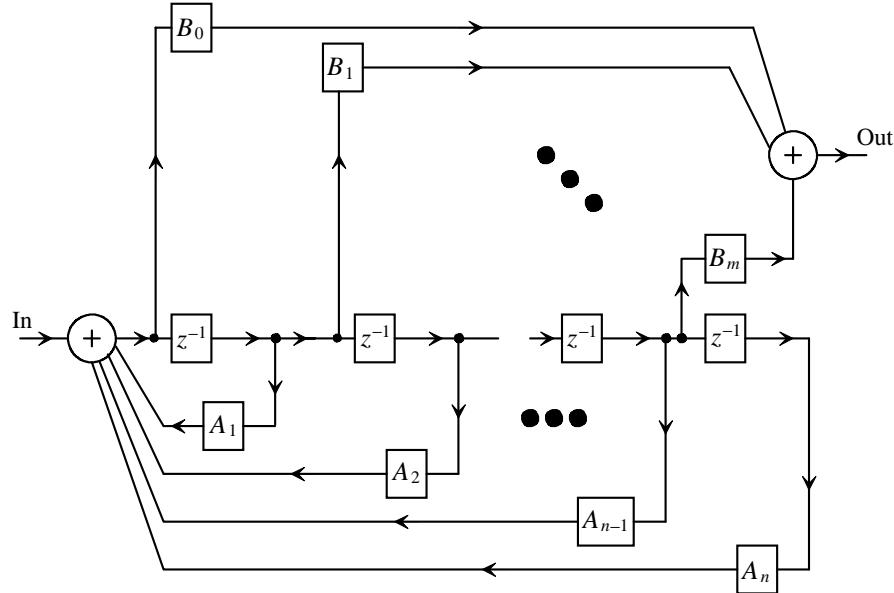
Before leaving this section, let's show in Fig. 35.42 the general form of an n^{th} -order canonic digital filter (where n indicates the number of poles in the transfer function). The z-domain transfer function of the filter is given by

$$H(z) = \frac{\sum_{i=0}^m B_i z^{-i}}{1 - \sum_{i=1}^n A_i z^{-i}} = \frac{\sum_{i=0}^m B_i z^{n-i}}{z^n - \sum_{i=1}^n A_i z^{n-i}} \quad (35.78)$$

If we want to write the frequency domain transfer function, we write, again assuming $f \ll f_s$ and using Eq. (35.69),

$$H(f) \approx \frac{\sum_{i=0}^{n-1} B_i \left(1 - j \frac{2\pi f}{f_s} \right)^i}{1 - \sum_{i=1}^n A_i \left(1 - j \frac{2\pi f}{f_s} \right)^i} \approx \frac{\sum_{i=0}^{n-1} B_i \left(1 + j \frac{2\pi f}{f_s} \right)^{n-i}}{\left(1 + j \frac{2\pi f}{f_s} \right)^n - \sum_{i=1}^n A_i \left(1 + j \frac{2\pi f}{f_s} \right)^{n-i}} \quad (35.79)$$

While we can design higher order digital filters using the topology of Fig. 35.42, we will restrict our analysis in the remainder of the book to first- and second-order filters where hand calculations are relatively easy to do. We can increase the attenuation of a filter using several of these sections, either cascading the stages in series or taking the outputs of several first- or second-order sections and adding them together.



Number of poles $n \geq$ number of zeroes.

Figure 35.42 General canonic form of a digital filter.

35.2.2 The Biquadratic Transfer Function

As we briefly indicated in the last section, higher order filters can be implemented by cascading first-order sections. However, because the pole and zero locations in these first-order filters are restricted to real values, the performance of these cascades is poorer than filters with complex pole and zero locations. For example, cascading two identical lowpass filters having f_{3dB} frequencies of 1.59 MHz would result in a filter that has an attenuation of 6 dB at 1.59 MHz and a -40 dB/decade roll off at higher frequencies. Using a second-order filter, we can design a filter to have a sharper transition at 1.59 MHz so that the attenuation is less than 6 dB at 1.59 MHz (however, the roll off remains -40 dB/decade). Further, we can use these sections to implement higher order filters using Butterworth, Chebyshev, Elliptic (Cauer), or Bessel responses [9].

The biquadratic, or "biquad" for short, filter transfer function (a ratio of two quadratic equations) is given by

$$\frac{V_{out}}{V_{in}} = \frac{a_2 s^2 + a_1 s + a_0}{s^2 + \left(\frac{2\pi f_0}{Q}\right)s + (2\pi f_0)^2} \quad (35.80)$$

where $2\pi f_0 = \omega_0$. The complex-conjugate poles are located at

$$p_1, p_2 = \frac{\pi f_0}{Q} \pm \frac{1}{2} \sqrt{\left(\frac{2\pi f_0}{Q}\right)^2 - 4(2\pi f_0)^2} \quad (35.81)$$

or

$$p_1, p_2 = \frac{\pi f_0}{Q} \pm j \cdot 2\pi f_0 \sqrt{1 - \left(\frac{1}{2Q}\right)^2} \quad (35.82)$$

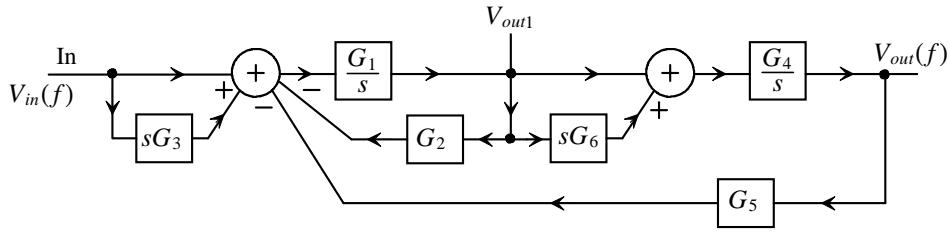


Figure 35.43 Implementation of a biquadratic transfer function using two integrators.

Toward the goal of implementing a biquad, consider the block diagram in Fig. 35.43. This block diagram is essentially the cascade of two first-order blocks (as seen in Fig. 35.29) except that instead of feeding the output of the second stage back to its input, we feed it back to the input of the first stage. We can determine the transfer function of this filter by writing

$$V_{out1} = (V_{in} + sG_3 V_{in} - G_5 V_{out} - G_2 V_{out1}) \cdot \frac{G_1}{s} \quad (35.83)$$

or

$$V_{out1} \left(\frac{s + G_1 G_2}{s} \right) = V_{in} \left(\frac{(1 + sG_3)G_1}{s} \right) - V_{out} \left(\frac{G_1 G_5}{s} \right) \quad (35.84)$$

Further, we can relate V_{out1} to the output using

$$V_{out} = V_{out1} (1 + sG_6) \cdot \frac{G_4}{s} \quad (35.85)$$

Using Eq. (35.84) with Eq. (35.85) gives

$$V_{out} = \left(V_{in} \cdot \frac{(1 + sG_3)G_1}{s + G_1 G_2} - V_{out} \frac{G_1 G_5}{s + G_1 G_2} \right) \cdot (1 + sG_6) \frac{G_4}{s} \quad (35.86)$$

or

$$V_{out} \left(1 + \frac{G_1 G_4 G_5 (1 + sG_6)}{s^2 + sG_1 G_2} \right) = V_{in} \left(\frac{(1 + sG_3)G_1 (1 + sG_6)G_4}{s^2 + sG_1 G_2} \right) \quad (35.87)$$

Finally, the transfer function of the biquad is given by

$$\frac{V_{out}}{V_{in}} = \frac{s^2 G_1 G_3 G_4 G_6 + s(G_1 G_3 G_4 + G_1 G_4 G_6) + G_1 G_4}{s^2 + s(G_1 G_2 + G_1 G_4 G_5 G_6) + G_1 G_4 G_5} \quad (35.88)$$

Equating terms in Eqs. (35.80) and (35.88) gives

$$a_2 = G_1 G_3 G_4 G_6 \quad (35.89)$$

$$a_1 = G_1 G_3 G_4 + G_1 G_4 G_6 \quad (35.90)$$

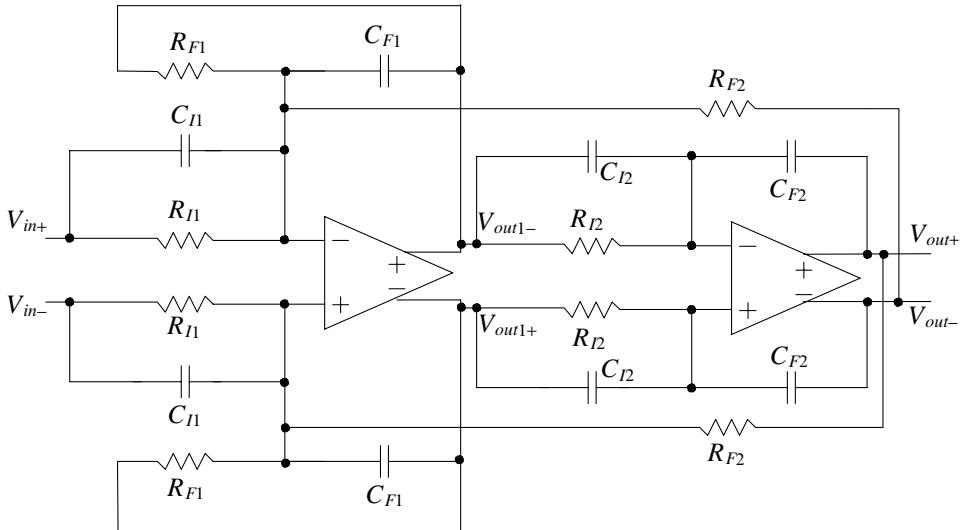
$$a_0 = G_1 G_4 \quad (35.91)$$

$$\frac{2\pi f_0}{Q} = G_1 G_2 + G_1 G_4 G_5 G_6 \quad (35.92)$$

$$(2\pi f_0)^2 = G_1 G_4 G_5 \quad (35.93)$$

Active-RC Implementation

Figure 35.44 shows the active-RC implementation of the biquad filter along with the associated design equations. It should be noted that this is the general design schematic. If, for example, a lowpass filter needs to be implemented, the filter can be greatly simplified.



$$G_1 = \frac{1}{R_{I1}C_{F1}} \quad G_2 = \frac{R_{I1}}{R_{F1}} \quad G_3 = R_{I1}C_{I1} \quad G_4 = \frac{1}{R_{I2}C_{F2}} \quad G_5 = \frac{R_{I1}}{R_{F2}} \quad G_6 = R_{I2}C_{I2}$$

$$a_2 = \frac{C_{I1}C_{I2}}{C_{F1}C_{F2}} \quad a_1 = \frac{C_{I1}}{R_{I2}C_{F1}C_{F2}} + \frac{C_{I2}}{R_{I1}C_{F1}C_{F2}} \quad a_0 = \frac{1}{R_{I1}C_{F1}R_{I2}C_{F2}}$$

$$\frac{\omega_0}{Q} = \frac{2\pi f_0}{Q} = \frac{1}{R_{F1}C_{F1}} + \frac{C_{I2}}{C_{F1}R_{F2}C_{F2}} \quad f_0 = \frac{1}{2\pi} \cdot \sqrt{\frac{1}{C_{F1}R_{I2}C_{F2}R_{F2}}}$$

Figure 35.44 Implementation of the active-RC biquadratic transfer function filter.

Figure 35.45 shows the frequency response, pole-zero locations in the s-plane, and transfer function for a second-order lowpass circuit made using an inductor (L), capacitor (C), and resistor (R). This LRC circuit has the same frequency response shape as a lowpass biquad filter. However, the DC gain of the LRC circuit must be unity while the DC gain of the biquad filter can be set to $a_0/(2\pi f_0)^2$. Note that if the pole quality factor, Q , is greater than $1/\sqrt{2}$ the response will show peaking. Setting Q to 0.707 results in the Butterworth or maximally flat response.

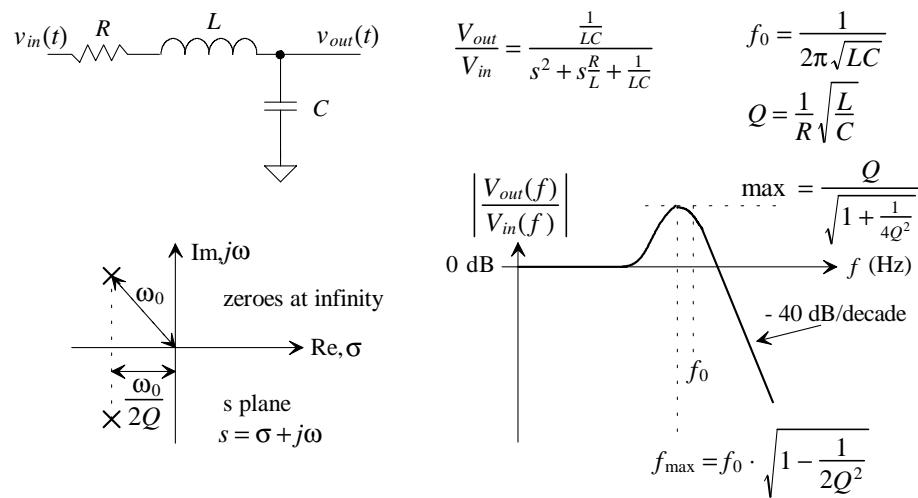


Figure 35.45 Second-order lowpass filter.

Example 35.11

Design an LRC circuit with a Q of 0.707 and a cutoff frequency (f_0) of 1.59 MHz.

From Fig. 35.45 we have two equations we need to solve

$$f_0 = \frac{1}{2\pi\sqrt{LC}} = 1.59 \text{ MHz} \rightarrow LC = 10 \times 10^{-15} \text{ and } Q = \frac{1}{\sqrt{2}} = \frac{1}{R\sqrt{C}}$$

We can set $C = 100 \text{ pF}$, then $L = 100 \mu\text{H}$, and $R = 1.414\text{k}$ (definitely not practical values if the circuit is going to be purely integrated). The response of the resulting LRC circuit is shown in Fig. 35.46. Note how the cutoff frequency is set by the inductor and capacitor, while the Q of the circuit is set by all three elements (variations in the resistance having the largest effect on the circuit's Q). Higher Q indicates the poles are moving toward the imaginary axis (keeping in mind that a system with right-hand plane poles is unstable [oscillates]) and more peaking, Fig. 35.47. ■

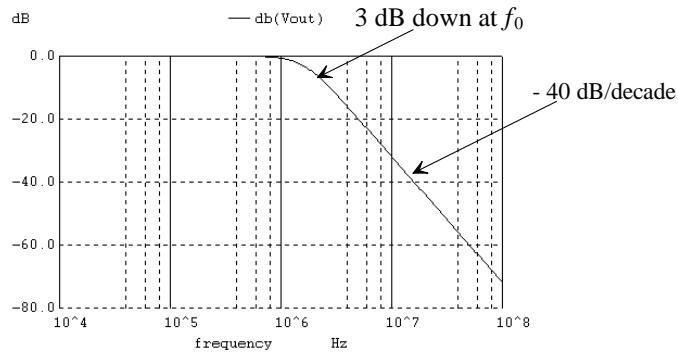


Figure 35.46 Second-order magnitude response for the circuit described in Ex. 35.11.

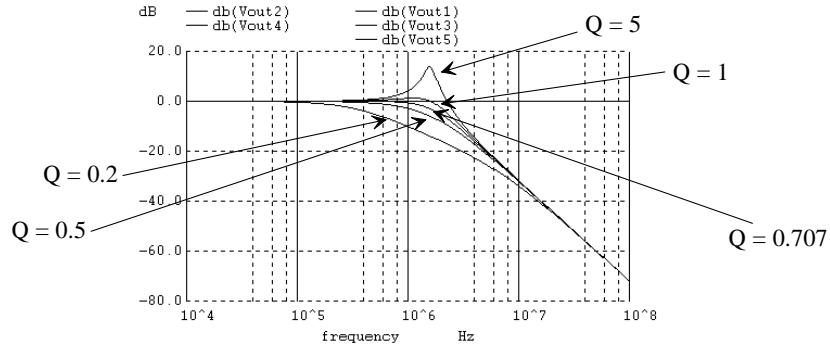


Figure 35.47 The effect of Q on the frequency response of a second-order lowpass filter.

Example 35.12

Simulate the design of an active-RC filter that has frequency characteristics similar to Fig. 35.46.

Using the basic topology of Fig. 35.44, we see that for a lowpass filter, $C_{I1} = C_{I2} = 0$ and therefore $G_3 = G_6 = 0$. Further

$$f_0 = 1.59 \text{ MHz} = \frac{1}{2\pi} \sqrt{\frac{1}{C_{F1}R_{I2}C_{F2}R_{F2}}}$$

which we shall use to set $R_{I2} = R_{F2} = 10k$ and $C_{F1} = C_{F2} = 10 \text{ pF}$. The Q of the filter is given by

$$Q = \frac{1}{\sqrt{2}} = 2\pi f_0 \cdot R_{F1} C_{F1} \rightarrow R_{F1} = 7.07 k\Omega$$

Knowing $a_2 = a_1 = 0$, the gain at DC is $a_0/(2\pi f_0)^2$ or R_{F2}/R_{I1} ($1/G_5$), which is 1 here. The simulation results are shown in Fig. 35.48. ■

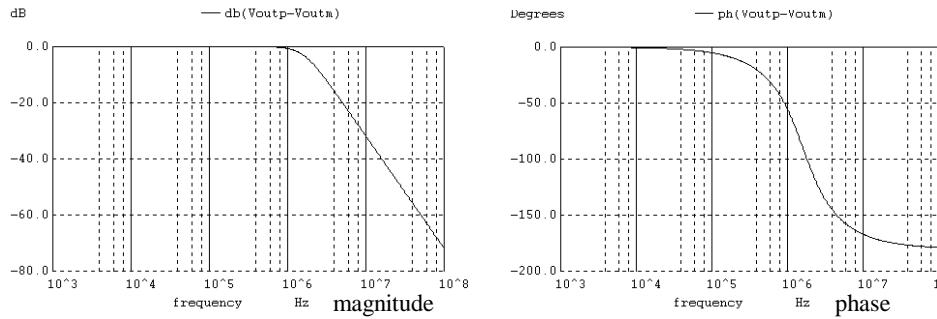


Figure 35.48 Magnitude and phase responses for the active-RC filter of Ex. 35.12.

Notice that at DC, when used in the lowpass configuration, the outputs of the first integrator in Fig. 35.44, V_{out1+} and V_{out1-} , must be equal. If not, the difference is integrated by the second section. As the frequency increases, so does the difference in these voltage levels.

Figure 35.49 shows the second-order bandpass response. Again, as with the second-order lowpass response, the center frequency (resonant frequency) is set by the values of the inductor and capacitor. The Q of the filter indicates how narrow the bandpass response is; higher Q indicates a narrower response. Note how the response eventually rolls off at -20 dB/decade. At low frequencies the capacitor can be thought of as an open (resulting in a first-order RL circuit response), while at high frequencies the inductor can be thought of as an open (resulting in a first-order RC circuit response).

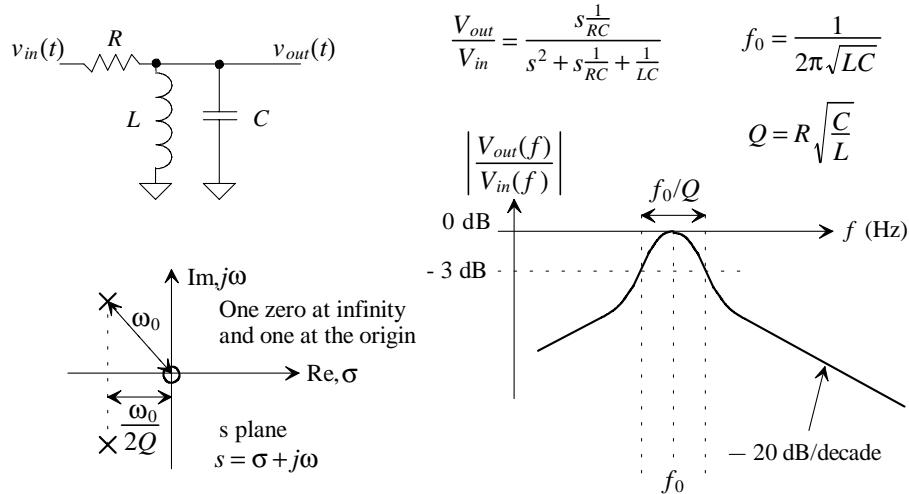


Figure 35.49 Second-order bandpass filter.

Example 35.13

Repeat Ex. 35.11 for the bandpass LRC circuit.

Again, we can set $C = 100 \text{ pF}$ and $L = 100 \mu\text{H}$. Solving for Q using the equation in Fig. 35.49 results in

$$Q = R \sqrt{\frac{C}{L}} = 0.707 = R \sqrt{\frac{100p}{100\mu}} \rightarrow R = 707$$

The simulation results are seen in Fig. 35.50. ■

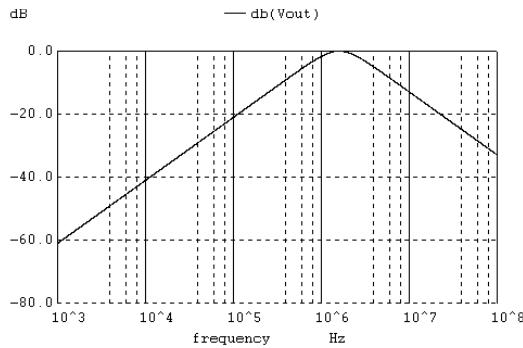


Figure 35.50 Bandpass response of a second-order circuit with a Q of 0.707.

Example 35.14

Repeat Ex. 35.13 if the Q is increased to 20.

Figure 35.51 shows the simulation results. To attain a Q of 20, we use a resistor of 20k with the inductor and capacitor values remaining unchanged. ■

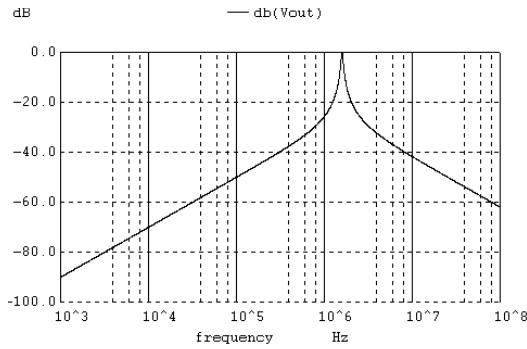


Figure 35.51 Bandpass response of a second-order circuit with a Q of 20.

Example 35.15

Use an active-RC filter to implement a filter with the response shown in Fig. 35.51.

Let's begin by writing the filter's transfer function

$$\frac{V_{out}}{V_{in}} = \frac{a_1 s}{s^2 + \left(\frac{2\pi f_0}{Q}\right)s + (2\pi f_0)^2} = \frac{a_1 s}{s^2 + \left(\frac{10 \times 10^6}{20}\right)s + (10 \times 10^6)^2}$$

Looking at this equation, Eq. (35.80), and Fig. 35.44 we see that $a_2 = a_0 = 0$ and so $C_{I2} = 0$ and $R_{I1} = \infty$. Further then

$$a_1 = \frac{C_{I1}}{R_{I2} C_{F1} C_{F2}} \quad f_0 = \frac{1}{2\pi} \sqrt{\frac{1}{C_{F1} R_{I2} C_{F2} R_{F2}}} = 1.59 \text{ MHz}$$

$$\frac{2\pi f_0}{Q} = \frac{2\pi \cdot 1.59 \text{ MHz}}{20} = \frac{1}{R_{F1} C_{F1}}$$

The passband gain (the maximum gain) occurs at f_0 and is calculated by replacing s in the transfer function above with $j2\pi f_0$. It is given by

$$A_{passband} = \frac{a_1 Q}{2\pi f_0}$$

If, again, we set $C_{F1} = C_{F2} = 10 \text{ pF}$ and $R_{I2} = R_{F2} = 10\text{k}$, we get an f_0 of 1.59 MHz. Further then, with a Q of 20, we can set R_{F1} to 200k. Finally, setting the passband gain to unity results in

$$a_1 = \frac{2\pi f_0}{Q} = \frac{C_{I1}}{R_{I2} C_{F1} C_{F2}} \rightarrow C_{I1} = 0.5 \text{ pF}$$

While these values do result in a biquad with the shape seen in Fig. 35.51, the values are not practical. Redoing the calculations while trying to minimize the component spread gives another possible solution: $R_{I2} = 100\text{k}$, $C_{F1} = 20\text{p}$, $R_{F1} = 100\text{k}$, $C_{F2} = 5\text{p}$, $C_{I1} = 5\text{p}$, and $R_{F2} = 1\text{k}$. The simulation results are seen in Fig. 35.52. ■

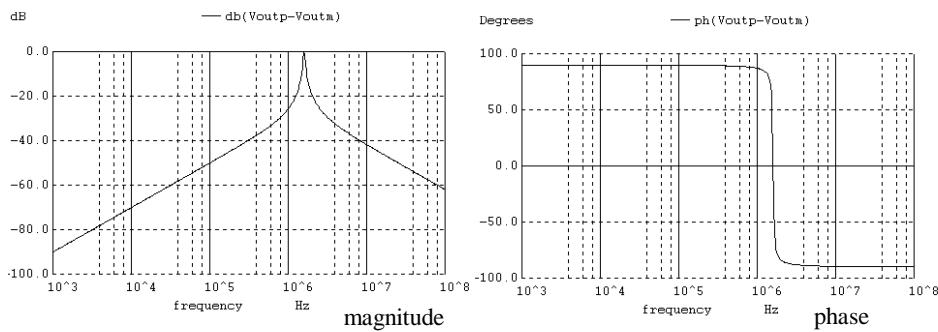


Figure 35.52 Outputs of the biquad of Ex. 35.15 using active-RC elements.

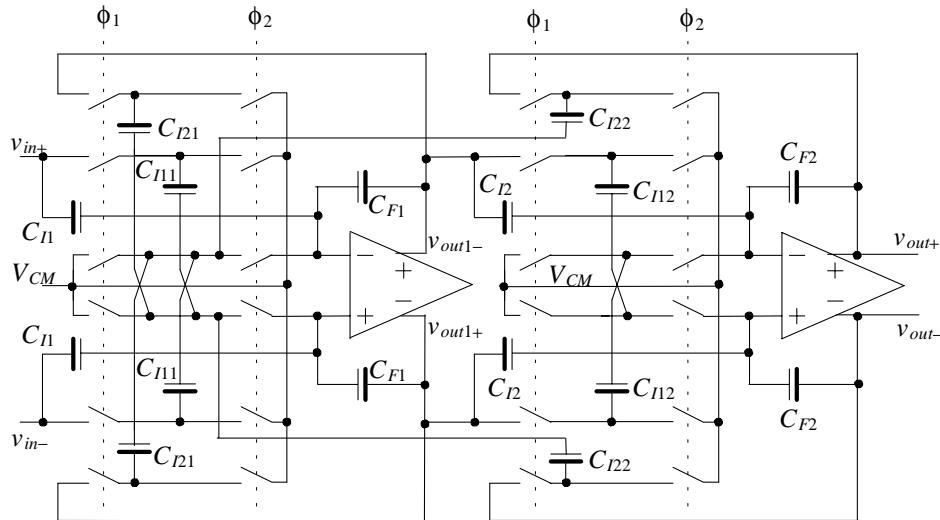
Switched-Capacitor Implementation

The switched-capacitor implementation of the biquad circuit is shown in Fig. 35.53. Note how this circuit is a simple translation of the active-RC circuit of Fig. 35.44. Again, if the filter designed using this section has a lowpass or bandpass response, it can be simplified. For example, from Figs. 35.45 and 35.49 (the implementation of lowpass and bandpass filters), we see that a_2 is zero. This indicates that G_6 can be set to zero (removing C_{I2} in Figs. 35.44 or 35.53). The resulting second-order filter response can be written as

$$\frac{V_{out}}{V_{in}} = \frac{a_1 s + a_0}{s^2 + \left(\frac{2\pi f_0}{Q}\right)s + (2\pi f_0)} = \frac{sG_1 G_3 G_4 + G_1 G_4}{s^2 + sG_1 G_2 + G_1 G_4 G_5} \quad (35.94)$$

Technically, the filter is no longer biquadratic, so we will refer to it as a second-order filter.

The biquad in Fig. 35.53 can look confusing until we start to dissect it. If we understand the topology of Fig. 35.32 we see that the switched-capacitor biquad is nothing more than two bilinear filters connected in cascade. The only difference is that the switched capacitance, C_{I22} , is fed back to the input of the first op-amp to simulate the feedback resistance, R_{F2} , in Fig. 35.44. This circuit, for the general lowpass or bandpass filter implementation, gets much simpler when the unused components are removed.



$$G_1 = \frac{C_{I11}}{C_{F1}} \cdot f_s \quad G_2 = \frac{C_{I21}}{C_{I11}} \quad G_3 = \frac{C_{I1}}{C_{I11} \cdot f_s} \quad G_4 = \frac{C_{I12}}{C_{F2}} \cdot f_s \quad G_5 = \frac{C_{I22}}{C_{I11}} \quad G_6 = \frac{C_{I2}}{C_{I12} \cdot f_s}$$

Figure 35.53 Implementing a biquad filter using switched capacitors.

High Q

We have a major concern alluded to in Ex. 35.15 when using either of the topologies of Fig. 35.44 or 35.53 with a large Q. As we saw in this example the capacitor values were within a factor of 4 of each other (20p and 5p) but the resistors used were two orders of magnitude different (100k and 1k). This large difference can be traced to, again assuming $G_6 = 0$,

$$\frac{2\pi f_0}{Q} = G_1 G_2 \text{ and } 2\pi f_0 = \sqrt{G_1 G_4 G_5} \text{ or } Q = \frac{\sqrt{G_4 G_5}}{\sqrt{G_1} G_2} = \sqrt{\frac{C_{F1}}{R_{F1} C_{F2} R_{F2}}} \cdot R_{F1} \quad (35.95)$$

This equation shows that R_{F1} has the largest direct dependence on Q. Using a large value of R_{F1} results in a smaller feedback signal (a smaller amount of current is fed back to the input of the first op-amp). In other words G_2 in Fig. 35.43 is small.

In order to minimize the amount of signal, V_{out1} , fed back and summed with the input signal, while at the same time forcing the components to have similar values, consider the modified, from Fig. 35.43, biquad block diagram shown in Fig. 35.54. All we have done here is added a separate signal path in parallel with the G_2 path. Instead of subtracting, though, we are now adding the signal to the input summing block. Equation (35.88) can be rewritten, assuming G_6 is zero (a bandpass or lowpass response), as

$$\frac{V_{out}}{V_{in}} = \frac{s(G_1 G_3 G_4) + G_1 G_4}{s^2 + sG_1(G_2 - G_{2Q}) + G_1 G_4 G_5} \quad (35.96)$$

or, equating the coefficient of s in the denominator of this equation with the coefficient of s in the denominator of Eq. (35.80), results in

$$\frac{2\pi f_0}{Q} = G_1(G_2 - G_{2Q}) \quad (35.97)$$

The implementation of the "high-Q" biquad is seen in Fig. 35.55 (with G_6 included). The additional gain from the figure is

$$G_{2Q} = \frac{R_{F1}}{R_{F1} \frac{Q}{Q-1}} = G_2 \cdot \frac{Q-1}{Q} \quad (35.98)$$

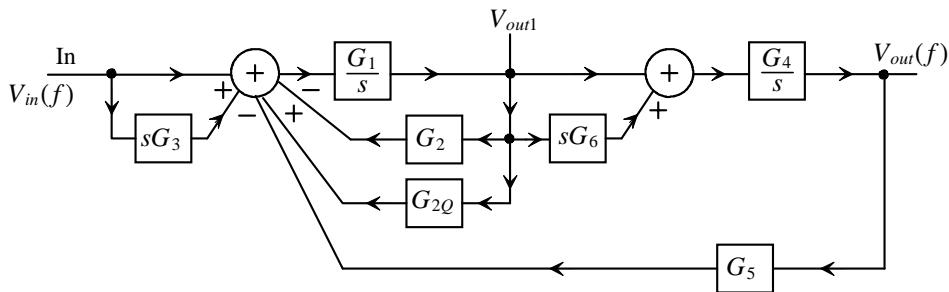


Figure 35.54 Implementation of a "high-Q" biquadratic transfer function.

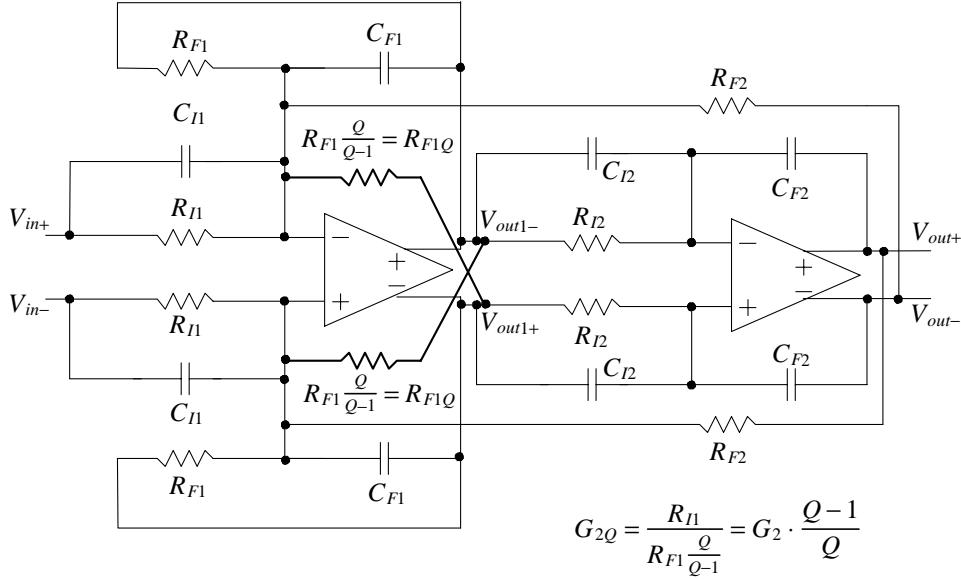


Figure 35.55 Implementation of the "high-Q" active-RC biquadratic transfer function filter.
The bold lines indicate the added components.

Rewriting Eq. (35.97) results in

$$\frac{2\pi f_0}{Q} = G_1 G_2 \left(1 - \frac{Q-1}{Q} \right) = \frac{G_1 G_2}{Q} \quad (35.99)$$

Let's use this result in the following example.

Example 35.16

Repeat Ex. 35.15 using the high-Q circuit of Fig. 35.55.

The passband gain is one so we know that

$$a_1 = \frac{2\pi f_0}{Q} = \sqrt{\frac{a_1}{G_1 G_3 G_4}} = G_1 (G_2 - G_{2Q}) = \frac{G_1 G_2}{Q}$$

or $2\pi f_0 = 10 \times 10^6 = G_1 G_2 = 1/R_{F1} C_{F1}$. We also know that

$$G_1 G_3 G_4 = \frac{C_{I1}}{C_{F1} R_{I2} C_{F2}} = G_1 (G_2 - G_{2Q}) = \frac{1}{C_{F1} R_{F1}} \cdot \frac{1}{Q}$$

and

$$2\pi f_0 = 10 \times 10^6 = \sqrt{\frac{1}{C_{F1} R_{I2} C_{F2} R_{F2}}}$$

In an attempt to minimize component spread let's set R_{F1} to 5k, R_{I2} to 20k, C_{I1} to 4p, C_{F2} to 20p, $C_{F1} = 20p$, $R_{F2} = 1.25k$, and finally $R_{FIQ} = 5.25k$ (roughly). The simulation results and schematic are shown in Fig. 35.56. ■

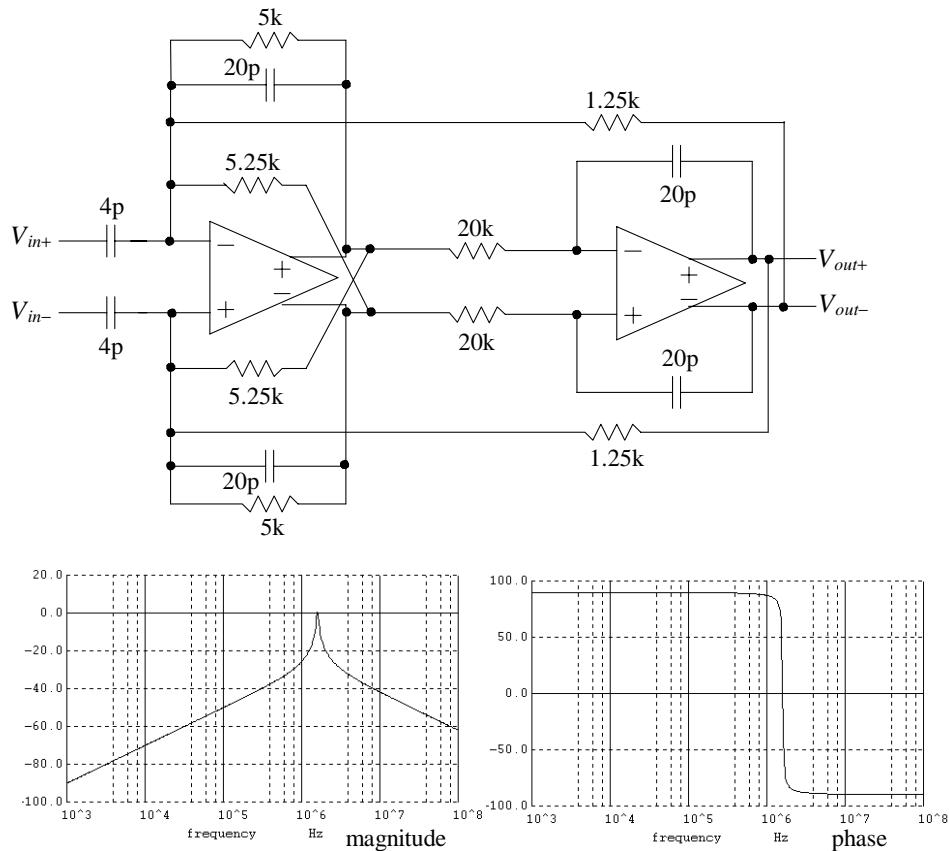


Figure 35.56 Bandpass filter discussed in Ex. 35.116.

Example 35.17

Repeat Ex. 35.16 using a switched-capacitor implementation. Assume that the filter is clocked at 100 MHz.

To implement the filter, we need to replace the resistors in Fig. 35.56 with switched capacitors. However, we notice in the gain equations that the resistors are all ratios of capacitors. This means we can reduce the size of the filter by scaling the values in Fig. 35.56. To do this let's divide all capacitors by 10 and multiply all resistors by 10. Therefore, we can write $C_{I1} = 0.4p$, $C_{F1} = 2p$, and $C_{F2} = 2p$. The resistors can be calculated using

$$R_{F1} = \frac{1}{C_{I21} \cdot f_s} = 50k \rightarrow C_{I21} = 0.2p$$

$$R_{F1Q} = \frac{1}{C_{I21Q} \cdot f_s} = 52.5k \rightarrow C_{I21Q} = 0.190p$$

$$R_{I2} = \frac{1}{C_{I12} \cdot f_s} = 200k \rightarrow C_{I12} = 0.05p (!)$$

$$R_{F2} = \frac{1}{C_{I22} \cdot f_s} = 12.5k \rightarrow C_{I22} = 0.8p$$

Looking at the value of C_{I12} , we see it may be too small. Let's change its scale factor from 10 to 4. This means

$$R_{I2} = \frac{1}{C_{I12} \cdot f_s} = 80k \rightarrow C_{I12} = 0.125p$$

We have to scale C_{F2} as well (so that G_4 remains constant). Now $C_{F2} = 5$ pF.

Figure 35.57 shows the implementation of the filter. Note how easy it was to implement the high-Q circuitry. All we did was add two capacitors to the circuit. Also note how the circuit is simplified after removing the unused components (R_{I1} and C_{I2}).

Again, as in Ex. 35.6, because this circuit can only be simulated using a transient analysis, we will input a sinewave at a known frequency and verify we get

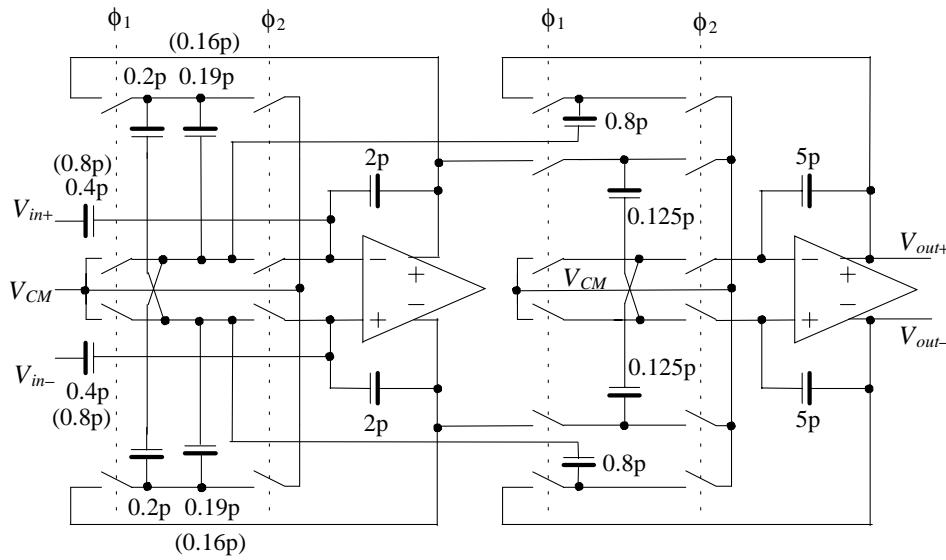


Figure 35.57 Switched-capacitor implementation of a high-Q filter; see Ex. 35.17.

the correct output. Looking at Fig. 35.56, we see that if we apply a 1 V signal to the filter at 1.59 MHz we should get a 1 V signal out at 1.59 MHz. However, as seen in Fig. 35.58, the filter is unstable and oscillates. Indeed, using simulations it's easy to show that even if we ground the inputs of the filter, the outputs will oscillate at f_0 (1.59 MHz). To understand why, remember in Fig. 35.49 that as the Q of the bandpass filter is increased, the poles move closer to the right-hand plane. If, for some reason, the poles move into the right-hand plane, the filter will become an oscillator (unstable). It's important to remember that when we designed the filter we approximated our discrete-time variable z as $1 + \frac{s}{f_s} = 1 + j\frac{2\pi f}{f_s}$ (when $f \ll f_s$ $\cos \frac{2\pi f}{f_s} \approx 1$ and $\sin \frac{2\pi f}{f_s} \approx \frac{2\pi f}{f_s}$). We could be more exact and write

$$z = e^{j\frac{2\pi f}{f_s}} = \cos \frac{2\pi f}{f_s} + j \sin \frac{2\pi f}{f_s} \quad (35.100)$$

which clearly will not follow $1 + \frac{2\pi f}{f_s}$ for frequencies f approaching the sampling frequency f_s . As we discussed in Ch. 30, sampled signals will have spectral content in excess of the sampling frequency. Practically, the spectral content is limited by the combination of the switches "on" resistance and the capacitors in the circuit. In Eqs. (35.68) - (35.71) we wrote, for a pole,

$$z - (1 - G_{1D}) \approx 1 + j \cdot \frac{2\pi f}{f_s} - 1 + G_{1D} = G_{1D} + j \cdot \frac{f}{f_s/2\pi} \quad (35.101)$$

The pole is ideally located at $(G_{1D}f_s)/2\pi$ (the frequency where the imaginary part is equal to the real part). Using Eq. 35.100, we can write, more exactly,

$$z - (1 - G_{1D}) = \underbrace{\cos \frac{2\pi f}{f_s} - 1 + G_{1D}}_{\text{Real}} + j \underbrace{\sin \frac{2\pi f}{f_s}}_{\text{Imaginary}} \quad (35.102)$$

It should be clear from this equation that as f gets larger, the cosine term will decrease from one, causing the real portion to get smaller. A decrease in the real component, as seen in the complex plane in Fig. 35.49, causes the pole to move closer to the right-hand plane (causing the Q to increase).

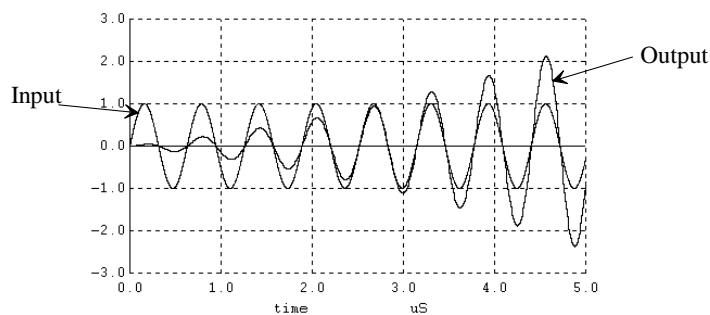


Figure 35.58 Output of the filter in Fig. 35.57 showing instability.

Practically, the maximum Q we can design for (not attain) is in the neighborhood of 5. If we redesign the biquad of Fig. 35.57 to have a Q of 5, we see that all we need to change is C_{I1} (from 0.4p to 0.8p) and C_{F2O} (from 0.19p to 0.16p). The simulation results are seen in Fig. 35.59. In this figure, we adjusted the input frequency until we reached a 3 dB point (the filter's center frequency was 1.59 MHz, as expected). This occurred at 1.52 MHz and 1.66 MHz. The Q of the circuit is not 5 as we designed for but is, from Fig. 35.49, $1.59/(1.66 - 1.52)$ or 11.36. (See problem 35.25 for reducing the spread between C_{F2} and C_{I2}). ■

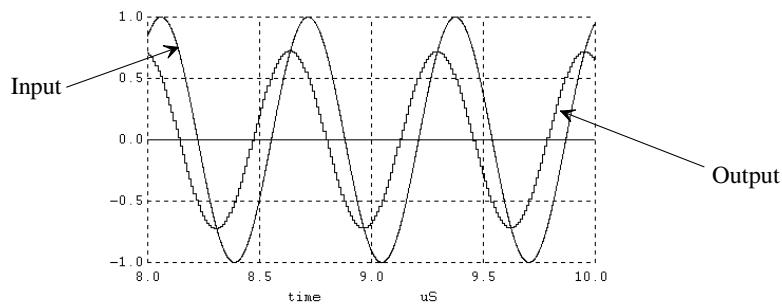


Figure 35.59 Output of the filter in Fig. 35.57 after lowering the Q to maintain stability.

Q Peaking and Instability

While it would appear the active-RC circuit is the best choice for high-Q filter implementations, we must remember that the discussion neglected the effects of the finite gain-bandwidth product (f_u) of the op-amps. We can model these effects by replacing the ideal integrator gain of $1/s$ with

$$\frac{1}{s} \rightarrow \frac{1}{s\left(1 + \frac{s}{2\pi f_u}\right)} \quad (35.103)$$

Using this result, we can rewrite the pole locations of Eq. (35.80) (see Eqs. [35.81] and [35.82]) as

$$\frac{1}{\left(s\left(1 + \frac{s}{2\pi f_u}\right) + p_1\right)} \cdot \frac{1}{\left(s\left(1 + \frac{s}{2\pi f_u}\right) + p_2\right)} \quad (35.104)$$

or, looking at a single term,

$$s\left(1 + \frac{s}{2\pi f_u}\right) + p_1 = s + p_1 - \underbrace{\frac{(2\pi f)^2}{2\pi f_u}}_{\text{Unwanted}} \quad (35.105)$$

This subtraction results in a shift in the pole toward the right-hand plane, increasing the Q of the circuit; Fig. 35.60. Reviewing Eq. (35.82), we can subtract the unwanted term in Eq. (35.105) to estimate the shift in the Q or

$$\frac{\pi f_0}{Q} - \frac{(2\pi f)^2}{2\pi f_u} \text{ or at } f=f_0 \text{ we can write } \pi f_0 \left(\frac{1}{Q} - \frac{2f_0}{f_u} \right) \quad (35.106)$$

The shifted Q is then

$$\frac{1}{Q_{shift}} = \frac{1}{Q} - \frac{2f_0}{f_u} \rightarrow Q_{shift} = \frac{Q}{1 - \frac{Q2f_0}{f_u}} \quad (35.107)$$

So, for the filter Q to remain finite, we require

$$\frac{Q2f_0}{f_u} \ll 1 \quad (35.108)$$

Let's use this result in the following example.

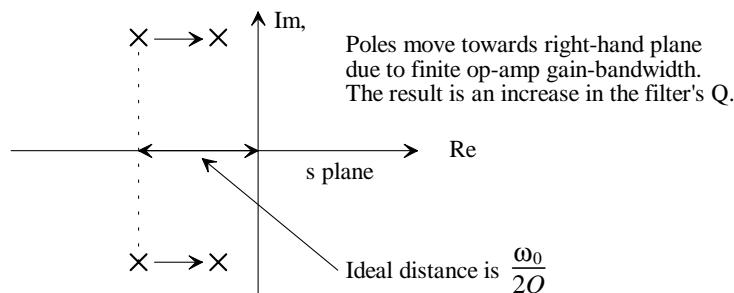


Figure 35.60 Showing Q peaking resulting from the op-amp finite gain bandwidth product.

Example 35.18

Resimulate the filter in Ex. 35.16 using op-amps that have an f_u of 100 MHz.

The center frequency, f_0 , of this filter is 1.59 MHz and the Q is 20. Using Eqs. (35.107) and (35.108), we can estimate the increase in Q due to op-amp finite gain-bandwidth product as

$$\frac{Q2f_0}{f_u} = \frac{20 \cdot 2 \cdot 1.59}{100} = 0.636 \rightarrow Q_{shift} = 55$$

Practically, this is too high of a Q (the poles are too close to the right-hand plane), and the filter will be unstable (noise in the circuit, or simulation noise in the simulation, will push the poles into the right-hand plane). Figure 35.61 shows the simulation results (see also Ex. 35.4). The inputs to the filter are grounded. The unstable oscillation frequency is close to the ideal, f_0 , but is shifted by a small amount. ■

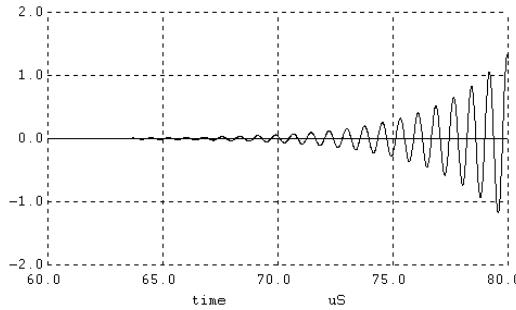


Figure 35.61 Showing how the filter of Ex. 35.16 becomes unstable due to finite op-amp bandwidth.

Transconductor-C Implementation

Let's redraw the bilinear filter in Fig. 35.31 as seen in Fig. 35.62. We redraw it like this to show how the feedback gain, G_2 , is implemented. In the block diagram of the biquad filter shown in Fig. 35.43, we used a similar scheme to implement the feedback gain, G_5 . Figure 35.63 shows the implementation of a biquad filter using transconductors where we have drawn it so that the transconductors appear to be connected in series. This topology can be redrawn so that it looks similar to Fig. 35.62 (showing a direct correspondence between it and Fig. 35.43). Note how we could have drawn the schematic without the crossing wires if we switched the output polarity of two of the transconductors (that is, put the minus output on the top of the output instead of the plus output).

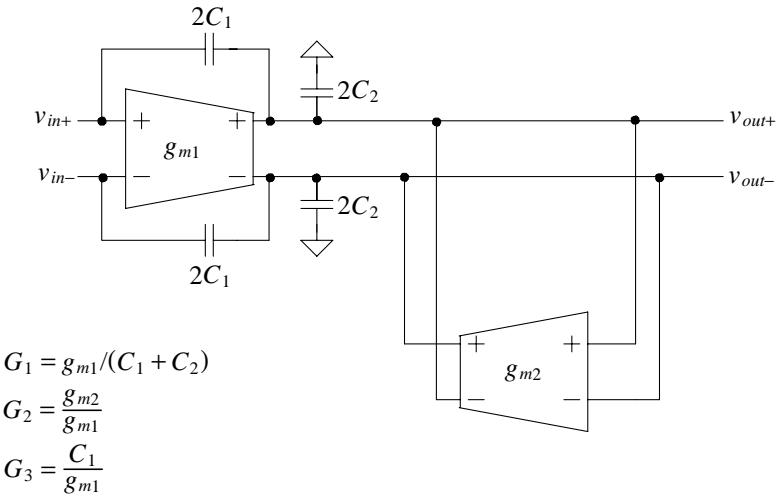
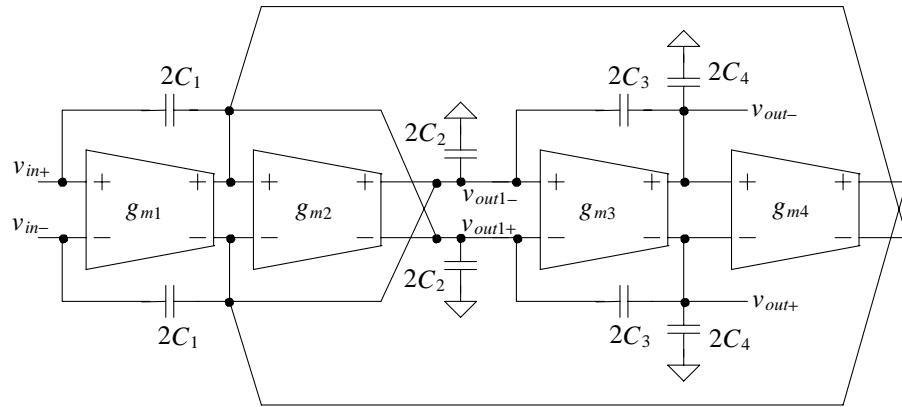


Figure 35.62 Redrawing the bilinear filter shown in Fig. 35.31.



$$G_1 = g_{m1}/(C_1 + C_2) \quad G_2 = \frac{g_{m2}}{g_{m1}} \quad G_3 = \frac{C_1}{g_{m1}} \quad G_4 = g_{m3}/(C_3 + C_4) \quad G_5 = \frac{g_{m4}}{g_{m1}} \quad G_6 = \frac{C_3}{g_{m3}}$$

Figure 35.63 Implementing a biquadratic filter using transconductors.

The Digital Biquad

The digital biquad filter based on the canonic form of Fig. 35.42 is shown in Fig. 35.64. The transfer function of this filter is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B_0 + B_1 z^{-1} + B_2 z^{-2}}{1 - A_1 z^{-1} - A_2 z^{-2}} = \frac{B_0 z^2 + B_1 z + B_2}{z^2 - A_1 z - A_2} \quad (35.109)$$

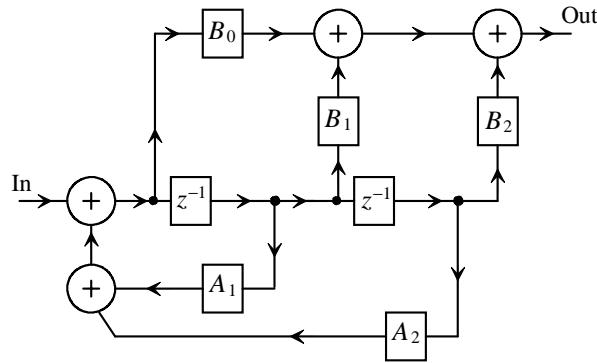


Figure 35.64 The digital biquad filter (see Fig. 35.42).

To translate this transfer function into the frequency domain, we use Eq. (35.69) and assume our frequencies of interest are much less than the sampling frequency

$$H(f) = \frac{B_0 \left(1 + \frac{s}{f_s}\right)^2 + B_1 \left(1 + \frac{s}{f_s}\right) + B_2}{\left(1 + \frac{s}{f_s}\right)^2 - A_1 \left(1 + \frac{s}{f_s}\right) - A_2} \quad (35.110)$$

After some algebraic manipulation we can put this equation in the form seen in Eq. (35.80)

$$H(f) = \frac{B_0 \cdot s^2 + f_s(2B_0 + B_1) \cdot s + f_s^2(B_0 + B_1 + B_2)}{s^2 + f_s(2 - A_1) \cdot s + f_s^2(1 - A_1 - A_2)} \quad (35.111)$$

where

$$a_2 = B_0 \quad (35.112)$$

$$a_1 = f_s(2B_0 + B_1) \quad (35.113)$$

$$a_0 = f_s^2(B_0 + B_1 + B_2) \quad (35.114)$$

$$\frac{2\pi f_0}{Q} = f_s(2 - A_1) \quad (35.115)$$

$$f_0 = \frac{f_s}{2\pi} \cdot \sqrt{(1 - A_1 - A_2)} \quad (35.116)$$

Example 35.19

Repeat Ex. 35.11 using the digital biquad clocked at 100 MHz.

In this example a lowpass filter is designed with $f_0 = 1.59$ MHz and $Q = 0.707$. Reviewing Fig. 35.45, we see that for a lowpass filter a_2 and a_1 are zero. This means, in Eq. (35.111), B_0 and B_1 are zero. Further,

$$Q = \frac{2\pi \cdot 1.59 \times 10^6}{100 \times 10^6 (2 - A_1)} = 0.707 \rightarrow A_1 = 1.859$$

and

$$1 - A_1 - A_2 = \left(\frac{2\pi f_0}{f_s}\right)^2 \rightarrow A_2 = -0.869$$

and finally, because the gain at DC is 1,

$$B_2 = 1 - A_1 - A_2 = 0.01$$

Note that if a scaling in the amplitude is allowable, we can remove this multiplication or approximate it with shifts in the digital word.

The simulation results are shown in Fig. 35.65. To implement this simulation in WinSPICE, we used transmission lines for the delay elements and voltage-controlled voltage sources for both the multiplications and the adders. Note how the frequency response is periodic with the filter's clocking frequency. ■

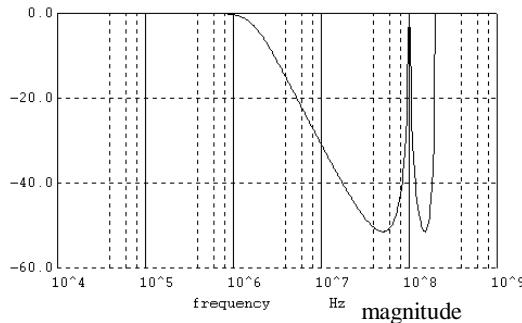


Figure 35.65 Simulating the digital filter of Ex. 35.19

35.3 Filters Using Noise Shaping

The basic idea of implementing a filter using noise-shaping is seen in Fig. 35.66. The modulator converts the analog input signal into a digital signal containing both the original input spectrum and the unwanted modulation noise. The output of the circuit is digital. If we want an analog output signal, then we would add a noise-shaping demodulator to the output of the circuit.

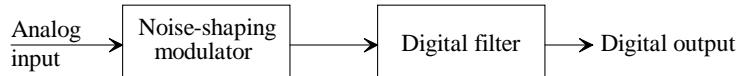


Figure 35.66 Combining a noise-shaping modulator with digital filter.

Removing Modulation Noise

Back in Ch. 32 we said that we can use a sinc averaging filter with an order L (see Eq. [32.20]) that is one greater than the order of the modulator, M . For a second-order modulator implemented using an oversampling ratio K , the filter would have a transfer function given by

$$H(z) = \left[\frac{1}{K} \cdot \frac{1-z^{-K}}{1-z^{-1}} \right]^3 \quad (35.117)$$

Assuming we don't design the filter to decimate (reduce the digital word output rate) the modulator's output, this filter will require more than $3K$ registers. As K gets large, this can result in a large layout area. To reduce this layout area, as discussed in Sec. 31.2.2, we can decimate the modulator's output. However, the big concern when using decimation to reduce the filter's size and complexity is aliasing. It would be nice to have a small-area filter to remove modulation noise.

Consider the frequency response of the Sinc decimating filter shown in Fig. 35.67. This filter uses a clocking frequency of 100 MHz and assumes K is 16. Note how, at the bandwidth of the desired signal, B , there is significant droop in the filter's response. It will be highly desirable to design a filter that doesn't have this droop or, even more desirable, contains a small amount of peaking at B to compensate for the eventual needed decimation filter response (a Sinc shape with 3.9 dB droop at B). Using Eqs. (35.114) - (35.116) (and some simulations), we can set, for a digital biquad, $B_2 = 0.03125$, $A_1 = 1.75$, $A_2 = -0.78125$ (there are several other solutions, depending on the desired complexity or performance of the filter). Knowing that the sinc filter was almost ideal for removing modulation noise and that a second-order (biquad) filter rolls-off at -40 dB/decade, we can estimate, with the help of Fig. 35.67, the number of biquads required to remove the modulation noise. One decade above 3.125 MHz (i.e., at 31.25 MHz) the attenuation of the sinc filter is approximately -70 dB. This might make us think that two biquads in cascade would be enough to remove the modulation noise since one decade above f_0 we would have -80 dB attenuation (and two may suffice in many applications). However, as seen in Fig. 35.67, the sinc filter response is not monotonic. We will use three biquads to remove the modulation noise (or again the number of filters cascaded is one more than the order of the modulator). The simulation results comparing the Sinc and biquad filters are seen in Fig. 35.68. The transfer function of the biquad filter is

$$H(z) = \frac{0.03125}{z^2 - 1.75z + 0.78125} \quad (35.118)$$

The block diagram of the filter is seen in Fig. 35.69.

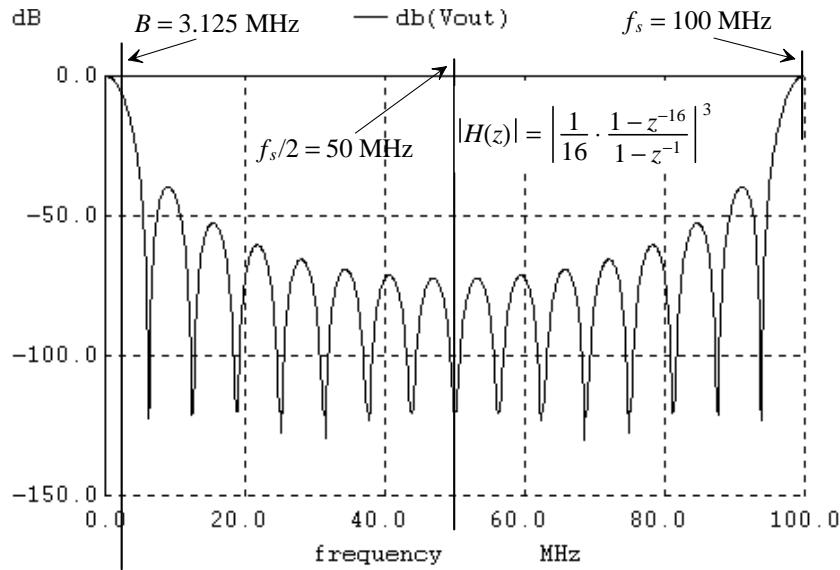


Figure 35.67 Frequency response of a third-order comb filter.

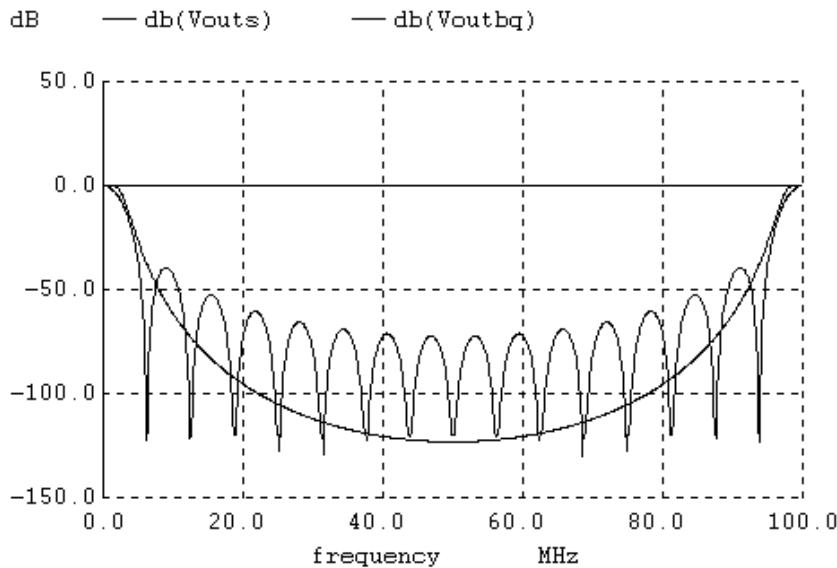


Figure 35.68 Comparing a third-order Sinc filter to a third-order biquad for removing modulation noise in a second-order noise-shaping modulator.

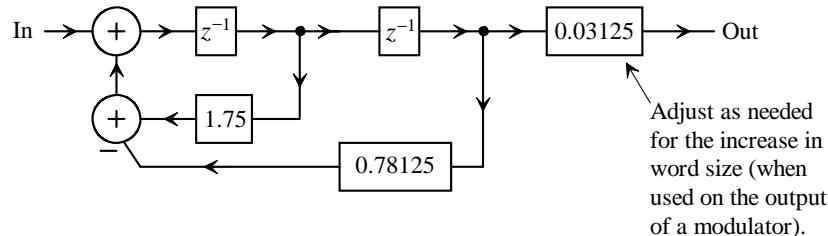


Figure 35.69 The digital biquad filter described by Eq. (35.118).

Example 35.20

Redesign the filter in Fig. 35.69 so that the response has a small amount of peaking at 3.125 MHz. Compare, with simulations, the new response to the sinc filter seen in Fig. 35.67.

Reviewing Eq. (35.115), we see that to increase the Q we need to increase A_1 . Keeping in mind that we want to have simple multiplications relying heavily on shifts, let's try increasing A_1 to 1.78125 and A_2 to 0.8125. The simulation results are seen in Fig. 35.70. In this figure we compare the modified filter response to the response of the Sinc filter. Note how we have a couple of dB peaking in the cascaded biquad filter's output. ■

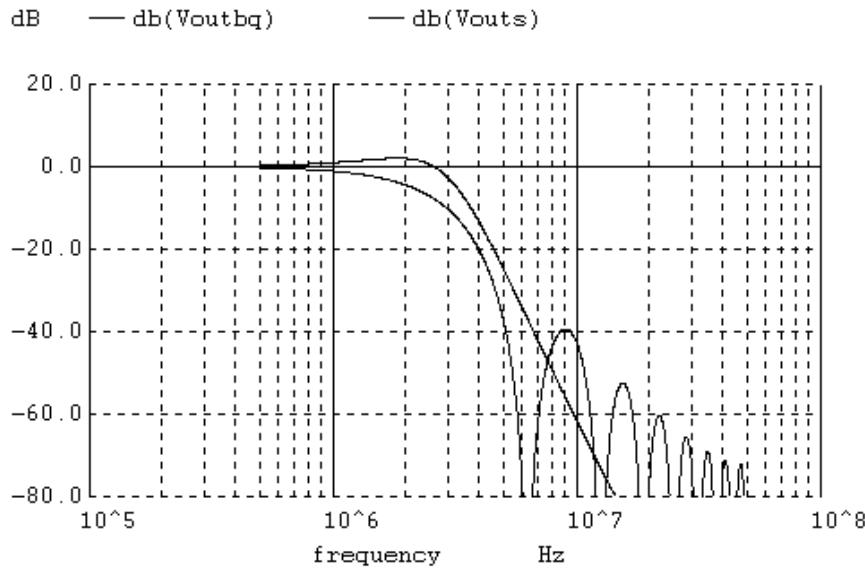


Figure 35.70 Simulation results for Ex. 35.20.

Implementing the Multipliers

An important component of any digital filter is the multiplier. While implementing general digital multipliers is outside the scope of this book, we can discuss practical multiplier implementation for custom digital filters. To begin let's remember from Ch. 31 (see Figs. 31.36 and 31.37) that in the binary offset representation of a number the most positive voltage in a digital system ($V_{REF_+} - 1$ LSB) is represented by 11111... The most negative voltage (V_{REF_-}) is represented by 00000... in binary offset while the center voltage, or common-mode voltage V_{CM} , is represented by 10000... To change between binary offset and two's complement, we simply invert the MSB (the sign-bit). We'll assume that two's complement numbers are used throughout the filter.

Let's say that a noise-shaping modulator uses a single-bit quantizer that has an output of 1 or 0. To filter this data stream, we begin by changing the output into a two's complement number

$$1 \rightarrow 01 \text{ (+1 in two's complement)} \quad (35.119)$$

and

$$0 \rightarrow 11 \text{ (-1 in two's complement)} \quad (35.120)$$

To multiply the output by two, we simply add a zero to the end of the word

$$+1 \times 2 \rightarrow 010 \text{ (+2 in two's complement)} \quad (35.121)$$

or

$$-1 \times 2 = 110 \text{ (-2 in two's complement)} \quad (35.122)$$

To multiply by one-half, we simply insert a zero in between the sign bit and the remaining part of the word if the word is positive

$$+1 \times 0.5 = 001 \text{ (+1 in two's complement)} \quad (35.123)$$

or a one if the word is negative

$$-1 \times 0.5 = 111 \text{ (-1 in two's complement)} \quad (35.124)$$

In either case, what we are doing is simply extending the sign bit one place to the left. Equations (35.123) and (35.124) may not make sense unless we rewrite Eqs. (35.119) and (35.120) if the single-bit modulator outputs were instead changed to 3-bit numbers

$$\text{modulator output of } 1 \rightarrow +1 \rightarrow 010 \text{ (+2 in two's complement)} \quad (35.125)$$

and

$$\text{modulator output of } 0 \rightarrow -1 \rightarrow 110 \text{ (-2 in two's complement)} \quad (35.126)$$

It's important to note that the word size increases after the multiplication to avoid rounding errors (unless, of course as seen in Eqs. [35.125] and [35.126], the LSB is always 0). Figure 35.71 shows the implementation of multiply by 2 and 0.5.

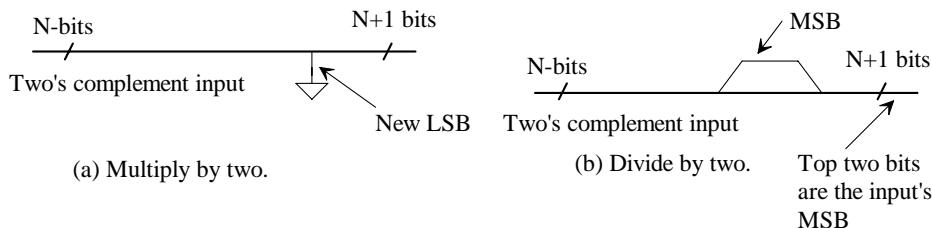


Figure 35.71 Multiply by (a) two and (b) one-half.

Example 35.21

Suppose a noise-shaping modulator uses a 4-bit quantizer where 1111 is the most positive output code and 0000 is the most negative output code. How would these codes be converted into two's complement format?

Inherent in the binary offset format shown back in Fig. 31.36 is an LSB offset. This resulted in our LSB being defined by, see Eq. (30.23),

$$1 \text{ LSB} = \frac{V_{REF+} - V_{REF-}}{2^N} \quad (35.127)$$

In a noise-shaping modulator (see question 30.14) we may use a quantizer that doesn't have this offset. When using a 1-bit quantizer, an output of 1 corresponds to V_{REF+} and an output of 0 corresponds to V_{REF-} . We can write 1 LSB for this situation as

$$1 \text{ LSB} = \frac{V_{REF+} - V_{REF-}}{2^{N-1}} \quad (35.128)$$

To convert the 4-bit modulator outputs ranging from 0 to 15 into two's complement, we simply complement the MSB. This would mean the most positive output in two's complement is 0111, while the most negative output is 1000. In order to eliminate the offset (asymmetry in the digital output code) we may use an additional output bit in our quantizer so that the maximum modulator output is 1 0000 corresponding to V_{REF+} . If this is the case, then the modulator outputs can be converted to two's complement as seen below.

Maximum modulator output 1 0000 → 0 1000 (+8 in two's complement)

Middle modulator output 0 1000 → 0 0000 (0 in two's complement)

Minimum modulator output 0 0000 → 1 1000 (-8 in two's complement)

■

We can extend our discussion of multiplying by 0.5 to multiplying by 0.25, 0.125, 0.0625, 0.03125, etc., by simply extending the sign-bit of the input word. To multiply a word by 0.03125, we simply shift the word to the right five times. For example, multiplying a two's complement code of 01 by 0.03125 would give

$$+1 \times 0.03125 = \overbrace{01}^{\text{+1 or +0.5}} \times 0.03125 = \overbrace{000\ 0001}^{\text{+1 or 0.015625}} \quad (35.129)$$

Multiplying a two's complement code of 11 by 0.0625 results in

$$-1 \times 0.0625 = \overbrace{11}^{\text{-1 or -0.5}} \times 0.0625 = \overbrace{11\ 1111}^{\text{-1 or -0.03125}} \quad (35.130)$$

While, in some filtering applications, simple shifts can prove very useful, we can implement more useful multipliers using adders. Figure 35.72 shows one possible implementation using a single adder along with the associated multiplication factors. We could implement the coefficients in Ex. 35.20 using a similar scheme. For example, $A_1 = 1.78125 = 2 - 0.25 + 0.03125$ and $A_2 = 0.8125 = 1 - 0.25 + 0.0625$ (both requiring two adders). Other creative ways can be used to implement multipliers. For example, a multiplication of 0.5625 by cascading two simple multipliers with multiplication factors 0.875.

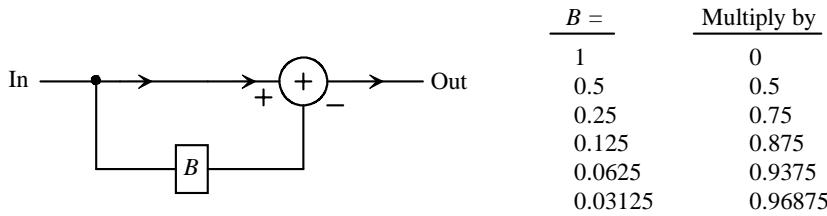


Figure 35.72 A simple multiplier using a single adder.

REFERENCES

- [1] R. E. Bogner and A. G. Constantinides (eds.), *Introduction to Digital Filtering*, John Wiley and Sons, 1975. ISBN 0-471-08590-1
- [2] P. R. Gray, D. A. Hodges, and R. W. Brodersen (eds.), *Analog MOS Integrated Circuits*, Wiley-IEEE, 1980. ISBN 0-471-08964-8
- [3] P. R. Gray, B. A. Wooley, and R. W. Brodersen (eds.), *Analog MOS Integrated Circuits II*, Wiley-IEEE, 1989. ISBN 0-87942-246-7
- [4] P. A. Lynn, *An Introduction to the Analysis and Processing of Signals*, Hemisphere Publishing Corporation, 1989. ISBN 0-89116-981-4
- [5] Y. P. Tsividis and J. O. Voorman (eds.), *Integrated Continuous-Time Filters: Principles, Design, and Applications*, Wiley-IEEE, 1993. ISBN 0-7803-0425-X
- [6] B. Nauta, *Analog CMOS Filters for Very High Frequencies*, Kluwer Academic Publishers, 1993. ISBN 0-7923-9272-8
- [7] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, Oxford University Press, 1998. ISBN 0-1951-1663-1
- [8] D. A. Johns and K. Martin, *Analog Integrated Circuit Design*, John Wiley and Sons, 1997. ISBN 0-471-14448-7
- [9] E. P. Cunningham, *Digital Filtering: An Introduction*, John Wiley and Sons, 1995. ISBN 0-471-12475-3
- [10] R. Schaumann and M. E. Van Valkenburg, *Design of Analog Filters*, Oxford University Press, 2001. ISBN 0-19-511877-4

QUESTIONS

- 35.1** Resketch Fig. 35.2 for the following circuit.

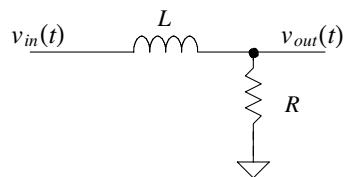


Figure 35.73 First-order lowpass filter using an inductor and a resistor.

- 35.2** Show that Eq. (35.6) is still valid if the circuit's inputs and outputs are referenced to the common-mode voltage, V_{CM} . (The op-amp inputs should also be at V_{CM})

- 35.3** Sketch the implementation of a first-order lowpass filter using a CAI with a 3 dB frequency of 10 MHz and a DC gain of 6 dB. Simulate your design to verify if it works as expected.
- 35.4** Plot, in the complex plane, the ideal pole location and the actual pole locations due to finite op-amp unity gain frequency for the filter described in Ex. 35.4.
- 35.5** Regenerate Fig. 34.21 of the last chapter using SPICE and the op-amp model shown in Fig. 35.8.
- 35.6** Suppose an antialiasing filter was required for a 12-bit data converter. Further assume the filter is to be implemented using an active-RC topology. If $VDD = 1.5$ V, estimate the minimum value of the integration capacitor used, assuming the filter's noise performance is dominated by thermal noise. Is it wise, for 12-bit system performance, to design the filter so that its SNR is equal to the SNR of the data converter?
- 35.7** Repeat question 35.6 if the op-amp used in the filter has a linear output swing of 80% of the power supply voltage.
- 35.8** Show, using the topology shown in Fig. 33.22 (and the same SPICE models), how using the two MOSFETs linearizes the change in resistance with V_{DS} .
- 35.9** Repeat Ex. 35.4 using a MOSFET-C filter. Use the MOSFET SPICE model given in Ch. 33. After performing the AC simulations, try a transient simulation with an input sinusoid at 1 MHz. Show how the output of the filter becomes distorted as the amplitude of the input signal increases. Determine the filter's SNDR when the input signal has a frequency of 1 MHz and an amplitude of VDD peak-to-peak.
- 35.10** Derive the transfer function for the filter shown in Fig. 35.16 if the transconductors have different g_m s. Sketch the block diagram, similar to the one seen in Fig. 35.6, for the filter.
- 35.11** Derive the transfer function for the following first-order transconductor filter.

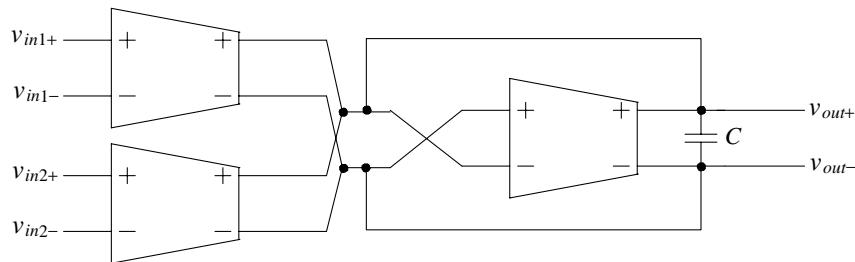


Figure 35.74 A first-order filter with two inputs.

- 35.12** Show the derivation details that result in Eqs. (35.44) and (35.46).

- 35.13** From the derivation of Eq. (35.48), what would happen if we repeated Ex. 35.6 with an input frequency of 101.59 MHz?
- 35.14** Show that if the values of A and B are restricted to 1, 0.5, 0.25, 0.125, etc. that the circuit of Fig. 35.75 can be used to implement multiplication by coefficients that aren't directly powers of two. How would a multiply by 0.75 be implemented? a multiply-by-0.9375? a multiply-by-0.5625?

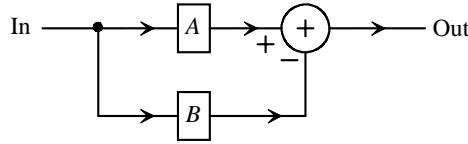


Figure 35.75 A simple multiplier where A and B simply shift the data.

- 35.15** From the results of the preceding question, sketch the implementation of one possible design of a multiplier topology that multiplies an input word by 0.8789 and 0.3164.

- 35.16** Show the details of how the gains, G , are derived in Fig. 35.30.

- 35.17** In Fig. 35.35 a filter section has a transfer function that can be written

$$H(z) = (1 + A) \cdot \frac{z - A/(1 + A)}{z}$$

For this transfer function generate a z-plane plot and a magnitude plot similar to what is seen in Fig. 35.27.

- 35.18** Plot the time-domain output of the filter in Fig. 35.37 when the input is a zero to one step function.

- 35.19** Design a first-order canonic digital filter that is clocked at 100 MHz and has a transfer function, in the frequency domain, given by

$$H(f) = \frac{1}{1 + j \frac{f}{4 \text{ MHz}}}$$

- 35.20** Redraw Fig. 35.42 using only two input adders.

- 35.21** Is it possible to tune the gain, Q , and cutoff frequency of the lowpass biquad independently? If so, how? Give examples using the simulation netlist used to generate Fig. 35.48.

- 35.22** What happens to the poles in the biquadratic equation, Eq. (35.80), if the Q is less than 0.5? (Hint: The filter behaves like the cascade of two first-order filters.) Is the f_{\max} equation in Fig. 34.45 valid?

- 35.23** Compare the size of the elements used in Exs. 35.11 and 35.12. Is there a benefit to using an active element for monolithic implementation?
- 35.24** Show, using the simulations from Ex. 35.17, that increasing the switch resistance, and thus the spectral content present in a switched capacitor circuit, can help to stabilize high-Q switched-capacitor bandpass filters.
- 35.25** Redesign and simulate the operation of the filter discussed in Ex. 35.17, with a Q of 5, while trying to minimize the difference between C_n and C_{F2} . Suggest a possible modification to the filter topology (similar to how we add G_{2Q} in Fig. 35.54) to reduce this component spread.
- 35.26** Derive the transfer function of the transconductor-C biquad shown in Fig. 35.63. Can this filter be orthogonally tuned? If so how?
- 35.27** Repeat Ex. 35.12 using the transconductor-based biquad.
- 35.28** How would a "high-Q" biquad be implemented using transconductors? Repeat Ex. 35.15 using the transconductor-based biquad.
- 35.29** Repeat Ex. 35.13 using a digital filter.
- 35.30** Repeat Ex. 35.14 using a digital filter.
- 35.31** Show, using biquad sections, how the following lowpass ladder filter would be implemented.

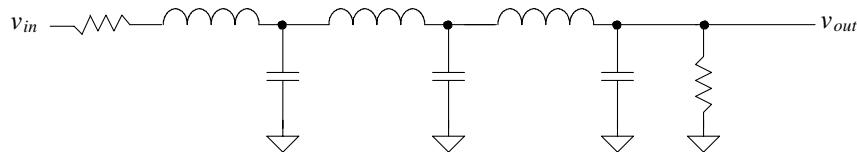


Figure 35.76 Implementing a ladder filter using biquads, see problem 35.30.

- 35.31** Show how to implement the multipliers used in Ex. 35.20.
- 35.32** Show that the filter shown in Fig. 35.77 can be implemented using a single multiplier.
- 35.33** Show how the output of a single-bit noise-shaping modulator would be multiplied by 1.9375. Make sure that the detail on converting the modulator's output to two's complement is shown.

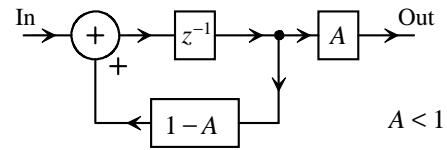


Figure 35.77 Filter used for Problem 35.32.

Chapter

36

At the Bench

In this chapter we present some practical prototyping techniques to illustrate a few of the concepts discussed in this book. The goal of the chapter is to simply provoke thought and show alternative possibilities (other than hand calculations and simulations) for looking at the performance of a mixed-signal circuit or system.

36.1 A Push-Pull Amplifier

The basic CMOS push-pull amplifier (see Ch. 22) is shown in Fig. 36.1a. Figure 36.1b shows the schematic of the implementation used in this section where we have AC coupled the input and output of the amplifier to allow the 9.1 MEG resistor to self-bias the circuit. The bold numbers shown adjacent to the MOSFET terminals correspond to the pin numbers of the 4007, Fig. 36.2, used to prototype the amplifier. Note that by AC coupling our input and output, we can use ground as our input/output reference.

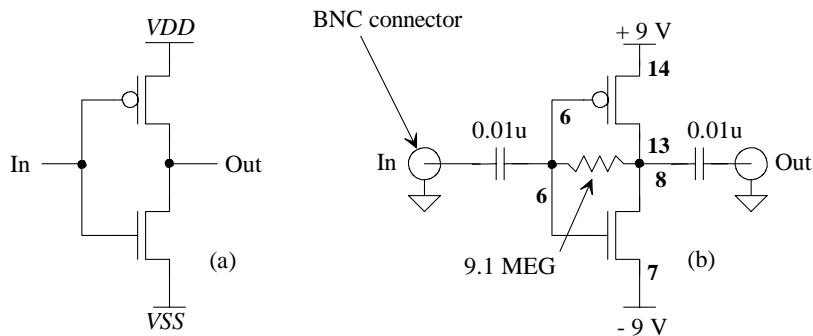


Figure 36.1 (a) Push-pull amplifier and (b) prototyping the amplifier.

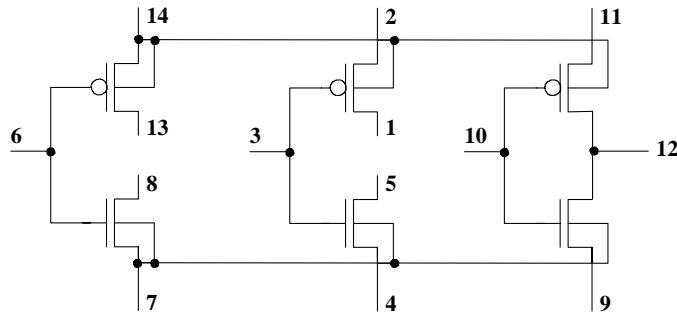


Figure 36.2 Pin diagram for the 4007. Note how the bodies of all PMOS devices are tied to pin 14, while the bodies of the NMOS devices are tied to pin 7. This means that pin 14 must be tied to the highest potential in the circuit (if the PMOS devices are used), and pin 7 must be tied to the lowest potential.

While the detailed datasheet for the 4007 can be found at the book's website (<http://cmosedu.com>), we will comment that the threshold voltage for these transistors is approximately 2 V and that they can drive around 1 mA or so into a load (this is a serious limitation and will limit the size of the load we can drive). The large threshold voltage and limited drive capability are the reasons we used ± 9 V power supplies.

Deadbug Prototyping

Figure 36.3 shows a chip flipped upside down and placed on a copper conductor (a glass epoxy, FR-4 material coated with copper used in printed circuit board manufacturing). The reason this technique is called "deadbug prototyping" should be obvious (unless the reader is so lucky they've never seen a dead cockroach). We use this approach instead of the common white protoboards found in most undergraduate electronics laboratories for prototyping because of the ability to use a good ground plane (the copper conductor). The big drawback is the need to solder all of the components together. A good (meaning equipotential) ground plane is essential to any low-noise, wide-dynamic range, measurement. We'll also use BNC connectors to pipe our signals on-to and off-of the board to avoid long wires, which tend to pick up coupled noise.

Figure 36.4 shows the prototype of the amplifier in Fig. 36.1b implemented using the deadbug technique. The black and red wires coming into the circuit provide power and ground. At the connections of power and ground on the board we add a decoupling capacitor (a capacitor soldered from the power-supply connection to the ground plane). This capacitor provides charge for any fast transients that may occur in the circuit. The capacitor leads can be twisted into small loops and soldered to the ground plane or to a pin on a chip and used as a contact point for the power supply clips. Before looking at some measurement results, we need to discuss probe loading and measurement techniques.

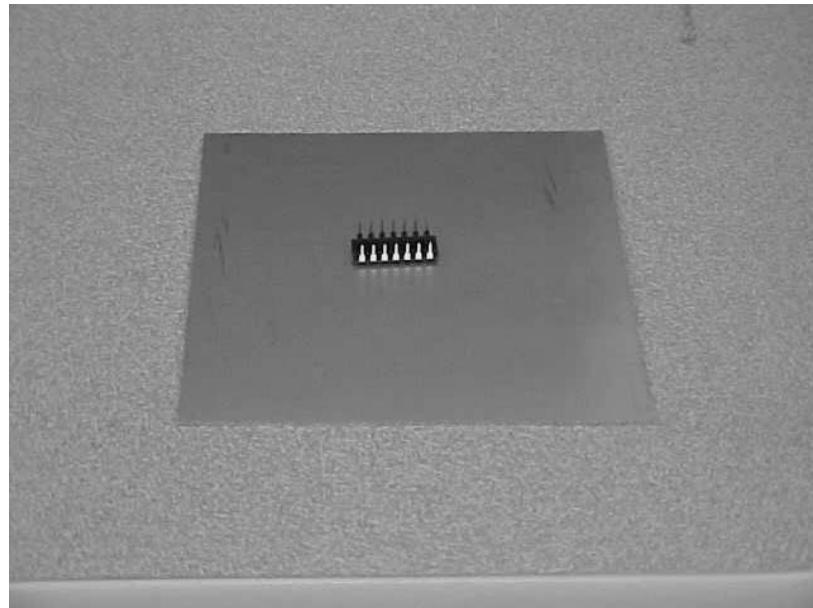


Figure 36.3 Deadbug prototyping using a copper ground plane.

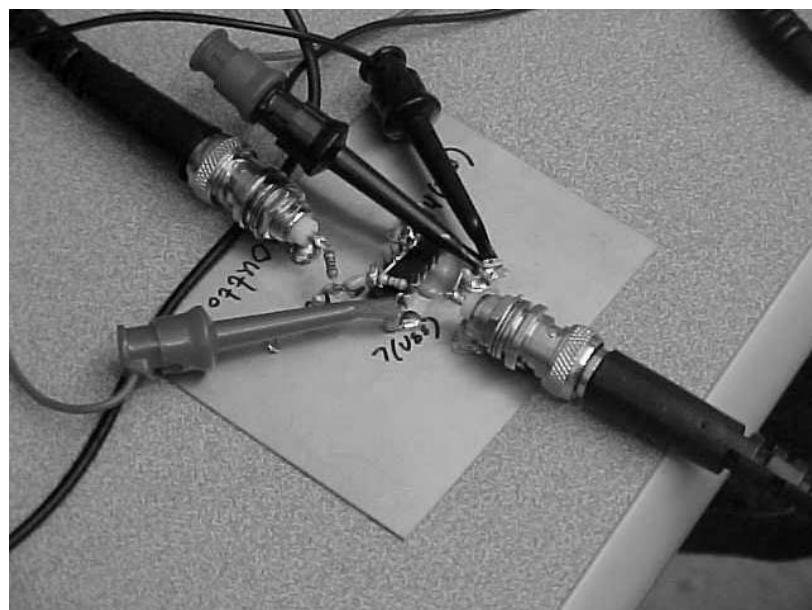


Figure 36.4 Deadbug prototype of the amplifier in Fig. 36.1b.

Probing

If we're not careful, we can load the circuit we are testing with our measuring system. Consider the connection of a piece of coaxial cable to the oscilloscope shown in Fig. 36.5. The scope has an input resistance of $1 \text{ M}\Omega$ and an input capacitance of 15 pF . When a coax cable is driven by a large impedance and is terminated with a large impedance, we think of it as a capacitor. If a 5-foot piece of coax is used to connect the push-pull amplifier to the oscilloscope, we would get the circuit shown in Fig. 36.6. Clearly, the measuring system will load the amplifier and keep us from accurately measuring the response of the circuit.

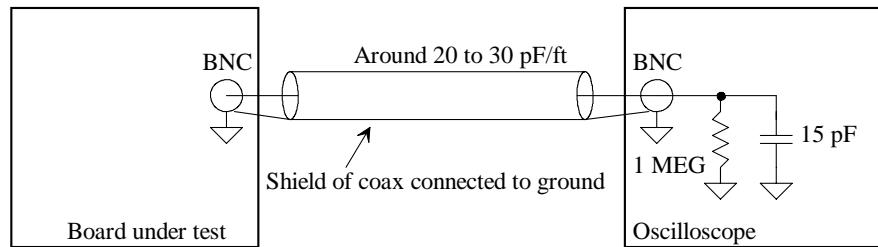


Figure 36.5 The loading when probing with a piece of coaxial cable.

To reduce the loading by the required coaxial interconnect cable (150 pF in Fig. 36.6), a scope probe trades off sensitivity for lighter loading. Figure 36.7 shows a *10:1 compensated* scope probe. The term $10:1$ represents the attenuation factor from the probe tip to the input of the scope. One-tenth of the voltage on the tip of the probe actually makes it to the input of the scope. The term "compensated" indicates that the probe is designed to compensate for the large loading of the coaxial cable. If the probe is compensated correctly, the impedance in the probe's tip ($9Z$) is exactly, independent of

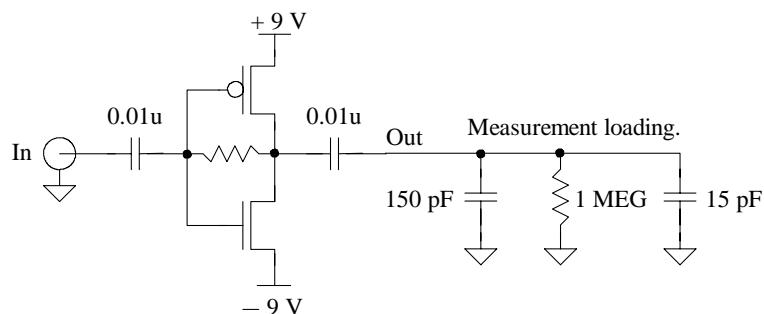


Figure 36.6 How we can mess up a measurement if we're not careful.

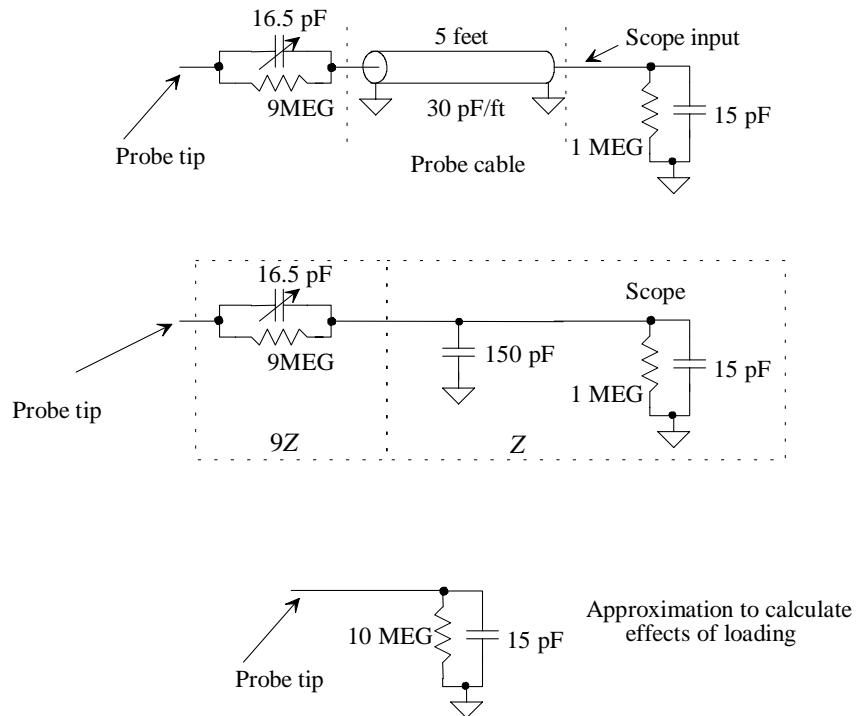


Figure 36.7 How a compensated probe loads a circuit.

frequency, nine times larger than the impedance of the combination of the coaxial cable and the scope (Z). Note how the loading of the probe at DC is 10 MEG while at high frequencies the loading is roughly 15 pF.

Testing the Circuit

To test this circuit, we'll use a vector signal analyzer, VSA, (an instrument similar to a spectrum analyzer with the capability to perform an inverse Fourier transform for viewing a signal in the time domain). A test setup is seen in Fig. 36.8. We'll use an input resistance of 50Ω to avoid the need for a compensated probe. Because of the limited drive capability of the amplifier, we'll add a 5k resistor in series with the output. This results in a 100:1 attenuation (-40 dB) from the amplifier's output to the input of the VSA. The schematic of the amplifier is seen in Fig. 36.9. Note that we also added a resistor to ground on the input of the amplifier to avoid a floating node.

Figure 36.10 shows the input signal to the amplifier in the time domain. It is a 100 mV sinewave with a frequency of 100 kHz. The spectrum of this signal is seen in Fig. 36.11. Note the units on the y-axis are dBm or decibels with respect to 1 mW of power.

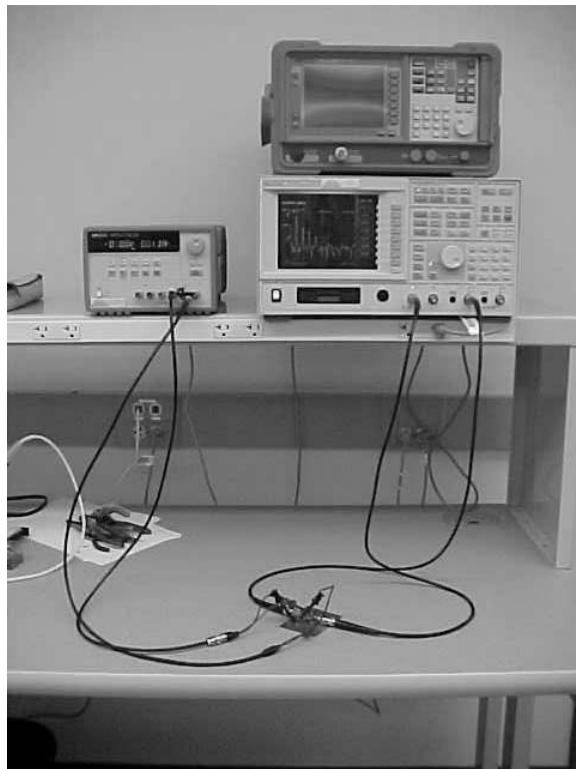


Figure 36.8 A test setup showing a VSA, spectrum analyzer (not in use), and power supply.

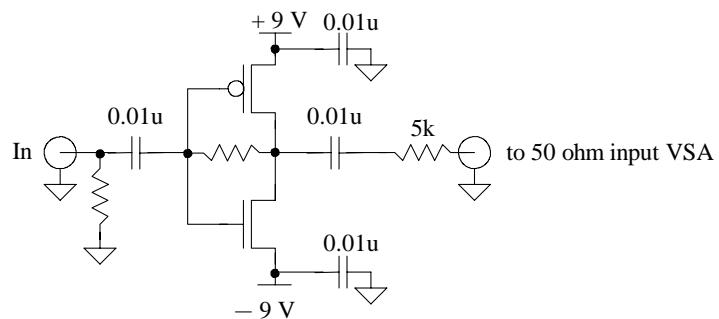


Figure 36.9 Final schematic of the push-pull amplifier shown in Fig. 36.3.

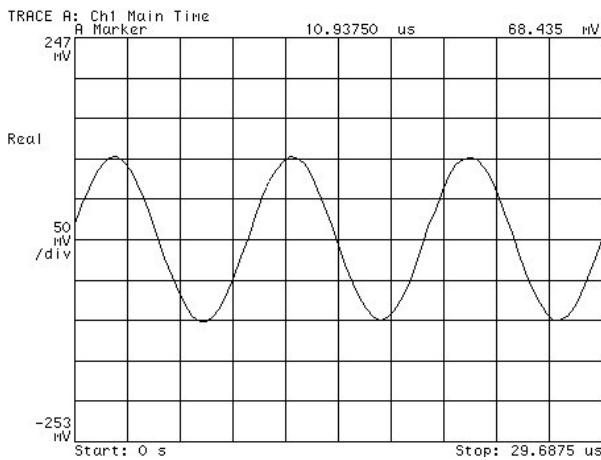


Figure 36.10 Input sinewave to the circuit of Fig. 36.9.

Because this is a 50Ω system, we can verify the power in the input sinewave is, as seen in Fig. 36.11, -10 dBm by writing

$$\text{dBm} = 10 \cdot \log \left(\frac{\overbrace{\text{RMS voltage of the input sinewave}}^2 / \sqrt{2}}{1 \text{ mW}} \right) / 50 \Omega = 10 \cdot \log \frac{0.1 \text{ mW}}{1 \text{ mW}} = -10 \text{ dBm}$$

(36.1)

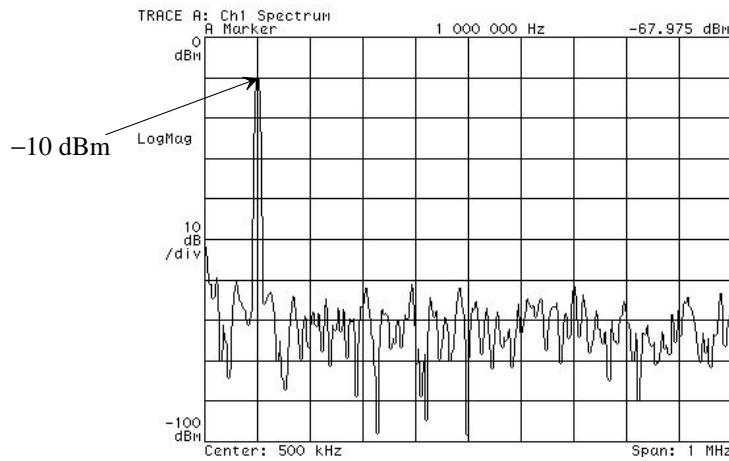


Figure 36.11 Spectrum of the input sinewave.

The spectrum of the amplifier's output is seen in Fig. 36.12. Keeping in mind that we have an attenuation of -40 dB between the amplifier's output and the VSA's input, we can estimate the amplitude of the output as the -37 dBm + 40 dB or 3 dBm. The gain is then 13 dB. This can be converted into a voltage amplitude (of the output sinewave) using

$$3 \text{ dBm} = 10 \cdot \log \frac{V_{\text{outpeak}}^2 / (2 \cdot 50)}{1 \text{ mW}} \rightarrow V_{\text{outpeak}} = 447 \text{ mV} \quad (36.2)$$

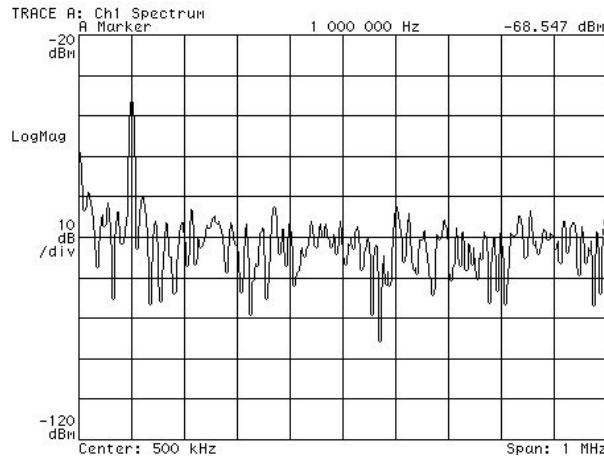


Figure 36.12 Spectrum of the amplifier's output with the input seen in Fig. 36.11.

Figure 36.13 shows the output spectrum if the amplitude of the input sinewave is increased to 1 V. Note the additional tones at multiples of the input frequency. The ideal resulting sinewave peak output amplitude is 4.47 V. Clearly this amplitude is well within the bounds of the power supply voltages. However, knowing the MOSFETs in the 4007 can supply only 1 to 2 mA and that our load is nominally 5 k (because of the added attenuating resistor seen in Fig. 36.9), we may run into some loading problems (resulting in the output becoming distorted). Further, we might expect some distortion simply because the amplifier is operating open-loop and, as indicated back in Ch. 22, the large signal gain varies with the input amplitude. Toward characterizing this distortion, we can specify the *total harmonic distortion (THD)*, as

$$THD = \sqrt{\frac{a_2^2 + a_3^2 + a_4^2 + \dots + a_n^2}{a_1^2}} \quad (36.3)$$

where a_1 is the amplitude of the fundamental, a_2 is the amplitude of the second harmonic (or the tone at twice the desired frequency), a_3 is the amplitude of the third unwanted tone, etc. The THD is usually specified as a percentage, e.g., 0.01% . We can determine the amplitude of the tones from the plot, neglecting the division by 1 mW (making the

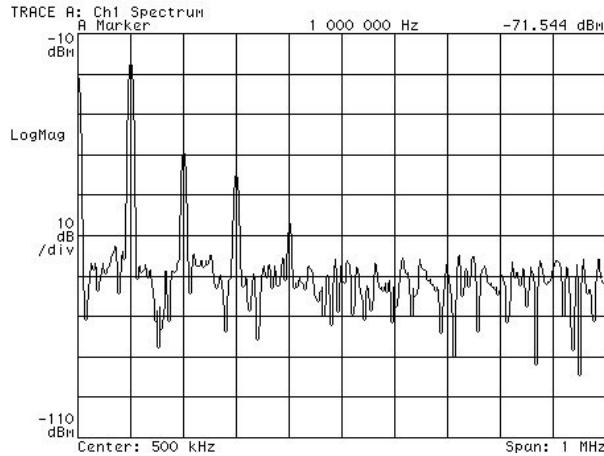


Figure 36.13 Output spectrum showing distortion when the input amplitude is increased to 1 V.

actual units dBm) because of the ratio in Eq. (36.3), as $-18 \text{ dB} = 10 \cdot \log a_1^2$ or $a_1^2 = 0.0159$, assuming the second harmonic's amplitude is -40 dB $a_2^2 = 0.0001$, assuming the third harmonic's amplitude is -45 dB , then $a_3^2 = 0.0000316$, and finally the fourth harmonic's amplitude is approximately 2×10^{-6} . The THD can then be calculated as

$$\text{THD} = \sqrt{\frac{0.1 + 0.0316 + 0.002}{15.9}} \rightarrow \text{THD} = 9.1\%$$

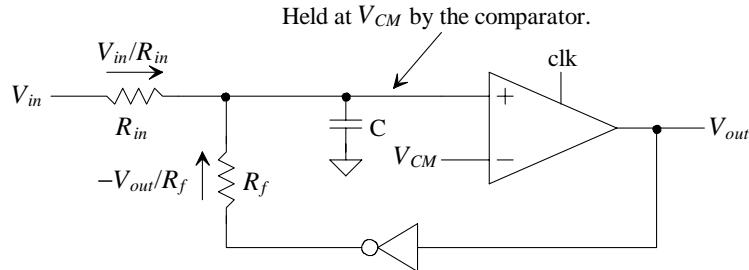
a large value (indicating that this isn't a good, low distortion, amplifier topology by itself).

36.2 A First-Order Noise-Shaping Modulator

Let's show how we can implement a simple noise-shaping modulator using a comparator, a capacitor, and a couple of resistors, Fig. 36.14. This type of modulator can be built using discrete components because we can precisely set the values of the resistors and capacitor. This topology may also find use in clever integrated versions of lower SNR NS data converters. When the circuit is operating correctly, the comparator holds its inverting input (the voltage across the integrating capacitor) at ground. Remembering from Ch. 32 that the forward gain of the modulator must be unity, we see that, because the gain of the integrator is much less than one over a significant portion of its operating frequency range, the performance of the comparator becomes very important. In order for the comparator to hold the voltage across the capacitor to a constant value, its gain must be very large. In the following analysis we assume infinite comparator gain, so the voltage across the capacitor is forced to zero by the feedback loop.

The input current (the input signal) can be written as

$$I_{in} = V_{in}/R_{in} \quad (36.4)$$

**Figure 36.14** A passive-integrator NS modulator.

while the feedback current can be written as

$$I_f = V_{out}/R_f \quad (36.5)$$

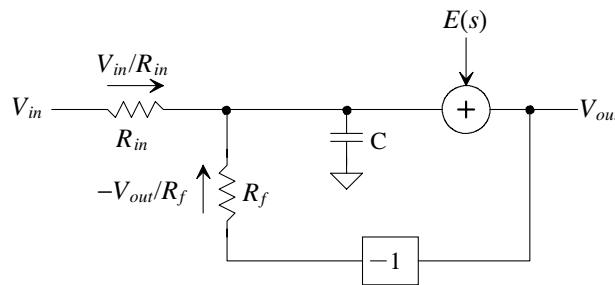
Note that we have not included the common-mode voltage, V_{CM} , in Eqs. (36.4) and (36.5). If the output of the comparator is 5 V (a logic 1) or 0 V (a logic 0), then the common-mode voltage is 2.5 V. Also note, as we'll show in a moment, the input signal amplitude can be scaled by adjusting the ratio of R_{in}/R_f .

Consider the block diagram of the modulator of Fig. 36.14 shown in Fig. 36.15. The comparator has been represented, as in Chs. 31 and 32, as a noisy circuit block. For example, assuming the inverting input of the compartor is precisely at 2.5 V and the noninverting input is at 2.6 V, the output of the comparator is 5 V and the quantization noise added to the signal, $E(s)$ (for this particular input sample), is 2.4 V. We can write

$$\left(\frac{V_{in}}{R_{in}} - \frac{V_{out}}{R_f} \right) \cdot \frac{1}{sC} + E(s) = V_{out} \quad (36.6)$$

After some manipulation, we can write

$$V_{out} = V_{in} \cdot \frac{R_f/R_{in}}{1 + sR_fC} + E(s) \cdot \frac{sR_fC}{1 + sR_fC} \quad (36.7)$$

**Figure 36.15** Block diagram of a passive-integrator NS modulator.

The desired signal is lowpass filtered, while the quantization noise is, again, highpass filtered (resulting in the modulation noise). Again, assuming the comparator gain is infinite, passing the output of the modulator through a digital filter with a bandwidth less than $1/2\pi R_f C$ results in a digital replica of the analog input signal. The practical problems with this topology are the importance of the comparator's gain and the kickback noise injected into the input signal when the comparator switches states.

Prototyping the Modulator

Figure 36.16 shows the schematic of the prototype modulator. The D flip-flop was added to make the LM339 comparator appear as though it were a clocked comparator. Also, the 74HC74 is implemented using CMOS and so its outputs swing all the way down to ground and up to +5 V. This is important when we use its output as the feed-back signal in our modulator. The resistors and capacitor on the input of the modulator form a lowpass filter (as seen in Eq. [36.7]) with a time constant of 100 μ s. The 3 dB frequency associated with this circuit is then 1.59 kHz. Input signal frequencies above this value will experience an attenuation. Figure 36.17 shows the deadbug prototype of the modulator.

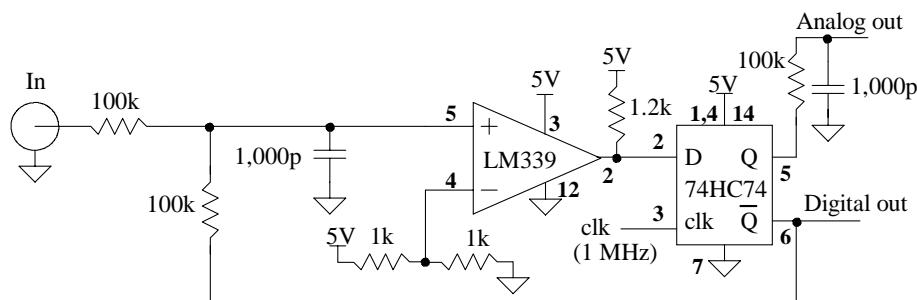


Figure 36.16 Schematic of the passive-integrator NS modulator.

The input to the modulator used to generate some test results, see Fig. 36.18, is a 4 V peak-to-peak sinewave at a frequency of 500 Hz centered around 2.5 V. The digital modulator output is shown in this figure as well. Looking at this digital data alone is somewhat meaningless using the oscilloscope (and so we'll look at the spectrum of this data). Figure 36.19 shows the spectrum of the digital data. A 3-foot coaxial cable is connected between the digital output in Fig. 36.16 and the VSA (with a 1 MEG input resistance so the loading will affect signal frequencies greater than roughly 100 kHz). Figure 36.20 shows the spectrum of the modulator's output up to 200 kHz. Note how, as seen back in Fig. 32.15, the modulation noise increases with increasing frequency. The resolution of the measurement, in Fig. 36.20, is 25 kHz. This causes the desired signal at 500 Hz to appear as though it were occurring at DC. Finally, the bottom trace in Fig. 36.18 shows what happens if we pass the digital data output from the modulator through an RC lowpass filter with a 3-dB frequency of 1.59 kHz. As expected, the resulting analog output is a very close replica of the input signal.

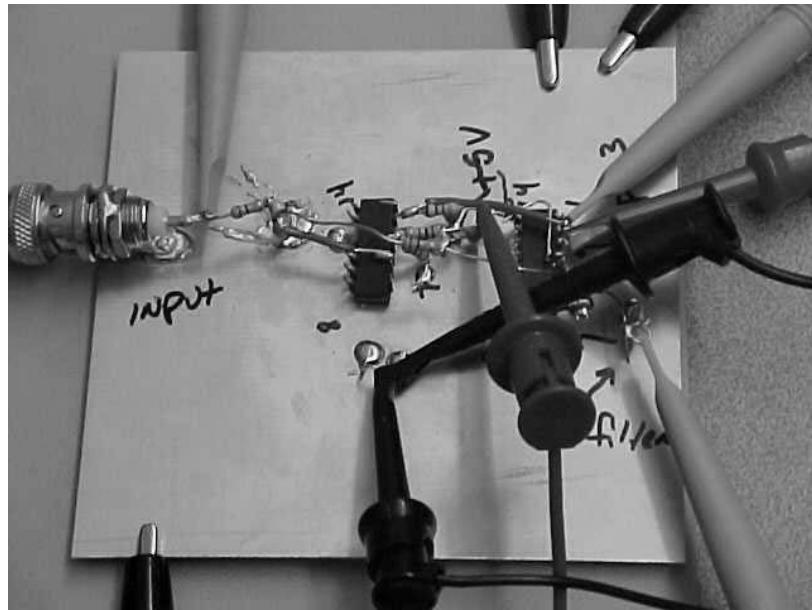


Figure 36.17 The prototype of the passive noise-shaping modulator of Fig. 36.16.

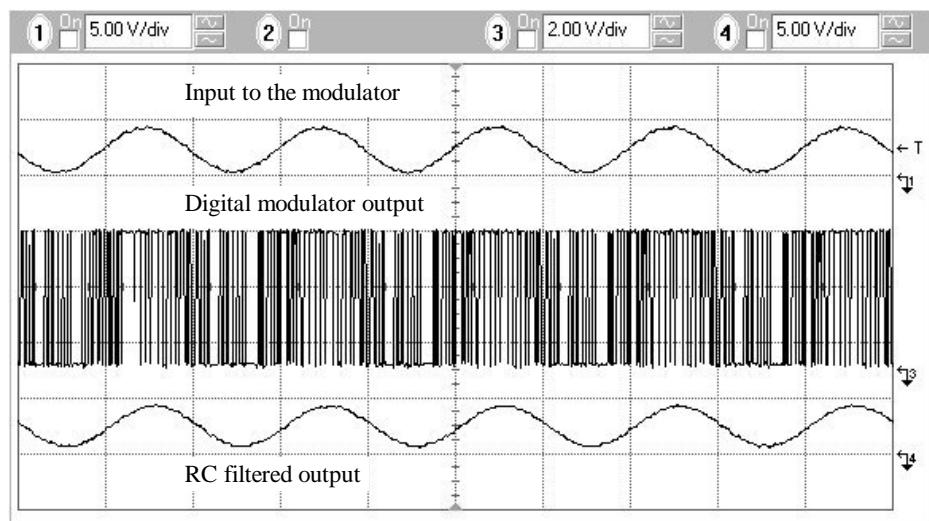


Figure 36.18 The outputs of the circuit in Fig. 36.16.

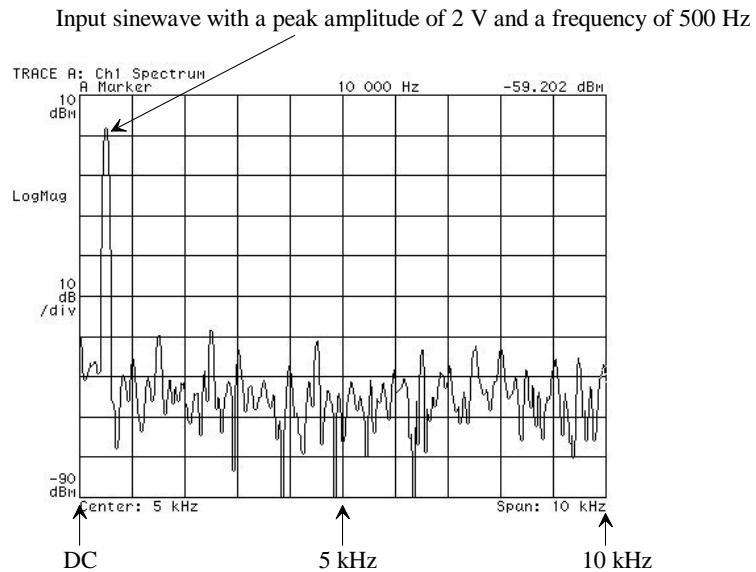


Figure 36.19 The base spectrum of the modulator's output data.

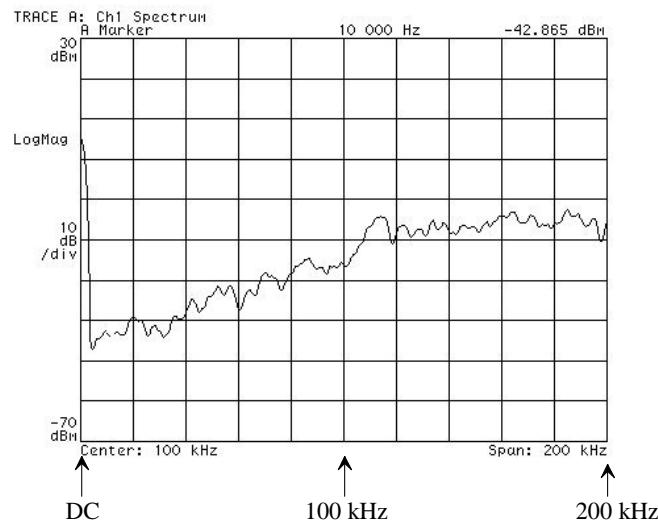


Figure 36.20 The spectrum of the modulator's output data up to 200 kHz.
Resolution bandwidth is 2.5 kHz (and so our input signal is smearing with DC).

36.3 Measuring 1/f Noise

Circuit noise was reviewed back in Sec. 33.3.3. In that section we showed that averaging thermal noise can be used to increase the SNR. Averaging, as seen in Chs. 30 and 31, can be thought of as lowpass filtering (and thus reducing the bandwidth of the signal and noise). We might wonder if averaging Flicker ($1/f$) noise results in a reduction of the circuit's input-referred RMS noise voltage. We'll show in this section that averaging a wideband signal has little effect on the input-referred contributions from $1/f$ noise.

Figure 33.85 showed the basic setup to measure $1/f$ noise. Figure 36.21 shows a lab setup used to measure the $1/f$ noise present in a submicron MOSFET. Because these devices aren't packaged, we need to use a probe station to pipe the bias signal on to the wafer and the noise signal off of the wafer.

The low-noise amplifier (LNA) shown in Fig. 36.21 is housed in a shielded box to avoid pickup of stray electromagnetic interference. Remembering that the LNA must both amplify the MOSFET's noise and bias the MOSFET to a specific operating point, we can sketch a possible LNA implementation, Fig. 36.22. The circuit is powered with 9 V batteries to avoid the possibility of regular bench power supplies injecting noise into the circuit and thus corrupting the measurement. The low-noise op-amp used, the OP-37, is configured in a gain of 100 configuration. A potentiometer is used to trim out the offset voltage of the op-amp. To see the basic op-amp's noise characteristics (without the

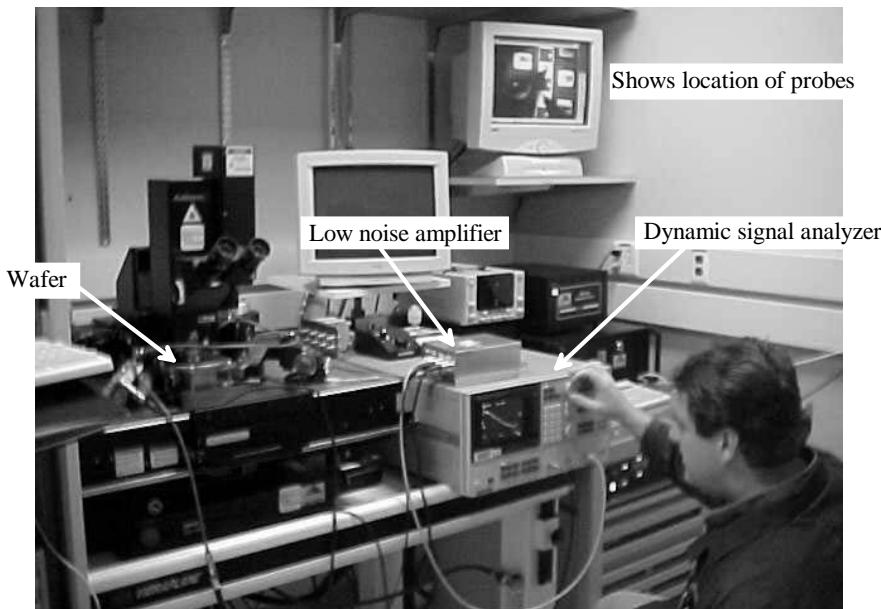


Figure 36.21 A lab setup used to measure Flicker noise.

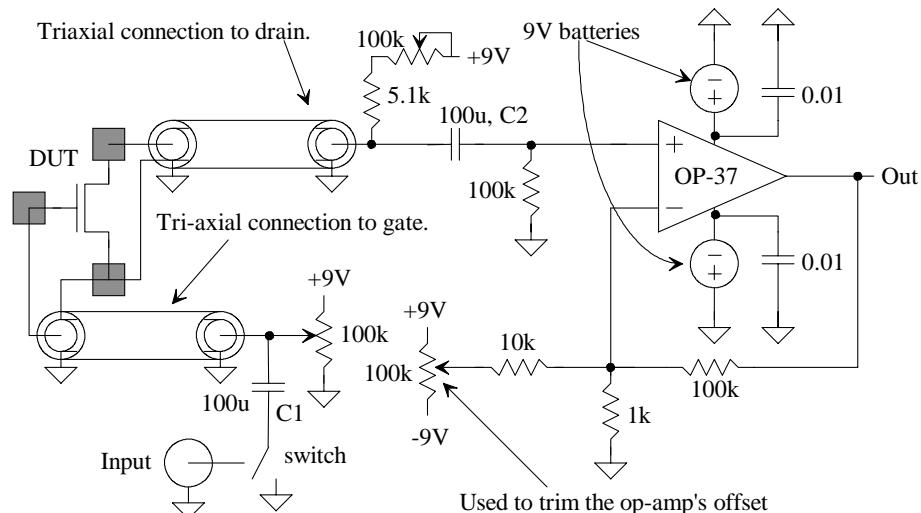


Figure 36.22 Schematic of an LNA.

MOSFETs connected), we simply remove C2 and connect the output to the spectrum analyzer (dynamic signal analyzer). The resulting spectrum is seen in Fig. 36.23. Note that the output spectral noise density of the LNA is roughly $-80 \text{ dBV}/\sqrt{\text{Hz}}$ ($100 \mu\text{V}/\sqrt{\text{Hz}}$) at 1 Hz (dividing this by the op-amp's gain of 100 results in $1 \mu\text{V}/\sqrt{\text{Hz}}$ at the MOSFET's drain). In the following, we ignore the LNA's contribution (to simplify the discussion).

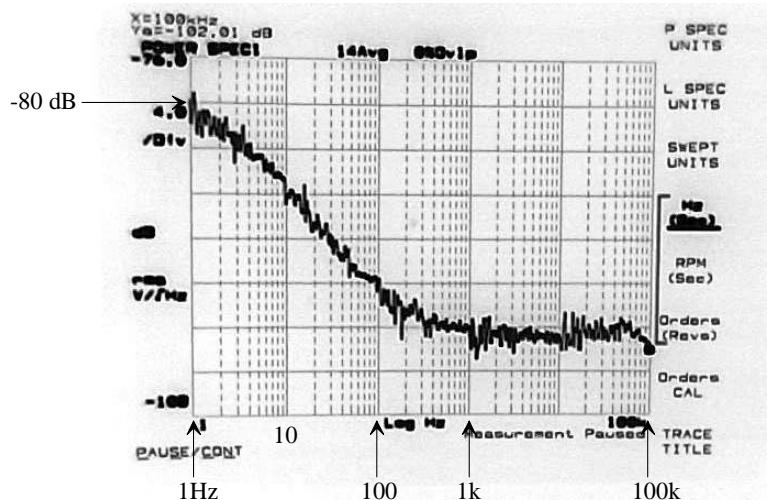


Figure 36.23 Measured noise characteristics of an LNA using the OP-37.

MOSFET Noise

Figure 36.24 shows the noise spectrum when we put C2 back in the circuit, bias the MOSFET at a specific operating point, and connect C1 to ground. The spectral density at 1 kHz is roughly $-70 \text{ dBV}/\sqrt{\text{Hz}}$ or

$$-70 \text{ dBV}/\sqrt{\text{Hz}} = 316 \mu\text{V}/\sqrt{\text{Hz}} \text{ at } 1 \text{ kHz} \quad (36.8)$$

or, calculating the $1/f$ spectral density (see Ch. 9)

$$\overline{v_{1/f,out}^2} = \left(\frac{316 \mu\text{V}}{\sqrt{\text{Hz}}} \right)^2 = \frac{100 \times 10^{-9} \text{ V}^2}{\text{Hz}} = \frac{\text{Flicker noise numerator (FNN)}}{1 \text{ kHz}} = \frac{\overbrace{100 \text{ pV}^2}^{\text{FNN}}}{f} \quad (36.9)$$

As a quick check, the spectral density of the noise at 10 kHz can be calculated as

$$\overline{v_{1/f,out}^2} = \frac{100 \text{ pV}^2}{10 \text{ kHz}} = \frac{10 \text{ nV}^2}{\text{Hz}} \rightarrow \sqrt{\overline{v_{1/f,out}^2}} = \frac{100 \mu\text{V}}{\sqrt{\text{Hz}}} = -80 \text{ dBV}/\sqrt{\text{Hz}} \quad (36.10)$$

which is what we see at 10 kHz in Fig. 36.24. (To determine the MOSFET's output noise spectral density alone we divide the spectral density in Fig. 36.24 by the LNA's gain of 100.) To determine the RMS output noise, we can integrate the $1/f$ noise spectral density

$$\sqrt{v_{on}^2} = \left[\int_{f_L}^{f_H} \overline{v_{1/f,out}^2} \cdot df \right]^{1/2} = \left[\text{FNN} \cdot \ln \frac{f_H}{f_L} \right]^{1/2} \quad (36.11)$$

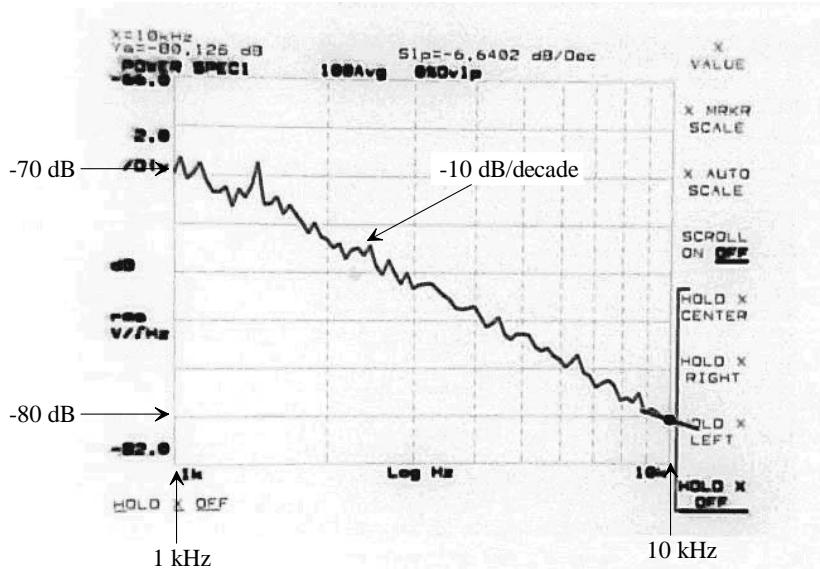


Figure 36.24 Measured Flicker noise from the MOSFET/LNA in Fig. 36.22.

This equation is fundamentally important to understand our statement at the beginning of the section, that is, averaging a wideband signal will have little effect on the contributions from $1/f$ noise to the input-referred or output RMS noise voltages. If we select the largest frequency, f_H , as 10 GHz (10^{10} Hz) and the lowest frequency as 1 Hz (once per second), then the natural log term in Eq. (36.11) is 23. However, if we change the lowest frequency of interest to 10^{-10} Hz (roughly once every 320 years), the natural log term increases to only 46! So to get a quick-and-dirty estimate of the contribution of $1/f$ noise to an output RMS noise voltage, we simply use

$$\text{contributions from } 1/f \text{ noise to RMS output voltage} = 7 \cdot \sqrt{FNN} \quad (36.12)$$

(knowing, of course, we can only add mean squared noise voltages). So, for the noise spectrum in Fig. 36.24, we can estimate the RMS output noise contributions as

$$\sqrt{v_{on}^2} = 7 \cdot \sqrt{100 \times 10^{-12} \text{ V}^2} = 70 \mu\text{V} \quad (36.13)$$

Again, this approximation is useful for wideband estimates of the RMS noise due to Flicker noise. It's not useful if a narrow bandpass filter is used on the output of a circuit where f_H and f_L are well defined.

Input-Referred Noise Voltage

While knowing the output noise is useful, it is generally more useful to refer this noise back to the input of the circuit so that it can be compared with an input signal. Towards this consider, in Fig. 36.22, connecting C1 to the BNC connector instead of ground. We can inject a signal, say a 1 mV sinewave, into the gate of the MOSFET and then look at the output of the circuit. If the overall gain of the circuit is 1,000 (= A), then we would see an output sinewave with an amplitude of 1 V. Knowing the gain of the circuit (MOSFET and op-amp), we can determine the input-referred noise by dividing the noise power spectral density with units of V^2/Hz , by A^2 or the noise voltage spectral density (or RMS output voltage) by A. Rewriting Eq. (36.12) for the input-referred RMS voltage

$$\text{Contributions from } 1/f \text{ noise to RMS input-referred voltage} = 7 \cdot \sqrt{FNN}/A \quad (36.14)$$

An important consideration when measuring the gain of the circuit is the frequency response of the gain. At the high end, the op-amp, in a gain of 100 configuration will have a bandwidth of approximately 10 kHz (assuming a gain bandwidth product of 1 MHz). Also, the MOSFETs have to drive the capacitance of the triax cables, which will result in an upper frequency roll-off in the amplifier's response. At the low end, C1 and C2 must be very large to keep the low-frequency roll-off point from becoming too large. The overall MOSFET/LNA's response has a bandpass shape. The point is that the input sinewave's frequency should be varied in order to find the passband gain of the circuit. Once C1 is grounded, its effect on the low-frequency roll-off is eliminated.

Clearly, $1/f$ noise can be a significant limiting factor when making sensitive measurements or when trying to attain large SNRs. Because averaging won't provide any help in reducing $1/f$ noise, let's show one very practical method that will help. While correlated double sampling (CDS) can be used here we discuss chopper stabilization (CHS). See reference [9] in Ch. 33 for additional information.

Chopper Stabilization

Consider the OTA shown in Fig. 36.25a and the associated noise spectral density shown in Fig. 36.25b. This circuit is essentially an integrator. In an ideal integrator, connecting the inputs together and to the common mode voltage would result in the outputs remaining unchanged. However, in a real integrator, the OTA's offset and the $1/f$ noise results in the outputs of the integrator eventually reaching the supply rails. It would be nice if we didn't have to worry about either the offset or the $1/f$ noise. What we are going to do in the CHS scheme is modulate the offset and noise to a place in the frequency spectrum where it won't interfere with our desired signal.

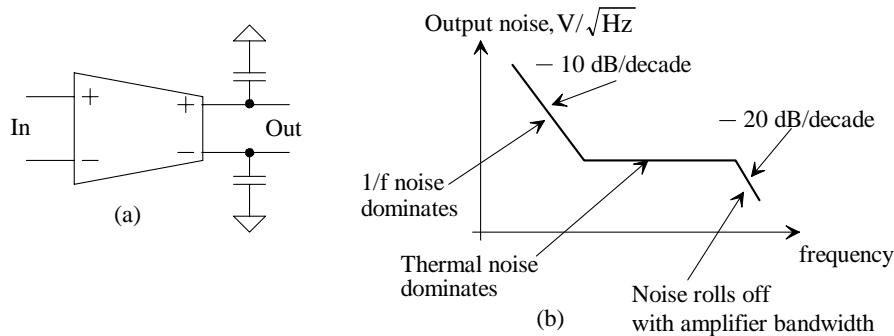


Figure 36.25 Integrator noise using an OTA.

Toward understanding this last statement consider the first-order noise-shaping modulator shown in Fig. 36.26. This topology is useful when measuring very small signals. It is very power-supply insensitive because of the current sources used. The noise and

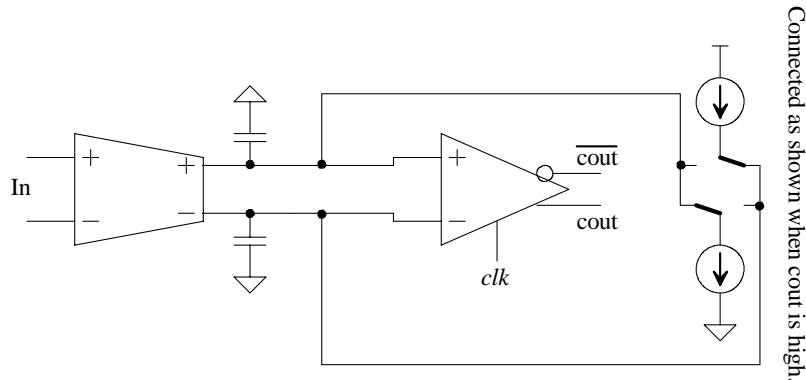


Figure 36.26 A first-order noise-shaping modulator using an OTA integrator.

offsets contributed by the OTA will directly affect the input sensitivity. Consider what would happen if we chopped (or switched back-and-forth) the input/output terminals of the OTA as seen in Fig. 36.27. When clk is high, the OTA is connected through switches so that it behaves as seen in Fig. 36.26. The OTA's offset, for example, causes a current to charge/discharge the capacitors. When clk is low, both the input and output terminals are switched so that the gain of the amplifier remains the same polarity. The offset now causes a current to flow in the capacitors in the opposite direction from the flow when clk was high. This effectively, if the rate at which we switch back-and-forth is fast, results in net zero current flow into the capacitors. A similar argument can be made for the low-frequency $1/f$ noise. The chopping, or switching, reduces both the offset and the Flicker noise on the output of the integrator (and so the input-referred noise is decreased as well).

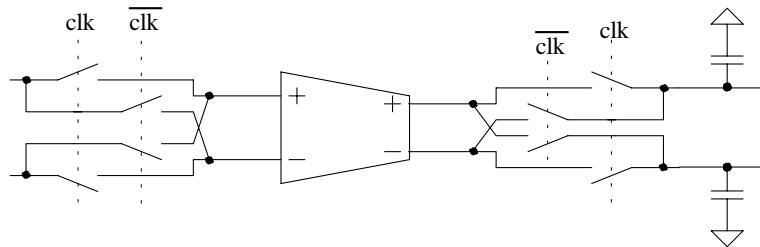


Figure 36.27 Switching (chopping) the inputs and outputs of the OTA integrator.

Figure 36.28 shows a possible implementation of the chopping switches. This should look familiar from Ch. 26. It was used to implement a multiplier. Here we are also using it for a multiplication. We are multiplying our OTA's input signal by +1 or -1 while doing the same to the OTA's output signal in order to maintain the same gain polarity. If the frequency we chop at is labeled f_{chop} , then we are multiplying the input by a square wave with this frequency. Looking at only the first harmonic of the waveform, we see this is simply amplitude modulation.

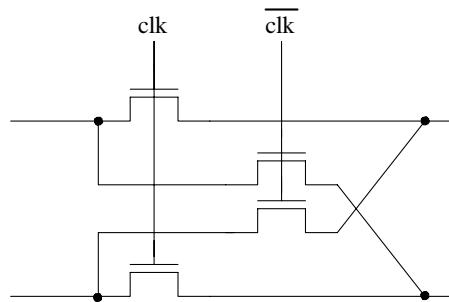


Figure 36.28 Showing a possible implementation of the chopping switches.

Consider the block diagram of the chopper and OTA of Fig. 36.27 shown in Fig. 36.29. Here we set the chopping frequency to one-half of the clock frequency (the clock used to strobe the comparator in the noise-shaping modulator of Fig. 36.26). We do this so that no aliasing occurs in our signal of interest from sampling the D signal at f_{clk} (the OTA noise doesn't fold into the signal of interest after sampling). If the settling time, when

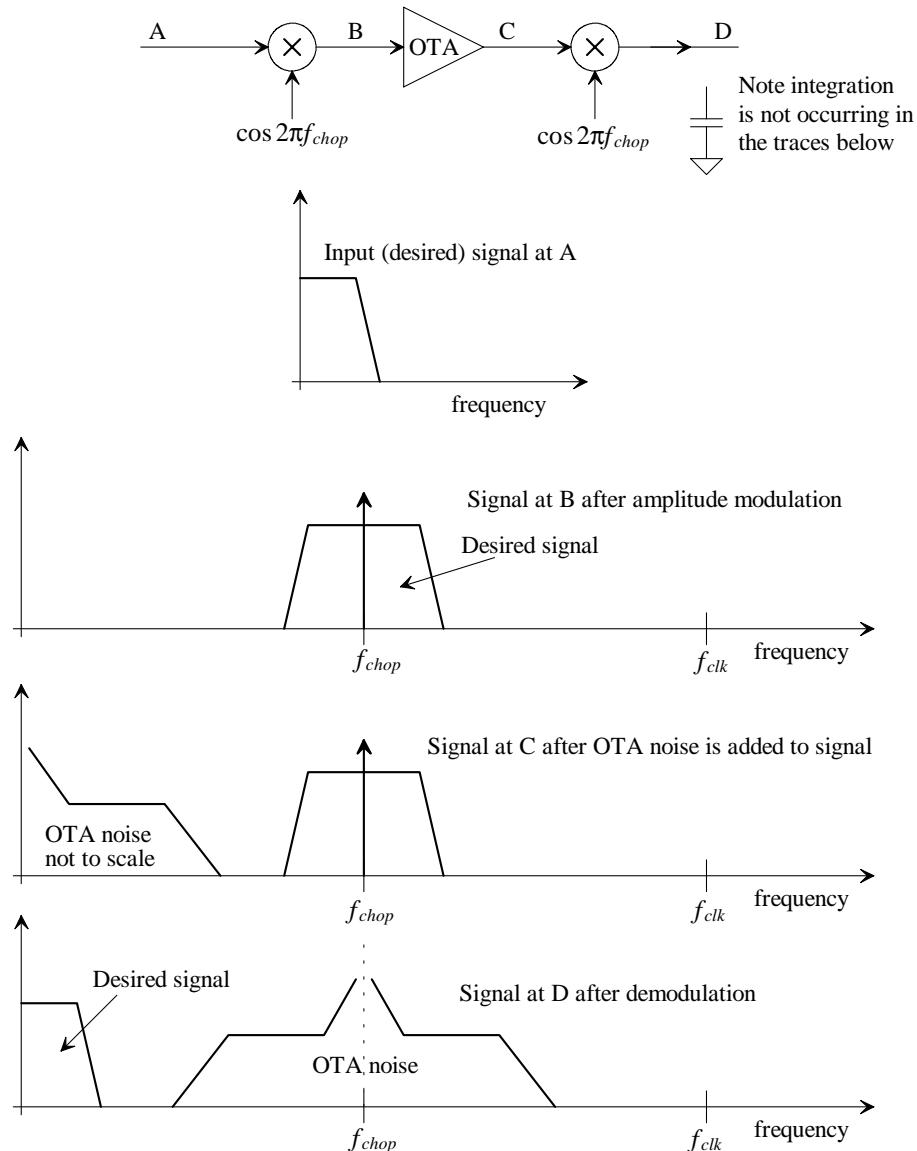


Figure 36.29 How chopping affects the noise and signal in an OTA.

chopping at f_{chop} , of the amplifier is longer than $2/f_{chop}$, then a lower chopping frequency can be used (ultimately set by the integrator's bandwidth). For example, if we are clocking the NS modulator of Fig. 36.26 at 100 MHz, then we might chop the OTA's input/output at a rate of 12.5 MHz (divide the NS modulator's clock by eight using a cascade of three of the circuits in Fig. 33.46).

In a second-order noise-shaping modulator the input-referred noise is mainly due to the first integrator as discussed in Ch. 32. The input-referred $1/f$ noise is passed directly to the output of the modulator. Modulators that use an autozeroed integrator don't have this problem because the autozeroing operation removes both the offset at DC and attenuates the $1/f$ noise spectral density. Looking at the power spectral density of $1/f$ noise on the output of an integrator, when not used in a modulator with feedback, results in a $1/f^3$ spectral shape. Averaging this noise results in linear growth with averaging time.

36.4 A Discrete Analog Integrator

Let's build a DAI-based first-order lowpass filter. Figure 36.30 shows the circuit (see Fig. 35.22). The charge stored on C_I when the ϕ_1 switches are closed is given by

$$Q_1 = C_I(v_{in}[(n - 1/2)T_s] - v_{out}[(n - 1)T_s]) \quad (36.15)$$

When the ϕ_2 switches close, this charge is transferred to the feedback capacitor, C_F ,

$$Q_1 = C_F(v_{out}[(n)T_s] - v_{out}[(n - 1)T_s]) \quad (36.16)$$

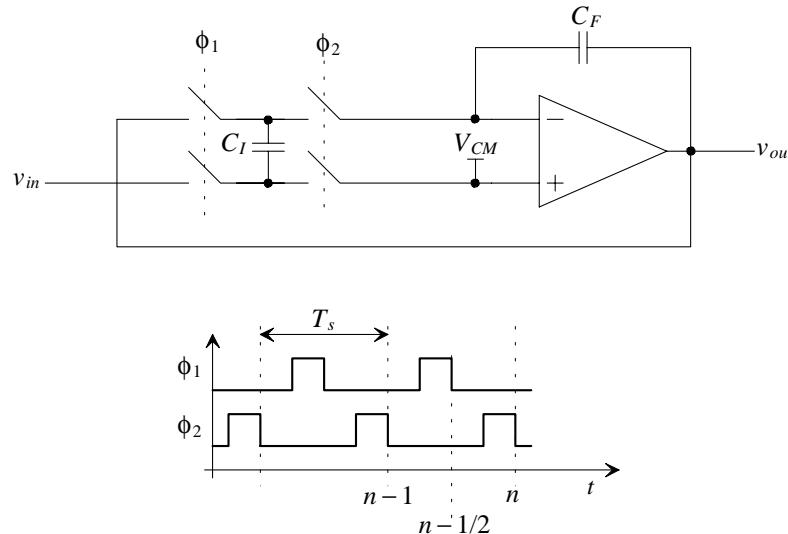


Figure 36.30 A first-order filter made using a DAI.

Taking the z-transform of this equation results in

$$C_I(V_{in}(z) \cdot z^{-1/2} - V_{out}(z) \cdot z^{-1}) = C_F(V_{out}(z) - V_{out}(z) \cdot z^{-1}) \quad (36.17)$$

$$C_I \cdot V_{in}(z) \cdot z^{-1/2} = V_{out}(z) \cdot (C_F - C_F \cdot z^{-1} + C_I \cdot z^{-1}) \quad (36.18)$$

or

$$\frac{V_{out}(z)}{V_{in}(z)} = \frac{z^{1/2}}{\frac{C_F}{C_I} \cdot z - \frac{C_F}{C_I} + 1} \quad (36.19)$$

noting the $z^{1/2}$ term in the numerator is simply a phase shift (a time delay), which will be neglected as long as our input frequencies, f , are much less than the filter's clocking frequency, f_s . Remembering from Eq. (35.69) that

$$z \approx 1 + \frac{s}{f_s} \text{ when } f \ll f_s \quad (36.20)$$

we can rewrite Eq. (36.19) as

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{1 + sR_{sc}C_F} \quad (36.21)$$

where

$$R_{sc} = \frac{1}{f_s C_I} \quad (36.22)$$

The filter's 3-dB frequency is located at

$$f_{3dB} = \frac{1}{2\pi R_{sc} C_F} \quad (36.23)$$

Clock Generation

The first thing we need to build is the clock generation circuit. Figure 36.31 shows the basic schematic of a nonoverlapping clock generator circuit. We use ± 9 V supplies. The two phases of the clock should transition between these voltages. We, again, use the 4007

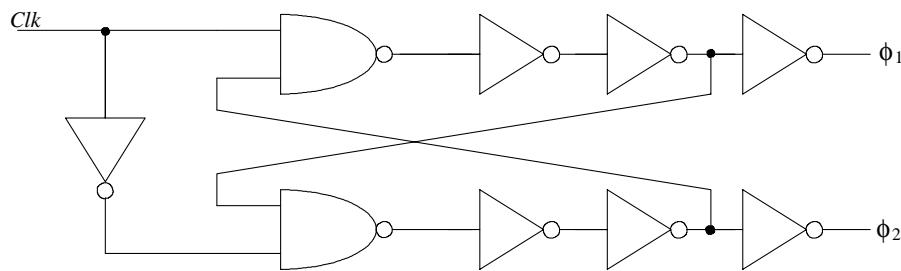


Figure 36.31 Nonoverlapping clock generation circuit.

CMOS transistors shown in Fig. 36.2 to implement the generator. Further, since this is a purely digital circuit, we breadboard the design (see Fig. 36.32). Figure 36.33 shows the outputs of this generator.

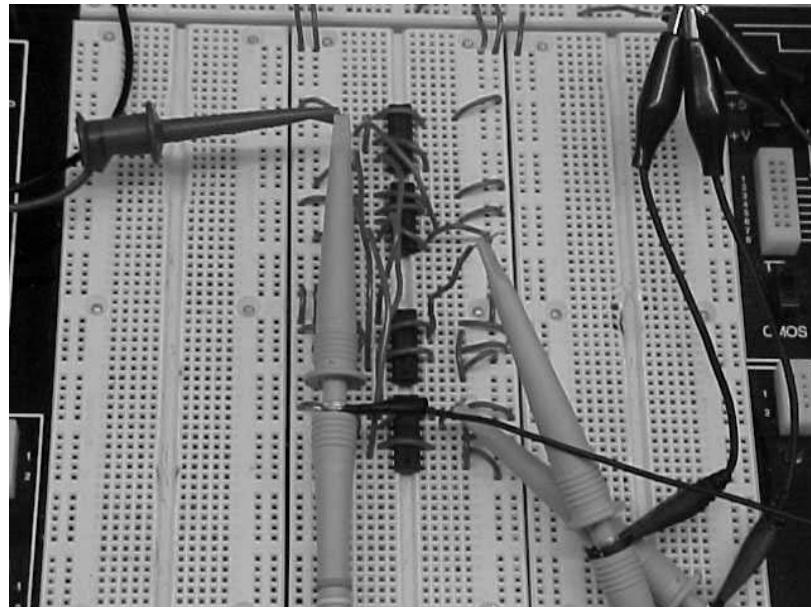


Figure 36.32 Breadboard of a clock generator.



Figure 36.33 Nonoverlapping clocks.

Prototyping the Filter

The schematic of our filter is seen in Fig. 36.34. The MOSFET switches are implemented using the 4007. The op-amp is an LT1365. Figure 36.35 shows the deadbug implementation of the filter.

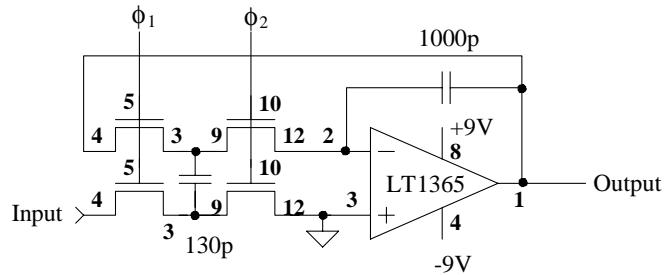


Figure 36.34 Schematic of the DAI-based filter.

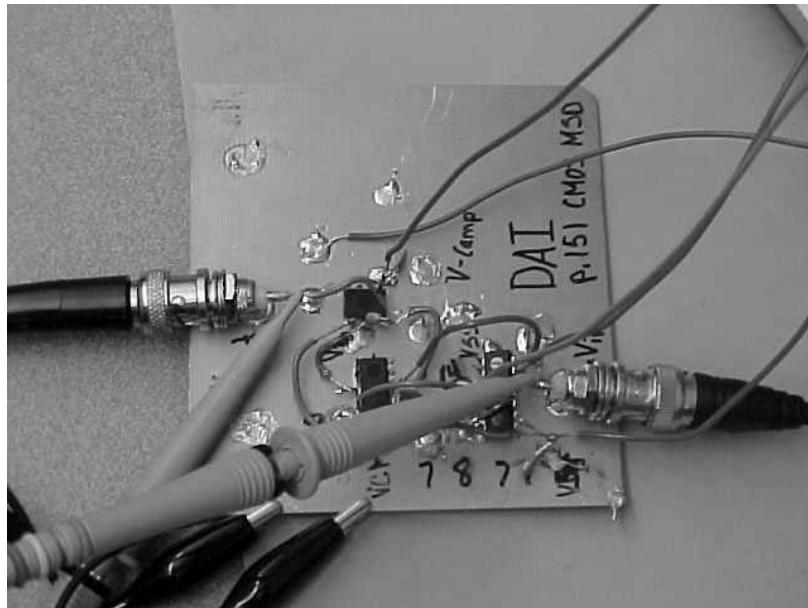


Figure 36.35 Deadbug prototype of the DAI filter in Fig. 36.34.

Using Eq. (36.23), we calculate the filter's 3-dB frequency as 2 kHz when the filter is clocked at 100 kHz. Figure 36.36 shows the filter's input and output at this frequency. If

one looks closely at the output signal, the discrete nature is obvious (see the steps in the output waveform shown in the simulation in Fig. 35.23). Figure 36.37 shows how the 130 pF capacitor (the node at the bottom of the schematic) charges to the input signal and then discharges back to ground (making the parasitic capacitance on this node unimportant).

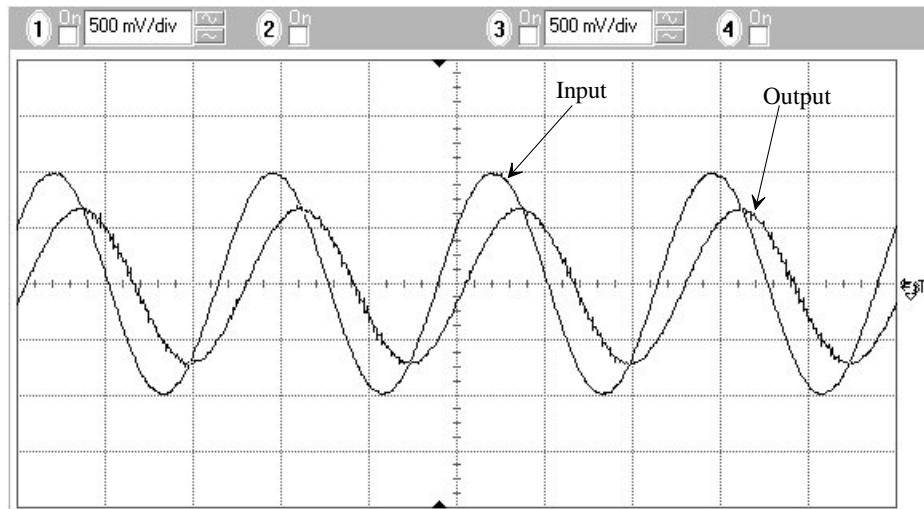


Figure 36.36 First-order filter's input and output at the 3-dB frequency of 2 kHz.

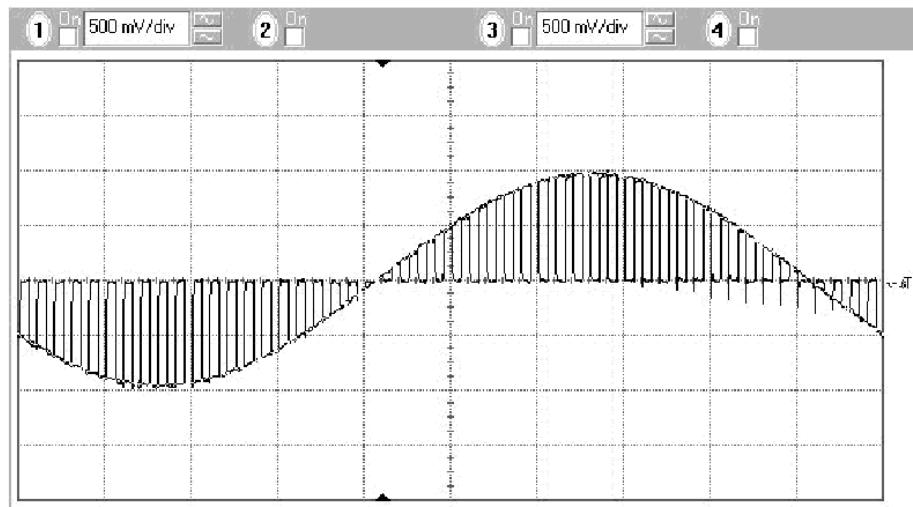


Figure 36.37 How the 130 pF capacitor charges and discharges.

Before leaving this section, let's show example input/output spectrums for this first-order switched-capacitor filter; Fig. 36.38. The desired signal is at 2 kHz and its peak amplitude has been decreased to 500 mV (to avoid overloading the VSA). The input signal amplitude is then $10 \cdot \log([(0.5/\sqrt{2})^2/1 \text{ M}\Omega]/1 \text{ mW})$ or -39 dBm . Because we are applying the 3 dB frequency, we expect our output amplitude to be -42 dBm (and it is). Finally, to show that the filter is indeed a sampled circuit, we increase the input frequency to 10 kHz and show the output images around the 100 kHz sampling frequency, Fig. 36.39.

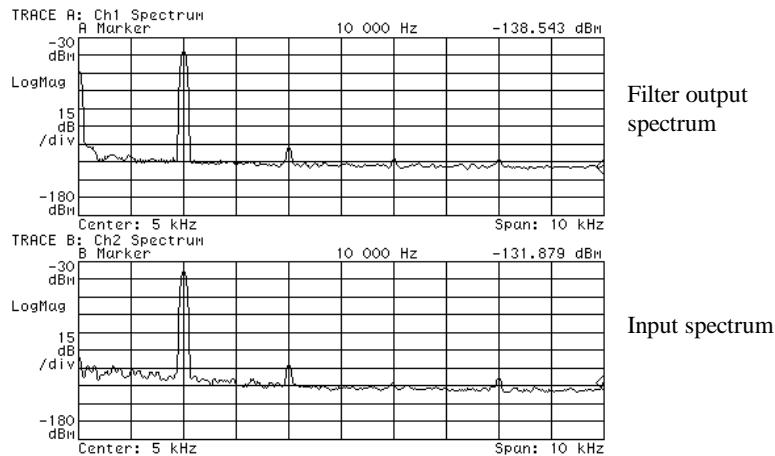


Figure 36.38 Input and output spectrums for the filter of Fig. 36.34.

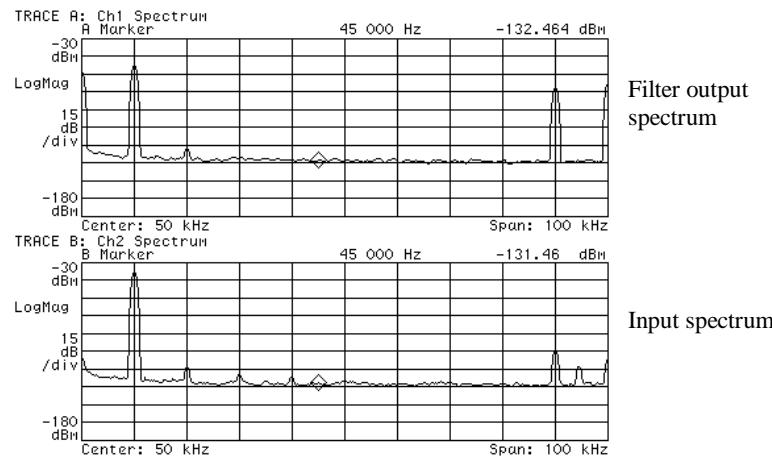


Figure 36.39 The spectrum up to the clocking frequency (100 kHz).

36.5 Quantization Noise

For the last section in this chapter, let's discuss, and show how to calculate, the quantization noise added to a spectrum from a data converter. We showed how to calculate the noise from a simulation spectrum back in Ch. 30; see Eq. (30.33). When presented with a data converter's output spectrum, Fig. 36.40, we can remove the desired signal (and perhaps the distortion if we calculate only the noise in the signal) and calculate the RMS quantization noise (or simply the noise in the spectrum) using

$$V_{Qe,RMS} = \sqrt{\int_{0 \text{ (DC)}}^{f_{\max}} V_{out}^2(f) \cdot df} \quad (36.24)$$

The signal $V_{out}(f)$ represents the data converter's output spectrum (after removing the desired signal and any distortion spikes) and has units of $\text{V}/\sqrt{\text{Hz}}$. The maximum frequency we integrate to, f_{\max} , is generally the Nyquist frequency, $f_s/2$. We assume that, when actually using the data converter, a reconstruction filter removes spectral content in the output signal above the Nyquist frequency. In a noise-shaping modulator, a digital filter sets f_{\max} . Note that we can't accurately calculate the quantization noise added to an input signal unless the input to the data converter is busy. A sinewave of sufficiently large amplitude can be used to exercise the data converter and "whiten" the quantization noise.

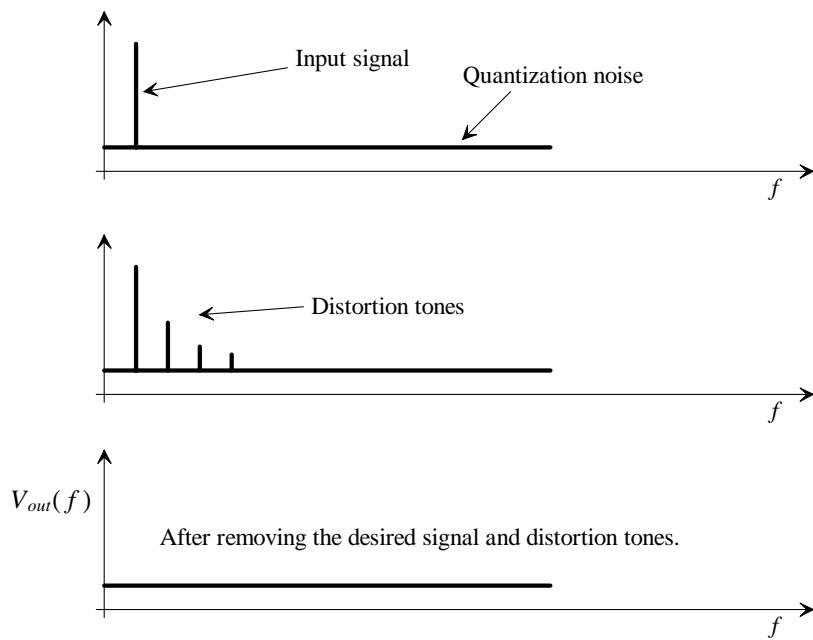


Figure 36.40 The output spectrum of a data converter.

Before going any further, let's show some example spectrums and the corresponding y-axis units. The top waveform in Fig. 36.41 shows the resulting spectrum when the input to the VSA is a 0.5 V (peak) waveform at 2 kHz. The RMS value of this waveform is 354 mV. In dBm (using a 1 MΩ VSA input resistance) this is $10 \cdot \log[(0.354)^2/1 \text{ M}\Omega]/1 \text{ mW}$ or -39 dBm; see the top trace in Fig. 36.42. The units for the top trace in Fig. 36.41 are volts, root-mean-square. Many of the spectrums we used in earlier chapters used peak voltages for the y-axis units.

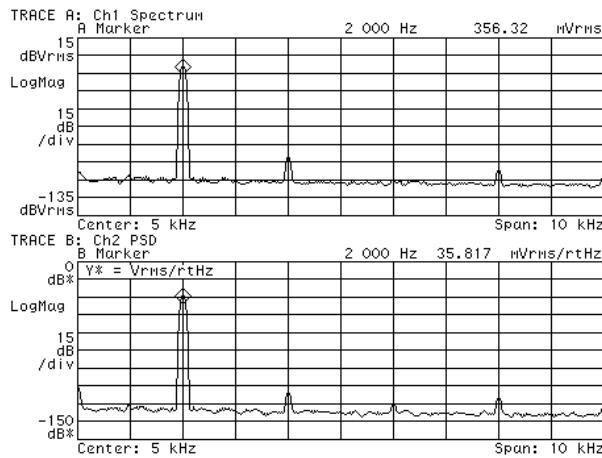


Figure 36.41 The relationship between units in spectral plots.

The bottom trace in Fig. 36.41 shows the voltage spectral density of our 0.5 V peak sinewave at 2 kHz. The units of a voltage spectral density are $\text{V}/\sqrt{\text{Hz}}$ (or more precisely volts RMS per root Hz). Because the same exact waveform is input to the VSA for each spectrum, we can relate the top and bottom waveforms in Fig. 36.41 simply by knowing the resolution bandwidth of the measurement

$$\frac{V_{RMS}}{\sqrt{\text{Hz}}} = \frac{V_{RMS}}{\sqrt{\text{Resolution bandwidth}}} \quad (36.25)$$

The resolution bandwidth used by the VSA, for the waveforms of Fig. 36.41, was 100 Hz. This means the amplitude of the sinewave is now $354 \text{ mV}/\sqrt{100 \text{ Hz}}$ or $35.4 \text{ mV}/\sqrt{\text{Hz}}$. In dB this would be $20 \cdot \log 0.0354$ or -29 dB. For a power spectral density, see the bottom trace in Fig. 36.42, we can use

$$\frac{\text{V}^2(\text{or Watts})}{\text{Hz}} = \frac{\text{V}^2(\text{or Watts})}{\text{Resolution bandwidth}} \quad (36.26)$$

Because our resolution bandwidth is 100 Hz, we expect the power spectral density to be 20 dB less than the power spectrum (top trace in Fig. 36.42) or -59 dBm/Hz. It should be obvious how to change from dBm/Hz to V^2/Hz .

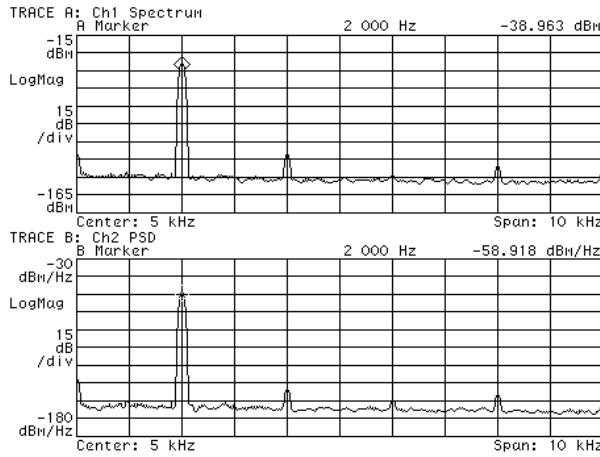


Figure 36.42 Power spectral density of the spectrums in Fig. 36.41.

Given a spectrum with noise (quantization, thermal, Flicker, or whatever), we can now generate the desired spectrum ($V_{out}[f]$ in Fig. 36.40) for calculating RMS noise. To illustrate how let's use the modulator output spectrum seen in Fig. 36.19. We begin by removing the desired tone in the spectrum (the sinewave at 500 Hz). Next we assume the noise is white (a flat spectrum) and has a value of -60 dBm. Because the VSA's input resistance is 50 ohms and the resolution bandwidth is 100 Hz, we can estimate the power spectral density of the noise using

$$10^{-60/10} = \frac{V_{RMS}^2 / 50}{1 \text{ mW}} \rightarrow V_{RMS}^2 = 50 \times 10^{-9} \text{ V}^2$$

and

$$\text{PSD} = V_{out}^2(f) = \frac{50 \times 10^{-9} \text{ V}^2}{100 \text{ Hz}} = 500 \times 10^{-12} \text{ V}^2/\text{Hz}$$

Using Eq. (36.24), we now need to estimate the maximum frequency used in the upper limit of the integration. Looking at the spectrum in Fig. 36.19, we see that the spectrum appears relatively constant over a wide frequency range. However, in any ADC we must use a reconstruction filter (or, for this case where a modulator is used, a digital averaging filter) to bandlimit the noise in the spectrum. For the modulator discussed in Sec. 36.2 a reasonable value of maximum frequency is 2 kHz (again set by a filter). Going above this frequency results, as seen in Eq. (36.7), in an undesired signal reduction (the signal sees the lowpass response of the integrator). The RMS noise in the modulator's output spectrum is then

$$V_{noise,RMS} = \left[\int_0^{2k} \frac{500 \times 10^{-12} \text{ V}^2}{\text{Hz}} \cdot df \right]^{1/2} = 1 \text{ mV}$$

The RMS value of the desired signal in Fig. 36.19 is 1.41 V. The SNR, for this modulator's output spectrum, is

$$\text{SNR} = 20 \cdot \log \frac{1.41 \text{ V}}{1 \text{ mV}} = 63 \text{ dB}$$

Using Eq. (31.5), the effective number of bits is roughly 10.

While these calculations are useful to illustrate how we manipulate data to calculate a SNR, it will be more useful to prototype an actual ADC and compare the quantization noise it adds to a signal to the values calculated theoretically.

Prototyping the ADC Circuit

In order to make our measurements practical and simple consider the circuit diagram shown in Fig. 36.43. An ADC and a purely resistive DAC are used to illustrate the noise (from the quantization process) added to an analog signal by the analog-to-digital conversion process. Using such a simple output DAC is useful as long as the ADC outputs swing from rail-to-rail and we use a relatively large resistance (say 10k) so the CMOS outputs can supply a current to the load resistors without a significant output voltage sag.

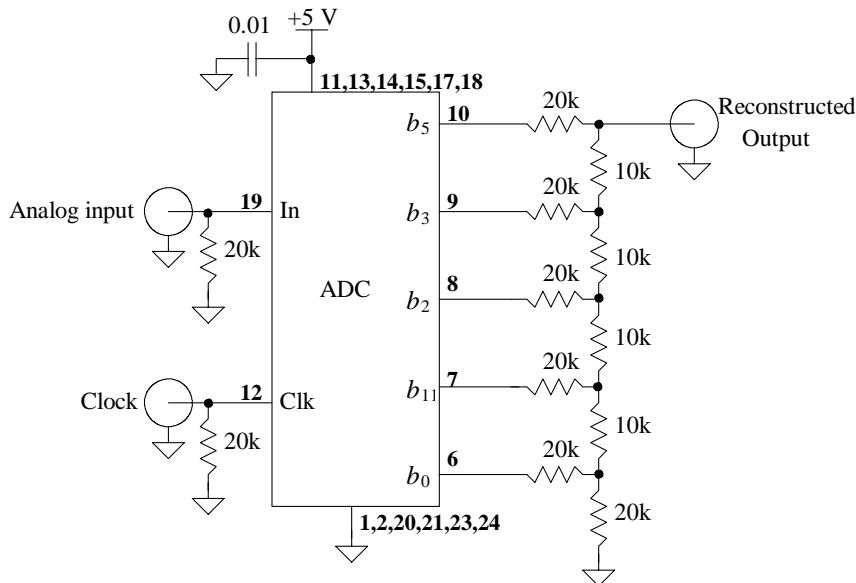


Figure 36.43 Schematic of an ADC and resistive DAC.

The ADC we selected is the TLC5540. It is an 8-bit ADC. However, we will only use the upper five bits of the ADC to illustrate the quantization process. The maximum reference voltage, V_{REF+} , is 5 V, while the minimum reference voltage, V_{REF-} , is ground. The clock pulse we'll use for our measurements will oscillate between ground and 5 V at 1

MHz. Because we are using five bits we can estimate the weighting of the LSB using Eq. (30.23) as 156 mV. Further, from Eq. (30.30), we can estimate the RMS value of the quantization noise as 45 mV. A picture of the prototyped ADC/DAC is seen in Fig. 36.44. The TLC5540 comes in a plastic small outline package (SOP). This SOP package is difficult to solder by hand in our deadbug prototyping scheme so we soldered it into a dual-in-line package (DIP) carrier. This makes prototyping the circuit much easier.

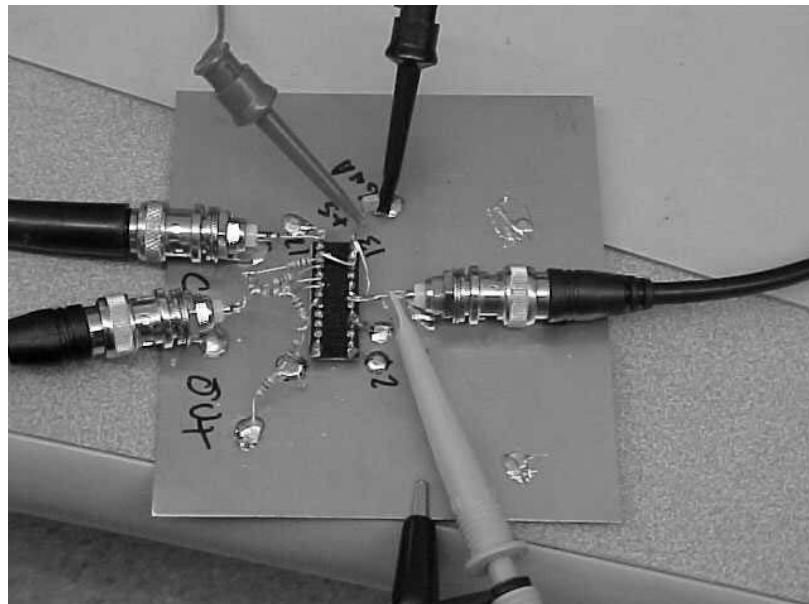


Figure 36.44 Photograph of the ADC/DAC prototype circuit.

Figure 36.45 shows the output spectrum for the ADC and resistive DAC seen in Fig. 36.43. Again, the clock frequency is 1 MHz. The input signal is a sinewave at 50 kHz with 0.5 V peak and centered around 2 V. The reason we don't see a DC signal in the spectrum is that the VSA's input was AC coupled. Again, the VSA's input resistance is 1 MΩ. Note how the spectrum rolls off with increasing frequency. We connected the output in Fig. 36.43 to the VSA's input through a piece of coax cable. The coax was 3 feet long and resulted in a capacitance of approximately 100 pF shunting the VSA's input, Fig. 36.46. If we model the ADC/DAC as a voltage source with 10k output resistance, then the frequency response of the measuring circuit is lowpass with a corner frequency of

$$f_{3dB} = \frac{1}{2\pi 10k \cdot 115 \text{ pF}} = 138 \text{ kHz}$$

Frequencies, in the output spectrum above this frequency will start rolling off at a rate of 20 dB/decade. The point is that instead of seeing a flat quantization spectrum (as in Fig. 30.57, for example), we will see a spectrum that rolls off with increasing frequency.

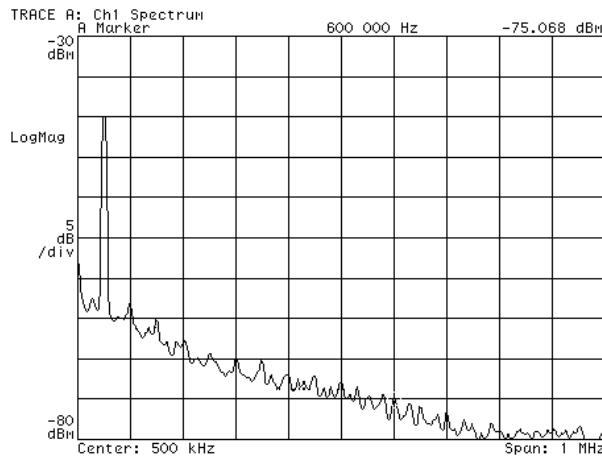


Figure 36.45 Output spectrum showing quantization noise for the 5-bit ADC in Fig. 36.43.

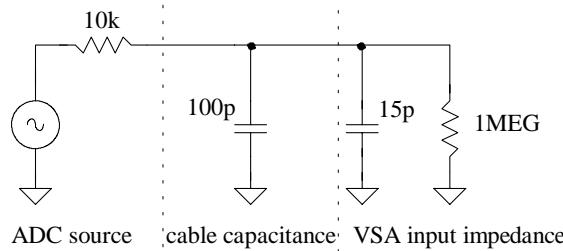


Figure 36.46 How loading affects the ADC's output spectrum.

Looking at Fig. 36.45 and knowing that the spectrum rolls off because of the measuring system, we can get an estimate for the quantization noise power at low frequencies as -65 dBm. Knowing the resolution bandwidth of the measurement was 10-kHz, we can estimate the power spectral density using

$$-65 \text{ dBm} = 10 \cdot \log \frac{V_{RMS}^2 / 1 \text{ M}\Omega}{1 \text{ mW}} \rightarrow V_{RMS}^2 = 316 \times 10^{-6} \text{ V}^2$$

or

$$PSD = V_{out}^2(f) = \frac{316 \times 10^{-6} \text{ V}^2}{10 \text{ kHz}} = 31.6 \times 10^{-9} \text{ V}^2/\text{Hz}$$

The Nyquist frequency is 500 kHz. If we again assume a filter is used to bandlimit the ADC output to the Nyquist frequency, we can calculate the RMS quantization noise as

$$V_{Qe,RMS} = \left[\int_0^{500k} 31.6 \times 10^{-9} \cdot df \right]^{1/2} = 125 \text{ mV}$$

This RMS noise is three times larger than what we calculated earlier (45 mV). We might speculate that the difference is due to not adequately randomizing the noise by using too high of an input frequency, relative to the sampling frequency, or too small an input amplitude (all of which are easy to verify at the bench).

Finally, let's show some time-domain waveforms showing quantization effects. Figure 36.47 shows the input and output waveforms when the input frequency is 5 kHz. Note how, as we calculated earlier, 1 LSB is 156 mV. Figure 36.48 shows the output when the input frequency is increased to 50 kHz, while Fig. 36.49 shows the circuit's inputs and outputs when the Nyquist frequency is applied to the circuit. Note how, as we would expect, the DAC output is simply a square wave at a frequency of 500 kHz. After this output is passed through a reconstruction filter with a frequency of just over 500 kHz we get our exact replica of the input signal simply shifted in time. While the signal in Fig. 36.45 was measured by providing a connection between the circuit and the VSA using a piece of co-ax cable, Fig. 36.46, the signals in Figs. 36.47 - 36.49 were measured using a compensated scope probe, Fig. 36.7. The significantly reduced loading resulting from using the compensated scope probe eliminates the spectrum roll off that was present in Fig. 36.45.

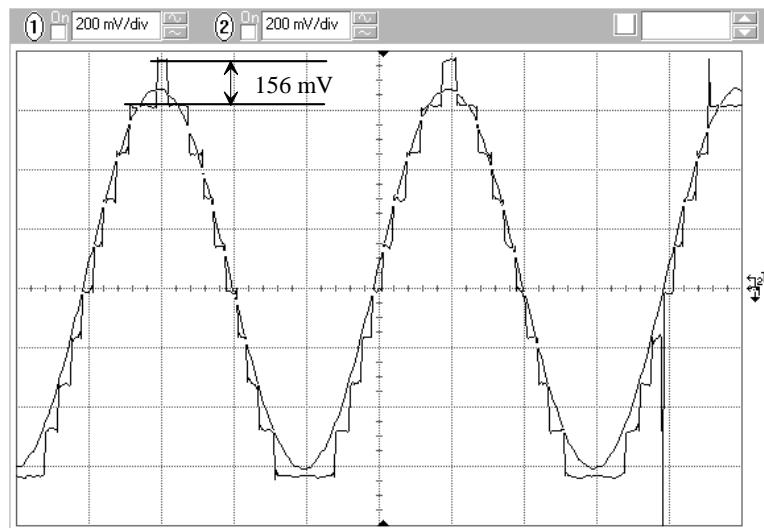


Figure 36.47 ADC input frequency of 5 kHz and the DAC output.

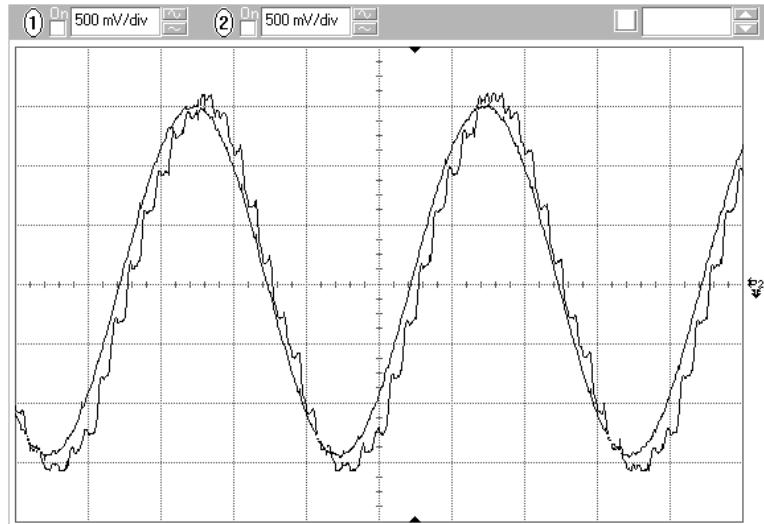


Figure 36.48 ADC input frequency of 50 kHz and the DAC output.

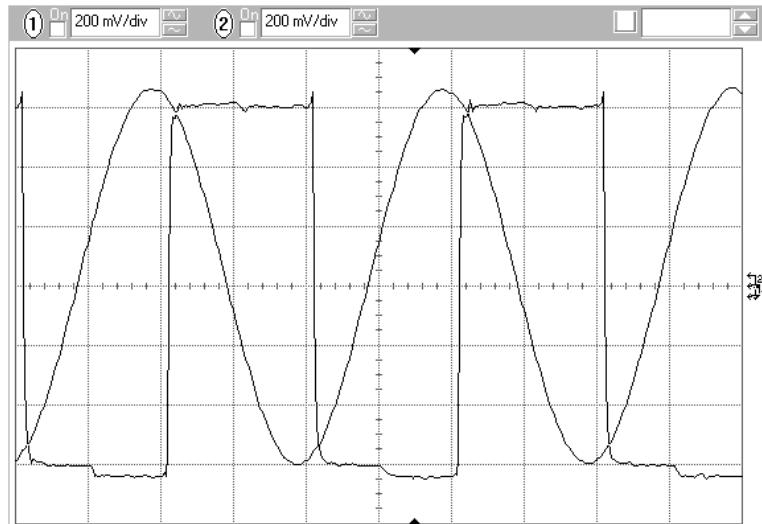


Figure 36.49 ADC input frequency of 500 kHz (the Nyquist frequency) and the DAC output.

While the circuits built in this chapter represent a small number of examples, relative to the material covered in the book, it is hoped that they are representative enough to make the engineer/student want to spend some time at the bench.

Index

A

AAF. *See* Antialiasing filter.
ABSTOL, 20, 24-25
Accumulate-and-dump circuit, 107-111
 cascading, 110-111, 167, 176-177
 See Digital averaging filter.
Accumulator, 116, 157-158
Accuracy, 96, 321, 329, *see also* Matching.
Active-RC Integrator, 395
 filter, 395, 419, 431
Adaptive filtering, 415
ADC. *See* Analog-to-digital converter.
Adder, 274-275
Algorithmic ADC, *see* Cyclic ADC
Aliasing, 3-4, 120, 122, 478
 quantization noise, 43
Analog RC filter, 178
Analog Sinc averaging filters, 169-171
Analog-to-digital converter (ADC), 26, 136
 1.5 bits/stage, 360-367
 averaging, 57
 calibrating, 372
 capacitor error averaging, 361, 368-376
 comparator placement, 352-354, 362, 374-376
 cyclic, 351-359
 decimating filters, 93, 106, 136
 digital error correction, 360
 ideal, 18, 19, 32-36
 implementing, 343
 modulator or coder, 136, 140-141
 pipeline, 24, 359
Antialiasing filter (AAF), 2-7, 94, 113, 120-121,
 125, 156, 393, 408, 415
Aperture jitter, 70
Autocorrelation function (ACF), 79
 discrete version, 81
Autozeroing, 342, 344
Auxiliary input port, 339-342
Average power, 62, 80-81
Average value, 82
Averaging, 54-57, 87-135, 299

bandpass filter, 132
circuits, 54, 89-91
clock jitter, 93-94
decimation, 106-126
digital filter, 106-114
filter, 167-171
highpass filter, 132
how it reduces noise, 55-57
improving signal-to-noise ratio (SNR), 87-105
interpolating filter, 126-131
limiting bandwidth, 92
linearity requirements, 56, 95-96
noise, 299, 472-474
op-amp, 370-372
quantization noise, 54-57, 89-92
without decimation, 111-112

B

B, bandwidth of the input signal, 73, 92
B device. *See* Nonlinear-dependent source.
Bandpass filter, 116, 132, 434
Bandpass modulators, 228-231
Bennett's criteria, 38, 52-53, 88, 127, 159
Beta-multiplier, 281
Biasing, 280-283
Bilinear transfer function, 418
Bin, 44
Binary offset, 105, 451
Biquad filters, 429-448
Biquadratic transfer function, 429
Boltzmann's constant, 299
Bottom plate sampling, 26, 343
Buffer,
 digital, 268
 push-pull, 284-286
Buried channel, 238

C

CAI, *See* continuous-time analog integrator.
Calibration, 326-327, 362
Canonic digital filter, *See* Digital filter.
Capacitance,
 input, 305
 metal, 241-244
 MOSFET, 239-241
Capacitive feedthrough, 340-341
Capacitor error averaging, 361, 368-376
Cascaded modulators, 221-227
Causal, 11

-
- Charge injection, 340-341
 Charge pump, 260-262
 Chopper stabilization, 306, 476-478
 Clock driver, 260-262
 Clock generation, 377-378, 480-481
 Clock jitter, 70-79, 82-86, 93
 averaging, 93-94
 defined, 70
 modeling with SPICE, 74
 oversampling, 73
 power, 83
 reducing stability requirements, 72
 related to resolution, 71
 spectrum, 76-77, 84
 stability, defined, 70
 CMOS, 235-238, 311
 analog circuit design, 276-298
 circuit design in a submicron process, *see Ch.*
 33
 digital circuit design, 255-276
 process flow, 236-239
 Coherent sampling, 45, 65
 Correlated double sampling, 306
 Comb filter.
 analog, 7-9
 digital. *See Digital comb filter.*
 simulating, 169-171
 Common-mode feedback (CMFB), 295-297,
 347-350
 dynamic, 382-385
 Common-mode noise elimination (CMNE),
 264-270, 296-297
 Common-mode rejection ratio (CMRR), 192, 290
 Common-mode voltage, 32, 136, 349
 Comparator,
 clocked, 155, 261-264, 469
 gain, 186, 351
 hysteresis, 186
 in an ADC, 352-354
 kickback noise, 263, 308, 353
 placement, 352-354, 362, 374-376
 SPICE modeling, 153
 Constant delay, 5
 Continuous-time analog integrator, 172, 395
 Continuous-time comb filter, 7-8, 169-171
 Continuous-time Fourier transform, 51
 Convergence, 25
 Convolution, 22
 Correlated double sampling (CDS), 306, 475
 Cosine window. *See Hanning window.*
 Counter, 134, 176-177, 216-217, 255
 ripple, 272-273
 up/down counter, 274
 See, also, accumulate and dump.
 Coupling noise, 385-388
 Current-mode DAC, 312-314, 316-327, 332-334
 Current steering DAC, 332-334
 Cyclic ADC, 351-359
-
- D**
- DAC. *See Digital-to-analog converter.*
 DAI. *See Discrete analog integrator.*
 Data converters,
 implementing, *see Ch.* 34
 modeling, *see Ch.* 30
 SNR, *see Ch.* 31
 dBc (decibels with respect to the carrier), 68, 86
 DC measurements, 176
 Deadbug, 460-461
 Decimation, 106-126, 166
 bandwidth limitations, 92, 120-126
 decimating filter, 93, 106, 136
 Delay elements, 271-273
 Delta modulation, 140
 Delta-sigma modulation, 154 *See noise-shaping.*
 Demodulator, 136
 Diff-amp, 276, 290-291
 Differential nonlinearity, 311-334
 improving using segmentation, 321-322
 Differentiator, 115
 Digital averaging filter, 57, 106-126
 Digital comb filter, 116-119, 128, 132, 135
 Digital differentiator, 115
 Digital error correction, 360, *see also* Analog-to-digital converter (ADC), 1.5 bits/stage
 Digital filter
 biquad, 446-448
 canonic, 425-429
 characteristics, 101, 421-429
 finite impulse response (FIR), 116
 infinite impulse response (IIR), 116
 removing modulation noise, 164-166,
 212-213, 448-451
 stability, 122-124
 Digital integrator, 102-104, 114, 416
 Digital resonator, 132-135
 Digital signal processing (DSP), 1, 415
 Digital-to-analog converter (DAC), 41, 125-131,
 136, 140, 485-492
 calibration, 326-327
 current steering, 332-334
 demodulator or decoder, 136
 feedback, 174-179
 ideal, 26-32
 interpolating filter, 93, 127, 136

-
- pipeline, 392
 $R\text{-}R$, 245-248, 312-331
 segmentation, 321-322
 selecting capacitors, 306
 wide-swing, 316-327
 without an op-amp, 328-334
 Dirac delta impulse, 9-10
 Discrete analog integrator (DAI), 136-139, 151, 154-155, 157, 173, 185, 198, 226, 393, 411-415, 418, 479
 feedback gains, 202-205
 fully-differential, 191
 noise performance, 306-307
 Discrete Fourier transform, 13-14, 51
 loading data using WinSPICE, 67
 Distortion, 66-69, 189, 197, 284, 347, 466-467, 484-485
 Dither, 38, 97-99, 180-181
 DNL. *See* Differential nonlinearity.
 Droop, 110, 112
 defined, 213
 DSP. *See* Digital signal processing.
 Dump and interpolate, 127-131
 Dynamic CMFB, 382-385
 Dynamic range, 68-69

E

- E device. *See* Voltage-controlled voltage sources.
 Effective number of bits (ENOB), 64, 67
 EKV model, 236, 248-253
 Error feedback, 217-220
 Excess gate voltage, 276-277
 Expected value, 82

F

- Filters,
 active-RC, 395-402, 431-434, 439-440
 accumulate and dump, 106-111
 antialiasing, 2-7, 94, 113, 120-121, 125, 156
 bandpass, 116, 132-135, 228, 418, 434-435, 437-442, 457, 475
 bilinear, 418-428
 biquad, 429-448
 comb filter, 7-9, 116-117
 decimating filters, 106-126
 digital. *See* Digital filter.
 exact frequency response of discrete-time filter, 416-417
 finite impulse response (FIR), 116

- g_m -C filters. *See* transconductor-C filter.
 highpass, 132, 116, 132, 214-215, 218, 401, 418, 420, 469
 in a higher-order modulator, 215
 infinite impulse response (IIR), 116
 integrating filters, 102-103, 124
 interpolating filters, 126-129
 ladder filter, 457
 lowpass, 9, 52, 59, 106, 114, 116, 133-134, 156, 159, 164, 179, 190, 201, 205, 208, 214-215, 228, 301, 415, 469, 472, 479-484
 MOSFET-C, 404-406
 Q peaking and instability, 443-445
 Sinc, 109
 SNR, 403-404
 stability in digital filters, 122-124
 switched-capacitor, 411-415, 420-421, 437, 440-441, 479-484
 transconductor-C filter, 407-411, 445-446
 tuning, 393, 404, 407, 411, 420, 456
 Finite impulse response (FIR) filter, 116
 First-order noise shaping, 154-193
 FIR. *See* Finite impulse response.
 Flicker noise. *See* Noise.
 FM. *See* Frequency modulation.
 Forward modulator gain, 182-186
 Fourier transform, 11, 17, 21, 51, 80-81
 Frequency modulation (FM), 74
 f_t (transition frequency), 279-280
 Fully-differential, 191, 295-297

G

- Gain-enhancement, 288, 290
 Gain margin, 335
 Gaussian probability distribution, 82, 84-85
 Glitch area, 331-332
 GMIN, 20

H

- Half-band digital filter, 125, 131
 Hanning window, 14, 47-48
 Harmonics, 191
 square wave, 77
 Highpass filter, 132
 Higher order modulators, 210-215
 HSPICE, 290
 Hysteresis, 186

I

- I, in phase or real component, 230
- IIR. *See* Infinite impulse response filter.
- Implementing data converters, *see* Ch. 34
- Implementing,
 - ADCs, 343-387
 - Impulse sampling, 2-18, 22
 - Infinite impulse response (IIR) filter, 116
 - INL. *See* Integral nonlinearity.
 - Integral nonlinearity, 311-334
 - Integrator
 - analog. *See* Continuous-time analog integrator.
 - digital, 102-104, 114
 - discrete analog integrator (DAI), 136-139, 151, 173, 411-415, 479-484
 - gain, 182-186, 198-201
 - initial conditions, 104
 - leakage, 186
 - Interpolation, 12, 126-131
 - interpolating filter, 93, 127, 136
 - Interpolative modulators, 217-218
 - Inverse Fourier transform, 11

J

- Jitter. *See* Clock jitter.

K

- K, oversampling ratio or averaging factor, 56, 87, 92, 97, 106-129
 - clock jitter, 72-74, 94
- Kickback noise, 263, 308, 353

L

- Ladder filter, 457
- Latch, 271-275
- Layout,
 - pipeline ADC, 385-387
 - resistors, 247-248
- Leakage,
 - cascaded modulators, 222
 - integrator, 186-188
- Least significant bit (LSB), 27, 63, 174
 - See* Question 30.14
- Limit cycle oscillations, 179-180
- Linear phase, 5, 91

- Linearity requirements for averaging, 56, 95-96, LOCOS, 236

- Lowpass filter (LPF), 3, 393-394, 479-484
 - active-RC, 395-404
 - bilinear, 418-429
 - biquadratic, 429-448
 - MOSFET-C, 404-406
 - g_m -C, 407-411
 - switched-capacitor, 413-417
- LPF. *See* Low pass filter.

- LSB. *See* Least significant bit.

M

- MASH modulator. *See* Noise-shaping
- Matching,
 - capacitors, 368-374
 - R-2R DAC, 317-318
 - resistors, 245-248, 330-331
- Matlab, 67
- Mean, 82
- Miller integrator, 395
- Mixed-signal system
 - block diagram, 2
- Modulation noise (removing), 164-166, 212-213, 448-451
- Modulator, 136, 140-141
 - bandpass, 228-231
 - cascaded, 221-227
 - first-order, RMS quantization noise, 162-164
 - fully-differential, 193
 - gain, 207
 - higher order, 210-215
 - input and output, 156
 - interpolative, 217
 - multibit, 215-221
 - noise-shaping, 140-141, 185
 - predictive, 140
 - second-order, 221-223
 - sigma delta, 154
 - third-order, 223-225
 - topology, second-order, 195
- Monotonic, 57, 330-331
- MOSFET, 25
 - biasing, 280-283
 - binary weighted current mirrors, 249
 - buried channel, 238
 - capacitance, 239-241
 - EKV model, 236, 248-254
 - fabrication, 236-238
 - floating capacitor, 241
 - g_m (transconductance), 279

f_T (transition frequency), 279-280
 long-L, 255
 noise, 303-305
 r_o (small-signal output resistance), 277-278
 scale, 254
 surface device, 238
 switch, 256-261
 $W\text{-}2W$ current mirror, 250
 Multibit modulators, 215-221
 Multiplexer (MUX), 118, 128, 158, 165, 221, 360, 362, 376
 Multipliers, 451-453
 Multiply by 2, 343-345, 452
 Multiply by 0.5, 452
 Multistage noise shaping modulator (MASH), 221-227
 MUX. *See* Multiplexer.

N

Native MOSFET capacitor, 239
 Natural MOSFET capacitor, *See* native MOSFET capacitor.
 Noise
 averaging, 299, 472-475
 cascaded amplifiers, 305-306
 chopper stabilization, 306, 476-478
 circuit, 298-307
 clock jitter, 85, 299
 correlated double sampling (CDS), 306, 475
 coupling, 385-388
 flicker ($1/f$), 230, 238, 304, 472-478
 MOSFET, 303-304
 noise equivalent bandwidth, 301-302
 of a random voltage, 145
 quantization noise, *See* Quantization noise.
 RMS calculations, 42, 44, 53, 55-56, 67, 80
 485, 491
 thermal, 238, 298, 472
 white, 97
 Noise equivalent bandwidth, 301-302
 Noise floor, 24
 Noise transfer function, 141, 154, 158-159, 187, 194-195, 214-215
 Noise-shaping (NS), 140-141
 analog implementation, 172-173, 467-471
 averaging filters, 167-171
 bandpass modulators, 228-231
 cascaded modulators, 221-228
 comparator gain, 182-186
 comparator offset, 186

constructing a high-resolution ADC, 136, 155-156
 continuous-time, 172-173, 467-471, 476-479
 implementing DACs, 217-221
 DAC linearity, 215
 data converter, *see* Ch. 32
 decimation filters, 106-126, 156, 166
 demodulator, digital first-order, 157-158
 differences between Nyquist converters, 177, 180
 dither, 180
 distortion, 189
 error feedback, 217-220
 feedback DAC, 174, 215
 first introduced, 217
 first-order, 154-193, 467-471
 fully-differential, 192-193
 fundamentals, 149-210
 higher order modulators, 210-215
 input-referred noise, 190
 integrator gain, 182-183
 integrator leakage, 186-188
 interpolating filters, 126-131, 157
 interpolative modulators, 217-218
 leakage, 186, 222
 limit cycle oscillations, 179
 linearizing, 201
 MASH, 221-227
 modulation noise, 159-162
 modulators, 140-141, 154
 multibit modulators, 215-221
 offset, 189, 476-477
 op-amp gain, 186-188
 OTA, 476
 passive, 467-469
 pattern noise, 179
 ripple, 177-179
 second-order, 194-209
 settling time, 188-189
 slewing, 189
 SNR, 163, 166, 194, 208-211
 stability, 183, 199-200, 213, 221
 topologies, 210-213
 transconductor, 476
 Nonlinear dependent source, 28
 Nonmonotonic, 57, 330-331
 Nonoverlapping clock generation, 261-262, 377-378
 Nutmeg, 13
 Nyquist
 differences between noise-shaping converters, 177, 180
 frequency, 3, 5, 7, 16, 51, 58, 68, 142

rate, 59, 142
rate data converter, 177

O

Offset, 99, 149, 175, 186, 189, 198, 263, 290, 306, 314-315, 323, 378-382, 477
autozero, 342, 344
calibrating DAC offset, 323-325
chopper stabilization, 476-479
common-mode, 378-380
comparator, 352-353
input-referred offset, 384, 479
op-amp, 339-342
random, 290
removing using an auxiliary input port, 339-342
storage, 384
systematic, 290
Offset binary format, 35, 104-105, 118, 147, 220-221
One-zero transitions, 181
Op-amp, 284-297, 349-350
auxiliary input port, 339-342
averaging, 370-372
biasing, 280-283
common-mode voltage variation, 350
compensation, 290-294
data converters, 334-337
differential outputs, 295-296, 345-350
floating current source, 286-287
frequency response, 335
gain, 186, 290-295, 337-338
gain-enhancement, 288, 290
gain margin, 335
input-referred noise, 190-191, 473-478
minimum open-loop gain, 337-338
modeling, 151-152, 192
noise, 306-307
offset, 189, 290, 307, 476
output swing, 284, 357-359
phase margin, 335
push-pull out, 284
S/H, 343-350
settling time, 188-189, 287-295
slew-rate, 188-189, 287-290
simulating, 292-293
stability, 286, 290-295
unity gain frequency, 339
Operational transconductance amplifier (OTA), 25, 476-478
OTA. *See* Operational transconductance amplifier.
Output swing, 357-359

Overdrive voltage, 276
Overflow, 119
Oversampling, 140
Oversampling ratio. *See* K, oversampling ratio or averaging factor.

P

Pattern noise, 179-181
PDF, probability density function,
 average value (mean or expected value), 82
 Gaussian, 84, 98, 298
 standard deviation, 83
 thermal noise, 298
 uniform, 82
 variance, 83
Phase-delay relationship, 6
Phase-locked loop (PLL), 71
Phase,
 discrete analog integrator (DAI), 138, 151-152
 digital filter, 100-103
 distortion, 5, 115
 excursion (peak), 75
 linear, 5, 60, 115
 margin, 293, 334-335, 347
 noise, 78, 85-86
 response, 6-9, 22, 90-91, 114, 115, 334, 396,
 399-402, 409, 418, 425-426, 434
 shift, 55, 80-81, 156, 480
Picket-fence effect, 44-45, 47
Pipeline,
 ADC, 24, 32-35, 67, 312, 343, 359-361, 368,
 374-376
 adder, 274-275
 DAC, 392
 Layout, 385-386
PLL. *See* Phase-locked loop.
Power
 average, 80-81, 300
Power spectral density (PSD), 62, 80, 85, 161-162,
 486, 490
 jitter, 85
 See also Spectral density.
 quantization noise, 91-92
 WinSPICE, 160-162
Power supply sensitivity, 476
Predictive coder. *See* Predictive modulator.
Predictive modulator, 140
Probability density function, *See* PDF.
PSD. *See* Power spectral density.
Push-pull amp, 284-286, 459-467

Q

Q (pole quality factor), 429-434
 high Q, 438-443
 peaking and instability, 443-445
 Quadrature or imaginary component, 230
 Quantization noise, 32, 35-57, 485-492
 as a random variable, 42
 calculating from a spectrum, 50
 defined, 35
 estimating, 220
 power, 62
 reducing quantization noise, 54
 spectral density, 52-57
 viewing the spectrum, 37-52
 Quantize, 18
 Quantizer, 158, *See also* Analog-to-Digital converter.

R

R-2R, 245-248, 312-331
 topologies for DACs, 312-332
 Random offset, 290
 Random signals
 quantization noise, 42
 spectral density, 82-86
 standard deviation, 83
 variance, 83
 RCF. *See* Reconstruction filter.
 Reconstruction filter (RCF), 3-4, 9-12, 24, 125-126, 130-131, 135, 157, 220
 RELTOL, 20, 24-25
 Resolution, 87
 loss because of jitter, 71-74
 Resistor,
 layout, 247-248
 properties, 245
 Resolution bandwidth, 486, 490
 Return-to-zero (RZ), 20-22
 r_o (small-signal output resistance), 277-278
 Ripple, 177-180
 RMS quantization noise voltage, 42, 50, 53, 55-56, 87-88, 92
 calculating from a spectrum, 43, 88, 91-92
 Root mean square (RMS) voltage, 80
 RZ. *See* Return-to-zero.

S

Sample and hold (S/H), 19-26, 39, 343-350

matching of the capacitors used, 349, 368-374
 single-ended to differential, 345-350
 SPICE model, 19
 subtraction, 354-359
 Sample frequency reduction. *See* Decimation.
 Sampler, 2
 Sampling and Aliasing, 2-26
 coherent. *See* coherent sampling.
 Sampling gate, 2
 Scale option, 254
 Second-order noise-shaping, 194
 Segmentation, 321-322
 Settling time, 188, 291-295, 319, 329, 334-335, 339
 SFDR. *See* Spurious-free-dynamic range.
 SFFM. *See* Single frequency frequency modulation.
 S/H. *See* Sample and hold.
 Shallow trench isolation (STI), 236-238
 Side lobes, 109-110
 Sigma delta modulation, 140, 154. *See also* Noise-shaping
 Signal-to-noise and distortion ratio(SINAD). *See* Signal-to-noise plus distortion ratio (SNDR).
 Signal-to-noise plus distortion ratio (SNDR), 64, 66-67, 69
 Signal-to-noise ratio (SNR), 63, 69
 data converter, *see* Ch. 31
 first-order modulator, 163
 higher-order modulators, 211
 ideal, 63-64, 208-209
 improving using averaging, 87-136
 improving using feedback, 136-141
 measured, 64
 second-order modulator, 194
 selecting input sinewave, 65
 specifying, 69
 spectral leakage, 65-66
 Signal transfer function (STF), 141, 154, 195
 Silicide, 236-238, 245-246
 Simulation program with integrated circuit emphasis. *See* SPICE.
 Sinc filter, 108-110, 112 *See* digital averaging filter.
 SPICE implementation, 169-171
 Sinc function defined, 11
 magnitude response, 22
 Sinc waveform, 46-48
 Single-ended to differential S/H, 345-350
 Single-frequency frequency modulation (SFFM), 74
 Single-ended to differential conversion, 345-350
 Slew-rate, 189, 287-290
 SOP, small outline package, 489
 Smoothing filter, 3 *See* reconstruction filter.

-
- SNR. *See* Signal-to-noise ratio.
 SNDR. *See* Signal-to-noise plus distortion ratio.
 Spectral analysis using SPICE (the spec command), 13-16
 Spectral density, 79-86
 - power spectral density (PSD), 80, 85
 - quantization noise, 52-57, 88
 - random signals, 82-86
 - voltage spectral density, 80
 Spectral leakage, 45-46, 65-66
 SPICE, 1, 13-15
 - ABSTOL, 20, 24-25
 - AC analysis, 8-10
 - ADCs and DACs, 26-35
 - B device, 28
 - behavioral models, 149-153
 - comparator modeling, 153
 - DAI modeling, 151-153, 192-193
 - digital filter, 447-451
 - E device, 13
 - EKV model, 236, 248-253
 - HSPICE, 290
 - impulse sampler, 13-14
 - jitter model, 74-79
 - level 1 model, 235-236
 - linearize before a spectral analysis, 14
 - modeling approach, 28
 - models for ADCs and DACs, 26-35
 - nonoverlapping clocks, 150
 - nutmeg, 13
 - op-amp (ideal), 13, 151, 192
 - op-amp (MOSFET), 290-293
 - pulse statement, 13, 19, 29
 - RELTOL, 20, 24-25
 - sample-and-hold (S/H) model, 19
 - scale, 254
 - S/H spectral response, 20
 - Sinc filter, 169-171
 - sinewave, 152-153
 - spectral analysis (spec command), 13-16
 - step sizes, 15, 78
 - subcircuit netlist, 30, 34
 - switch, 13, 150
 - .tran statement, 15
 - VNTOL, 20, 24-25
 - Spike, 69, 77-78
 - Spur, 69
 - Spurious free dynamic range (SFDR), 64, 68, 180
 - Square wave, 77
 - Stability,
 - digital filter, 123
 - filter, 442-445
 - noise-shaping modulator, 183, 199-200, 213, 221
 - op-amp, 286, 290-295
 - Standard deviation, 83-84, 298
 - STI, *See* shallow trench isolation.
 - Submicron CMOS circuit design, *see* Ch. 33
 - Subtraction, 119, 354
 - Switched-capacitor filters, 411-415, 420-421, 437, 440-441
 - Switches, 150
 - capacitive feedthrough, 340-341
 - charge injection, 340-341
 - MOSFET, 256-260, 319
 - Systematic offset, 290
-
- T**
- THD, *See* total harmonic distortion.
 Thermal noise. *See* noise.
 Thermometer code, 361
 Thermometer decoder, 322
 Time domain description of reconstruction, 9-13
 Time-interleaved operation of an ADC, 359-360
 Total harmonic distortion, 466
 TPSC, *See* true single-phase clocking.
 Transconductor-C filters, 407-411
 - bilinear, 419-420
 - biquad, 445-446
 Triangular response, 168
 True single-phase clocking, 272-273
 Tuning, 325, 393, 400, 404-411, 415, 419-420
 - orthogonal tuning, 325, 420, 457
 Two's complement, 105, 117-119, 220, 231, 451-453
 - subtraction, 119
-
- U**
- Understanding output swing, 357
 Unit step function. *See* $u(t)$.
 $u(t)$, 17-18, 21
-
- V**
- Variance, 83, 299
 V_{CM} , 32, 136 ($VDD/2$ or 0.75 V in this book)
 VDD , 59 (1.5 V in this book)
 VSA (Vector Signal Analyzer), 463-464
 VNTOL, 20, 24-25
 Voltage coefficient, 245-246, 333
 Voltage-controlled voltage source, 13, 204, 401

Voltage spectral density, 62, 80, 475, 485-486

Von Hann window. *See* Hanning window.

V_{SP} , Inverter switching point, 266, 296-297

VSS, 60 (ground or 0 V in this book)

W

W-2W current steering DAC (mirror), 250-251,
333-334

White noise, 97, 303, 485

Windowing, 13-14, 46-48

WinSPICE, *See also* SPICE.

 loading external data to perform a DFT, 67

Wireless, 230

 I/Q extraction, 230-231

Z

z

 defined, 17

 domain, 16-17, 141

 to s-transform (practical), 424, 480

z -plane, 99-102

Zeroes padding, 131

About the Author



Russel Jacob (Jake) Baker (S'83 - M'88 - SM'97) was born in Ogden, Utah, on October 5, 1964. He received the B.S. and M.S. degrees in electrical engineering from the University of Nevada, Las Vegas, and the Ph.D. degree in electrical engineering from the University of Nevada, Reno.

From 1981 to 1987, he was in the United States Marine Corps Reserves. From 1985 to 1993, he worked for E. G. & G. Energy Measurements and the Lawrence Livermore National Laboratory designing nuclear diagnostic instrumentation for underground nuclear weapons tests at the Nevada test site. During this time he designed over 30 electronic and electro-optic instruments including high-speed (750 Mb/s) fiber-optic receiver/transmitters, PLLs, frame- and bit-syncs, data converters, streak-camera sweep circuits, micro-channel plate gating circuits, and analog oscilloscope electronics. From 1993 to 2000, he was a faculty member in the Department of Electrical Engineering at the University of Idaho. In 2000, he joined a new electrical engineering program at the Boise State University. Also, since 1993, he has consulted for various companies and laboratories including the Lawrence Berkeley Laboratory, Micron Display, Amkor Wafer Fabrication Services, Tower Semiconductor, Rendition, and the Tower ASIC Design Center. Since 1994, he has also held a position at Micron Technology, Inc., as a senior design engineer.

Dr. Baker holds over 20 granted or pending patents in integrated circuit design and is a member of the electrical engineering honor society Eta Kappa Nu. He is a co-author of *CMOS: Circuit Design, Layout, and Simulation*, Wiley-IEEE, 1998, and *DRAM Circuit Design: A Tutorial*, Wiley-IEEE, 2001. His research interests are in the areas of CMOS mixed-signal integrated circuit design and the design of memory in new and emerging fabrication technologies. Dr. Baker was a corecipient of the 2000 Prize Paper Award of the IEEE Power Electronics Society.