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](image)

**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

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

\[
x(t) = V_p \sin(2\pi f_{in} \cdot t)
\]

\[
y(t) = x(t) \cdot \delta_u(t - nT_s)
\]

![Figure 30.3](image)

**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

$$Y(f) = \frac{V_p}{T_s} \sum_{k=\infty} ^ {\infty} \delta(f - f_{in} + kf_s) + \delta(f + f_{in} + kf_s)$$  \hspace{1cm} (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 $f_s$ [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).

**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 in 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. ■
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, \( f_v \), \( 2f_s \), \( 3f_v \), 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.
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.

$$|H(j\omega)| = 1 \text{ for AAF}$$

$$\angle H(j\omega) = \frac{f_s}{f_n} \text{ for RCF}$$

$$f_n = \frac{f_s}{2}$$

slope = $-2\pi t_o$

**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. 

$$v_{in}(t) \xrightarrow{\text{AAF}} v_{out}(t) = v'_{in}(t - t_o)$$

**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.

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

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

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., \( \theta = 0 \). 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° (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

\[ \theta(f) = \frac{t_o \cdot f}{T} \cdot 360 = t_o \cdot f \cdot 360 \]
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?"

\[ V_{out} = \frac{V_{in} + V_{in} \cdot e^{j2\pi f(-\tau)}}{2} \]

**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 (= \( \tau \)) 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
The transfer function can be written as

\[
\frac{V_{out}}{V_{in}} = \frac{1}{2}(1 + e^{j2\pi f(-t_o)}) = \frac{1}{2} \left( \frac{\text{Real}}{1 + \cos 2\pi f \cdot (-t_o)} + j \cdot \sin 2\pi f \cdot (-t_o) \right)
\]

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 ±90. Using \( \cos^2x = \frac{1 + \cos 2x}{2} \) and \( \sin 2x = 2\sin x\cos x \)

the phase response is given by

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

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
.Rt1 Vout Vout 0 50
.Rt2 Vin Vout 500
```

*Figure 30.12 CMOS: Circuit Design for Mixed-Signal Systems*
Chapter 30 Data Converter Modeling

\begin{verbatim}
T1   Vin  0   Vout  0   ZO=50   TD=5n
.end
\end{verbatim}

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 \( |v_{\text{out}}/v_{\text{in}}| \) 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](image)

**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 \( \delta(t) = 0 \)).
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

\[ T_s = \frac{1}{f_s} \]

Figure 30.14  Impulse sampling, at 100 MHz, a sinewave at 20 MHz.
the inverse Fourier transform of the transfer function. The ideal RCF's transfer function (Fig. 30.7) can be defined by

$$|H(f)| = \frac{1}{f_s} \text{ for } |f| < f_n \text{ else } |H(f)| = 0$$  \hspace{1cm} (30.3)

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

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

$$\frac{\sin 2\pi f_n \cdot t}{\pi f_s \cdot t} = \frac{\sin \pi f_s \cdot t}{\pi f_s \cdot t} = Sinc(\pi f_s \cdot t)$$  \hspace{1cm} (30.4b)

where

$$\frac{\sin x}{x} = Sinc(x)$$  \hspace{1cm} (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

![Diagram of impulse response](image-url)
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.

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 $f_s \to \infty$, 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 \texttt{linearize} command with no arguments will linearize all of the variables available in the simulation. Next, we use the \texttt{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 \texttt{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=\text{db}(Vout+1e-6)
let vindb=\text{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 µV 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

\[
\text{Baseline reduction (or duty cycle)} = \frac{2 \cdot \text{(step size)}}{T_s}\tag{30.7}
\]

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-stepsize <UIC>
```

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

\[
\text{step size} = 1\% \cdot T_s \text{ or with ideal components } 10\% \cdot T_s\tag{30.8}
\]

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
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

\[
y(t) = x(t) \cdot \sum_{k=-\infty}^{\infty} \delta(t - kT_s)
\]

(30.9)
We can rewrite Eq. (30.9) as
\[ y(t) = x(t)\sum_{n=-\infty}^{\infty} + \delta_a(t + T_s) + \delta_a(t - T_s) + \delta_a(t - 2T_s) + \delta_a(t - 3T_s) + \ldots \]  
(30.10)

Taking the Fourier transform of this equation results in
\[ Y(f) = \sum_{n=-\infty}^{\infty} + x(-1)e^{(1)2\pi f T_s} + x(0)e^{(0)2\pi f T_s} + x(1)e^{(1)2\pi f T_s} + x(2)e^{(2)2\pi f T_s} + \ldots \]  
(30.11)

where the term \( e^{2\pi f T_s} \) 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
\[ z = e^{2\pi f T_s} = e^{2\pi f T_s} \]  
(30.12)

then the output of our sampler can be written as
\[ Y(z) = \sum_{n=-\infty}^{\infty} + x(-1)z + x(0)z^0 + x(1)z^{-1} + x(2)z^{-2} + \ldots \]  
(30.13)

or
\[ Y(z) = \sum_{k=-\infty}^{\infty} x(k) \cdot z^{-k} \]  
(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
\[ u(t) = 1 \text{ for } t \geq 0 \]
\[ u(t) = 0 \text{ for } t < 0 \]

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, \( H(z) = 1 \).
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 $L \cdot T_s$.

An Important Note

It's important to note that our impulse sampler quantizes\(^1\) 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.

```plaintext
* 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
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 GΩ) 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 μs (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 \( f \), 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

$$y(t) = \sum_{n=0}^{\infty} \left( V_p \sin(2\pi f_{in} \cdot nT_s) \cdot h(t) \cdot \left[ u(t-nT) - u(t-(n+1)T) \right] \right)$$  \hspace{1cm} (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

$$\text{Fourier}[u(t-nT) - u(t-(n+1)T)] = \int_{0}^{T_s} [u(t-nT) - u(t-(n+1)T)] e^{-j2\pi ft} \cdot dt$$  \hspace{1cm} (30.16)

which is evaluated as

$$H(f) = \left. \frac{e^{-j2\pi ft}}{-j \cdot 2\pi \cdot f} \right|_0^{T_s} = e^{-j\pi fT} \cdot \frac{\sin(\pi ft)}{\pi ft} \cdot \left. \frac{e^{-j\pi fT}}{j \cdot 2\pi \cdot f} \right|_0^{T_s} = e^{-j\pi fT} \cdot T \cdot \text{Sinc}(\pi \cdot f \cdot T)$$  \hspace{1cm} (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 $H(f)$.

$$h(t) = u(t-nT) - u(t-(n+1)T)$$

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

$$Y(f) = H(f) \ast V_{in}(f) = \int_{-\infty}^{\infty} H(L) \cdot V_{in}(f-L) \, dL$$

or

$$|Y(f)| = \frac{V_p}{T_s} \cdot \left| \sum_{k=-\infty}^{\infty} \delta(f - f_{in} + kf_s) + \delta(f + f_{in} + kf_s) \right|$$

(30.18)

As $T \rightarrow 0$ ($h(t) \rightarrow \delta_o(t)$), 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 $\text{Sinc} \left( \frac{\pi f}{f_s} \right)$ or $\text{Sinc} \left( \frac{\pi f}{2f_n} \right)$. 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 \( f_s \)), 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 \( \text{Sinc} \) response filter. Second,
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 \( \text{Sinc}(\pi f_s) \) 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

\[
H_{RCFSH}(f) = \frac{1}{\text{Sinc}(\pi f_s)} = \frac{\pi f}{2 f_s \sin\left(\frac{\pi f}{2 f_s}\right)}
\]

(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.

![Ideal reconstruction filter response for a S/H](image)

**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 \( \mu \)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 \( \mu \)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

\[
\text{ABSTOL or RELTOL} \cdot I_{\text{simulated}}
\]

while for a voltage we use the larger of

\[
\text{VNTOL or RELTOL} \cdot V_{\text{simulated}}
\]

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 μV 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 \(\phi_1\) and \(\phi_2\) switches are closed while the \(\phi_3\) 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 \(\phi_1\) switch opens and for a very short time (set by \(t_2 - t_1\)) 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 \(\phi_3\) switches turning off are independent of the input signal. When the \(\phi_2\) 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 \(\phi_2\) 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

\[\text{Figure 30.31 Single-ended S/H operation.}\]
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 $\phi_2$ 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 $\phi_2$ 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 $\phi_1$ switch while, in Fig. 30.32, the bottom plate of the hold capacitor is connected to the $\phi_1$ switch.

![Bottom plate sampling](image)

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 $\phi_3$ 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

\[
1 \text{ LSB} = \frac{V_{REF^+} - V_{REF^-}}{2^N} = V_{LSB}
\]  

(30.23)

If, for example, \( V_{REF^+} = 1.25 \text{ V} \) and \( V_{REF^-} = 0.25 \text{ V} \) 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](image-url)  

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 $V_{REF+} = VDD = 1.5 \text{ V}$ and $V_{REF-} = VSS = 0 \text{ V}$. 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

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

or

$$V_{OUT} = (V_{REF+} - V_{REF-}) \cdot \frac{1}{2^N} \cdot \left( b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \ldots + b_1 \cdot 2^1 + b_0 \right) + V_{REF-} \tag{30.25}$$

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)*
\(+v(B1L)*2048+v(B1L)*1024+v(B9L)*512+v(B8L)*256\)
\(+v(B7L)*128+v(B6L)*64+v(B5L)*32+v(B4L)*16+v(B3L)*8+\)
\(+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
VREFPVREFP 0 DC 1.5
VREFM VREFM 0 DC 0.0
VB7 B7 0 DC 0 pulse 1.5 0 200p 200p 1279.8n 2560n
VB6 B6 0 DC 0 pulse 1.5 0 200p 200p 639.8n 1280n
VB5 B5 0 DC 0 pulse 1.5 0 200p 200p 319.8n 640n
VB4 B4 0 DC 0 pulse 1.5 0 200p 200p 159.8n 320n
VB3 B3 0 DC 0 pulse 1.5 0 200p 200p 79.8n 160n
VB2 B2 0 DC 0 pulse 1.5 0 200p 200p 39.8n 80n
VB1 B1 0 DC 0 pulse 1.5 0 200p 200p 19.8n 40n
VB0 B0 0 DC 0 pulse 1.5 0 200p 200p 9.8n 20n
X1 VDD VREFP VREFM Vout B7 B6 B5 B4 B3 B2 B1 B0 DAC8bit
```
In this netlist we are assuming $V_{REF+} = 1.5\text{ V}$ 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

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

(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

\[
V_{OUT} = (V_{REF+} - V_{REF-}) \cdot \frac{1}{2^N} \cdot (b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \ldots + b_1 \cdot 2^1 + b_0 + 1) + V_{REF-}
\]

(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

\[
V_{CM} = \frac{V_{REF+} + V_{REF-}}{2} \rightarrow V_{CM0} = \frac{V_{REF+} - V_{REF-}}{2}
\]

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

* Level shift by \( V_{\text{REFM}} \) and 1/2LSB

\[
\text{BPIP PIPIN 0 V=V(\text{OUTSH})-V(V_{\text{REFM}})+((V(V_{\text{REFP}})-V(V_{\text{REFM}}))/2^9)}
\]

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

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

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 \( V_{\text{REF}^+} = 0.5 \text{ V} \) and \( V_{\text{REF}^-} = 0.25 \text{ V} \). Note that if \( V_{\text{REF}^-} = 0 \) and \( V_{\text{REF}^+} = V_{\text{DD}} \), 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
.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
.ends

*** END ADC Subcircuit ****************************

Figure 30.38 SPICE subcircuit netlist for an ideal 8-bit ADC.
Chapter 30 Data Converter Modeling 35

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=(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 μV.) 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](image1)

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

![Figure 30.41](image2)

**Figure 30.41** Seven MHz ADC input and the corresponding DAC output.
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 ADC's 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 fed-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.

![Diagram](image)

**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, $V_{OUTSH}$.

![Graphs](image)

**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](image)

*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 mV/1 LSB (1 LSB = 1.5/256 = 5.86 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, when the ADC input is a slow ramp.
value. This value can be calculated using
\[
V_{Qe,RMS} = \sqrt{\frac{1}{T} \int_0^T (0.5 \text{ LSB} - \frac{1 \text{ LSB}}{T} \cdot t)^2 dt} = \frac{1 \text{ LSB}}{\sqrt{12}} = \frac{V_{LSB}}{\sqrt{12}} \tag{30.30}
\]

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, \(V_{Qe,RMS} = 1.69 \text{ mV} \text{ or } -55.43 \text{ dB}\).

**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 ±0.5 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.](image)

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, \(Qe\), is zero, by
\[
P_{Qe} = \int_{-1/2LSB}^{1/2LSB} \rho \cdot (Qe)^2 \cdot dQe = \frac{V_{LSB}^2}{12} \tag{30.31}
\]
so that, once again, the RMS quantization noise voltage is
\[
V_{Qe,RMS} = \frac{V_{LSB}}{\sqrt{12}} \tag{30.32}
\]

Again, if our LSB voltage is 5.86 mV, then, once again, \(V_{Qe,RMS} = 1.69 \text{ mV} \text{ or } -55.43 \text{ dB}\). 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.](image-url)
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 $V_{DM}(f)$ in Fig. 30.42 as an example. If the resolution of the DFT is $f_{RES}$, then we can write this as

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

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,

```plaintext
display
```
and the following appears:

```plaintext
frequency : frequency, real, 401 long [default scale]
v_in : 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:

```plaintext
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

\[
\frac{1}{\text{simulation time}} = \frac{1}{T_{\text{stop}}} > \frac{2}{\text{DFT resolution}} = \frac{2}{f_{\text{res}}}
\]  

(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, \(T_{\text{stop}}\). 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.

(a) 
(b) 
(c) 
(d) Frequency spectrum of (a) 
(e) Frequency spectrum of (c) 
(f), Von Hann (Hanning) window

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 ($= 1$ MHz if $T_{stop} = 2,000$ 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.](image)

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.

\[ T_{\text{stop}} = \frac{2}{f_{\text{res}}} \]  

(30.35)

**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.
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:

```plaintext
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](image) 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., RELTOL = 10u) 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

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

(30.36)

or, if we assume a finite simulation time,

$$
V(f) = \int_{0}^{\text{stop}} v(t) \cdot e^{-j2\pi ft} \cdot dt
$$

(30.37)

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

$$
dt = \Delta t \text{ and } t = k \cdot \Delta t \text{ where } k = 0, 1, \ldots N
$$

(30.38)

The variable $\Delta t$ 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 ($T_{\text{stop}} = N \cdot \Delta t$). The frequencies where the DFT is calculated, assuming Eq. (30.35) is valid, are given by

$$
f = n \cdot f_{\text{res}} = \frac{2n}{T_{\text{stop}}} \text{ where } n = 0, 1, \ldots M - 1
$$

(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

$$
V(f) \big|_{f_{\text{res}}} = \sum_{k=0}^{N} v(k \cdot \Delta t) \cdot e^{-j(4\pi/N)nk} \cdot \Delta t
$$

(30.40)

(noting that $4\pi$ is used in the exponent because our DFT resolution was twice the reciprocal of the simulation time), or
52 CMOS Mixed-Signal Circuit Design

\[ V(f) \big|_{f=nf_{\text{fs}}} = \Delta t \cdot V_{\text{DFT}}(n) \]  

(30.41)

In general, a DFT output is scaled (divided by \( \Delta t \)) 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 \( V_{\text{Qe, RMS}} \), \( V_{\text{Qe}}^2(f) \) with units of \( V^2/Hz \), or the noise voltage spectral density, \( V_{\text{Qe}}(f) \) with units of \( V/\sqrt{Hz} \) by solving

\[
\frac{V_{\text{LSB}}^2}{12} = 2 \int_0^{f_s/2} V_{\text{Qe}}^2(f) \cdot df
\]

(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

\[
V_{\text{Qe}}(f) = \frac{V_{\text{LSB}}}{\sqrt{12f_s}} = \frac{V_{\text{REF+}} - V_{\text{REF-}}}{2^N \sqrt{12f_s}}
\]

(30.43)

with units of \( V/\sqrt{Hz} \). 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 \( V_{\text{Qe}}(f) \) added to the input signal.

![Figure 30.56 Modeling ADC quantization noise.](image)

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
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

\[ V_{Qe}^2(f) = 2 \int_{f_L}^{f_H} V_{Qe}^2(f) \cdot df \quad \text{where} \quad f_L < f_H \leq f_s/2 \]  

(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. ■
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

$$V_{OUT} = \frac{V_{OUTA} + V_{OUTB}}{2}$$  \hspace{1cm} (30.45)

**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.

**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](image)

**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 \(V_{\text{LSB}}/\sqrt{12(f_s)}\) to \(V_{\text{LSB}}/\sqrt{12 \cdot (2f_s)}\) 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

\[
V_{Qe,RMS}^2 = 2 \int_0^{f_s/2} \frac{1}{2} \frac{V_{\text{LSB}}^2}{12f_s} \, df
\]

or

\[
V_{Qe,RMS} = \frac{1}{\sqrt{2}} \frac{V_{\text{LSB}}}{\sqrt{12}}
\]
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

\[
V_{Qe,RMS} = \frac{1}{\sqrt{K}} \cdot \frac{V_{LSB}}{\sqrt{12f_s}}
\]

(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
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.

[Diagram of Analog and Digital Signals]

**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


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
$\Delta t$ - Time difference between points used in a DFT
$\phi$ - Clock signal
$f_{ck}$ - 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
Chapter 30 Data Converter Modeling

\( f_{res} \) - 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

LSB = \( V_{LSB} = \frac{V_{REF+} - V_{REF-}}{2^N} \) = 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

\( \rho \) - Probability density function

\( \text{Sinc}(x) = \sin(x)/x \)

\( \theta \) - Phase of a function

\( T \) - period of a periodic waveform

\( t_o \) - A time delay

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

\( T_{\text{stop}} \) - Final simulation time

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

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

\( V_{DD} \) - positive power supply voltage which is 1.5 V in this chapter

\( V_{DFT(n)} \) - Discrete Fourier Transform of V
$V_{OUTD} - \text{Difference between an analog input and a digitized output, see Fig. 30.45.}$

$V_{OUTDB} - \text{Output signal in decibels}$

$V_{INDB} - \text{Input signal in decibels}$

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

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

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

$V_{inbuf} - \text{Input signal after buffering}$

$V_{ins} - \text{Input signal after sampling}$

$V_{LSB} - \text{See LSB}$

$V_{out} - \text{Output voltage}$

$V_p - \text{Peak sinewave amplitude}$

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

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

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

$V_{SS} - \text{negative power supply voltage which is 0 V in this chapter}$

$z = e^{2\pi f T_s} = e^{j\frac{2\pi f}{f_s}}$

**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 $e^{j2\pi f \tau_d}$ 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.

![Sample and hold (S/H)](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?