CHAPTER 7

ADC Testing

As mentioned in Chapter 6, "DAC Testing," there are many similarities between DAC testing and ADC testing. However, there are also a few notable differences. In this chapter, we will examine their differences as they relate to the intrinsic parameters of an ADC such as DC offset, INL, and DNL. A discussion will then follow about testing the dynamic operation of ADCs.

7.1 ADC TESTING VERSUS DAC TESTING

7.1.1 Comparison of DACs and ADCs

The primary difference between DAC and ADC testing relates to the fundamental difference in their transfer curves. As discussed in Chapter 6, the DAC transfer curve is a one-to-one mapping function, while the ADC transfer curve is a many-to-one mapping functionas captured by the 4-bit DAC and ADC example shown in Figure 7.1. In this chapter, we will see that the ADC curve in Figure 7.1b is one that never occurs in practice. The output codes generated by a real-world ADC are affected by noise from the input circuits. As a result, an ADC curve is statistical in nature rather than deterministic. In other words, for a given input voltage, it may not be possible to predict exactly what output code will be produced. Before we can study testing methods for ADCs, we should first examine the statistical nature of a true ADC transfer curve.

7.1.2 Statistical Behavior of ADCs

To understand the statistical nature of ADCs, we have to model the ADC as a combination of a perfect ADC and a noise source with no DC offset. The noise source represents the combination of the noise portion of the real-world input signal plus the self-generated noise of the ADCs input circuits. Figure 7.2 shows this noisy ADC model.

Applying a DC level to the noisy ADC, we can begin to understand the statistical nature of ADC decision levels. A noise-free ADC might be described by a simple output/input relationship such as

$$\text{output code} = \textit{Quantize}\ (\text{input voltage}) \qquad \textbf{(7.1)}$$

Figure 7.1. Comparing transfer curves. (a) DAC and (b) ADC.

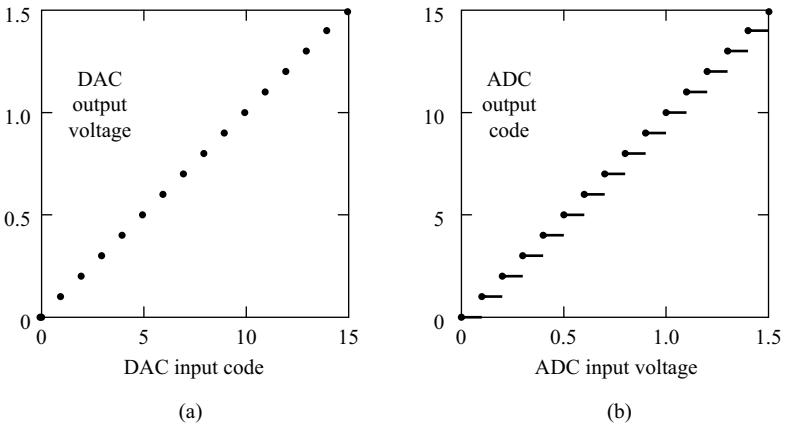
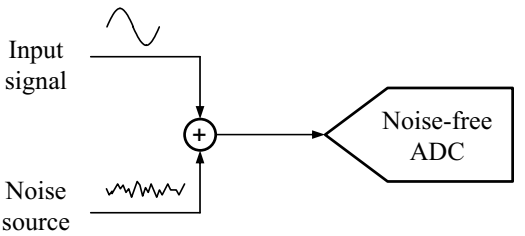


Figure 7.2. ADC model including input noise.



where the function *Quantize* ( ) represents the noise-free ADC's quantization process. The noisy ADC can be described using a similar equation

$$\text{output code} = \textit{Quantize}\ (\text{input voltage} + \text{noise voltage}) \tag{7.2}$$

Now consider the case of a noisy ADC with a DC input voltage. If the DC input level lies exactly between two ADC decision levels, and the noise voltage rarely exceed ±½ V_{LSB}, then the ADC will for the most part produce the same output code. The noise voltage for all practical purposes never gets large enough to push the total voltage across either of the adjacent decision levels. We depict this situation in Figure 7.3 where we described the input signal v_{IN} with a probability density function given by

$$f\left(v_{IN}\right) = \frac{1}{\sigma_n \sqrt{2\pi}} e^{\frac{-(v_{IN} - V_{DC})^2}{2\sigma_n^2}} \tag{7.3}$$

Here we assume that the noise present at the ADC input is modeled as a Gaussian-distributed random variable with zero mean and a standard deviation of σ_n (i.e., the RMS noise voltage).

On the other hand, if the input DC voltage is exactly equal to a decision level (i.e., $v_{IN} = V_{TH}$) as depicted in Figure 7.4, then even a tiny amount of noise voltage will cause the quantization process to randomly dither between the two codes on each side of the decision level. While we

Figure 7.3. Probability density plot for DC input between two decision levels.

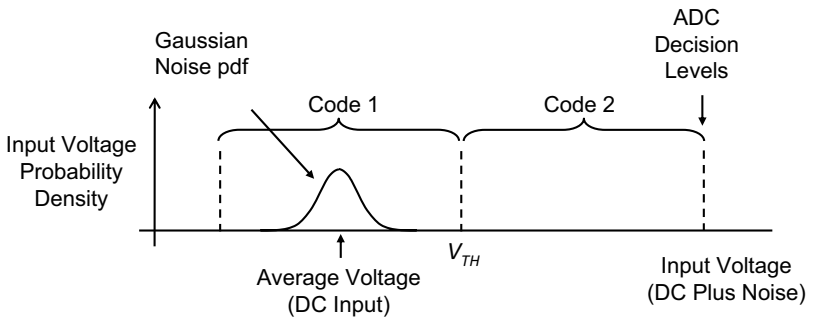
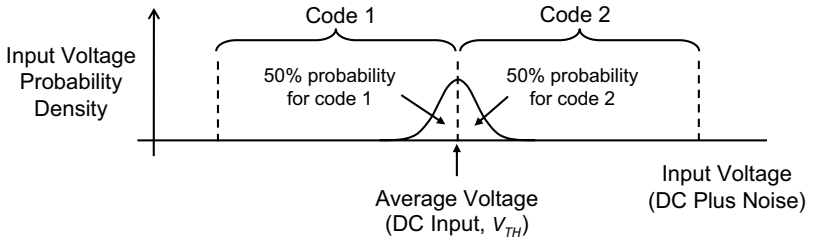


Figure 7.4. Probability density plot for DC input equal to a decision level.



are assuming that the noise is Gaussian distributed, this conclusion will be same regardless of the nature of the noise as long as its pdf is symmetrical about its mean value.[1] Since the area under the pdf is equally split between code 1 and code 2, we would expect 50% of the ADC conversions to produce code 1 and 50% of the conversions to produce code 2.

For input voltages that are close but not equal to the decision levels, the process is little more complicated but tractable using the probability theory from Chapter 4. Consider the DC input V_{DC} as being some value less than the decision level V_{TH} that separates code 1 and code 2, such as situation depicted in Figure 7.5. The probability that the input v_{IN} will trip the ADC quantizer to the next code value is given by

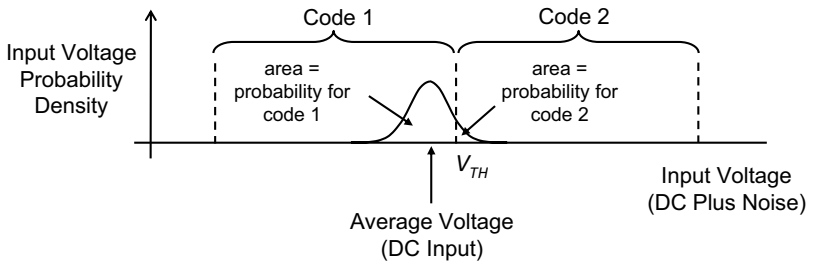
$$P\left(V_{TH} < v_{IN}\right) = \int_{V_{TH}}^{\infty} \frac{1}{\sigma_n\sqrt{2\pi}} e^{\frac{-(v_{IN}-V_{DC})^2}{2\sigma_n^2}} dv_{IN} = 1 - \Phi\left(\frac{V_{TH}-V_{DC}}{\sigma_n}\right) \tag{7.4}$$

where $\Phi(z)$ is the standard Gaussian cumulative distribution function. Likewise, the probability that the input signal will not trip the ADC decision level is

$$P\left(v_{IN} < V_{TH}\right) = \Phi\left(\frac{V_{TH}-V_{DC}}{\sigma_n}\right) \tag{7.5}$$

We can therefore conclude that N samples of the ADC output will contain $N \times \Phi\left(\Delta V/\sigma_n\right)$ code 1 codes and $N \times [1 - \Phi\,(\Delta V/\sigma_n)]$ code 2 codes where $\Delta V = V_{TH} - V_{DC}$. Of course, this assumes that the noise level is sufficiently small that other codes are not tripped.

Figure 7.5. Probability density plot for DC input input less than the decision level V_{TH}.



EXAMPLE 7.1

An ADC input is set to 2.453 V DC. The noise of the ADC and DC signal source is characterized to be 10 mV RMS and is assumed to be perfectly Gaussian. The transition between code 134 and 135 occurs at 2.461 V DC for this particular ADC. Therefore, the value 134 is the expected output from the ADC. What is the probability that the ADC will produce code 135 instead of 134? If we collected 200 samples from the output of the ADC, how many would we expect to be 134 and how many would be 135? How might we determine that the transition between code 134 and 135 occurs at 2.461 V DC? How might we characterize the effective RMS input noise?

Solution:

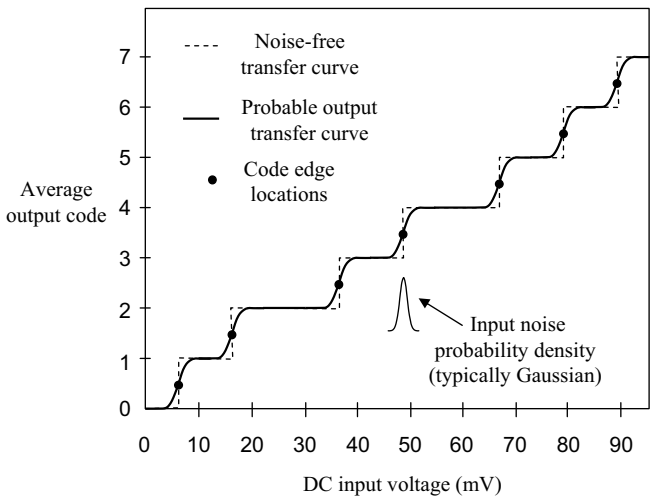
With an input of 2.453 V DC, the ADC's input noise would have to exceed 2.461 V – 2.453 V = 8 mV to cause the ADC to trip to code 135. This value is equal to +0.8σ, since σ = 10 mV. From Appendix A, the Gaussian cdf of +0.8σ is equal to 0.7881. Therefore, there is a 78.81% probability that the noise will *not* be sufficient to trip the ADC to code 135. Thus we can expect 78.81% of the conversions to produce code 134 and 21.19% of the conversions to produce code 135. If we collect 200 samples from the ADC, we would expect 78.81% of the 200 conversions (approximately 158 conversions) to produce code 134. We would expect the remaining 21.19% of the conversions (42 samples) to produce code 135.

To determine the transition voltage, we simply have to adjust the input voltage up or down until 50% of the samples are equal to 134% and 50% are equal to 135. To determine the value of σ, we can adjust the input voltage until we get 84.13% of the conversions to produce code 134. The difference between this voltage and the transition voltage is equal to 1.0σ, which is equal to the effective RMS input noise of the ADC.

Because the circuits of an ADC generate random noise, the ADC decision levels represent *probable* locations of transitions from one code to the next. In the previous example, we saw that an input noise level of 10 mV would cause a 2.453-V DC input voltage to produce code 134 only 79% of the time and code 135 21% of the time. Therefore, with an input voltage of 2.453V, we will get an *average* output code of 134 × 079 + 135 × 0.21 = 134.21. Of course, the ADC cannot produce code 134.21. This value only represents the average output code we can expect if we collect many samples.

If we plot the average output code from a typical ADC versus DC input levels, we will see the true transfer characteristics of the ADC. Figure 7.6 shows a true ADC transfer curve compared

Figure 7.6. ADC probable output code transfer curve.



to the idealized, noise-free transfer curve. The center of the transition from one code to the next (i.e., the decision level) is often called a *code edge*. The wider the distribution of the Gaussian input noise, the more rounded the transitions from one code to the next will be. In fact, the true ADC transfer characteristic is equal to the convolution of the Gaussian noise probability density function with the noise-free transfer curve.

Code edge measurement is one of the primary differences between ADC and DAC testing. DAC voltages can simply be measured one at a time using a DC voltmeter or digitizer. By contrast, ADC code edges can only be measured using an iterative process in which the input voltage is adjusted until the output samples dither equally between two codes. Because of the statistical nature of the ADC's transfer curve, each iteration of the search requires 100 or more conversions to achieve a repeatable average value. Since this brute-force approach would lead to very long test times in production, a number of faster methodologies have been developed to locate code edges. Unfortunately, these production techniques generally result in somewhat less exact measurements of code edge voltages.

Exercises

7.1. If V is normally distributed with zero mean and a standard deviation of 2 mV, find $P(V< 4\text{ mV})$. Repeat for $P(V> -1\text{ mV})$. Repeat for $P(-1\text{ mV} < V < 4\text{ mV})$.

ANS. $P(V< 4\text{ mV}) = 0.9772$; $P(V> -1\text{ mV}) = 0.6915$; $P(-1\text{ mV} < V < 4\text{ mV}) = 0.6687$

7.2. If V is normally distributed with zero mean and a standard deviation of 100 mV, what is the value of V_{IN} such that $P(V < V_{IN}) = 0.9641$.

ANS. $V_{IN} = 180$ mV.

7.3. An ADC input is set to 1.4 V DC. The noise of the ADC and DC signal source is characterized to be 15 mV RMS and is assumed to be perfectly Gaussian. The transition between code 90 and 91 occurs at 1.4255 V DC. If 500 samples of the ADC output are collected, how many do we expect to be code 90 and how many would be code 91?

ANS. # of code 90 = 95.54% or ~ 478 and # of code 91 = 4.46% (~22).

In the next section, we will examine the various ways in which the code edges of an ADC can be measured, both for characterization and production. Once the code edges have been located, we can apply all the same tests to ADCs that we applied to DACs. Tests such as INL, DNL, DC gain, and DC offset are commonly performed using the code edge information.

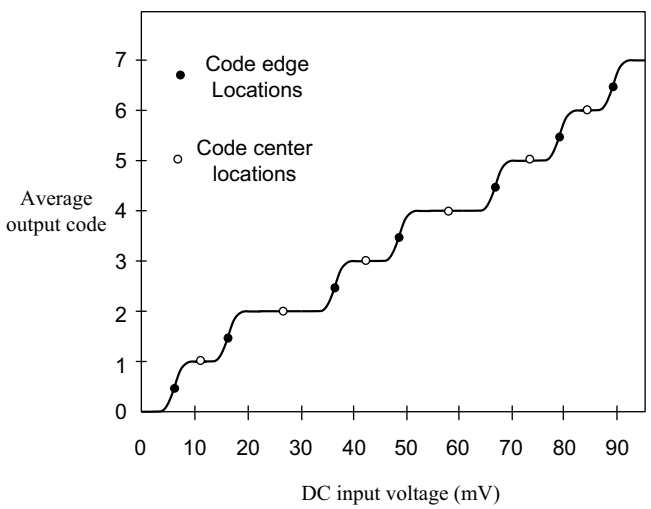
7.2 ADC CODE EDGE MEASUREMENTS

7.2.1 Edge Code Testing Versus Center Code Testing

To measure ADC intrinsic parameters such as INL and DNL, we first have to convert the many-to-one transfer curve of the ADC into a one-to-one mapping function similar to that of a DAC. Then we simply apply the same testing methods and criteria from Chapter 6 to the one-to-one transfer curve of the ADC. There are two ways to convert the many-to-one transfer curve of an ADC into a one-to-one curve. These two methods are known as *center code testing* and *edge code testing*. Figure 7.7 illustrates the difference between edge code testing and center code testing. Code centers are defined as the midpoint between the code edges. For example, consider a case in which the decision level between code 57 and code 58 corresponds to an input voltage of 100 mV and the decision level between codes 58 and 59 corresponds an input of 114 mV. In this example, the center of code 58 corresponds to the average of these two voltages, (114 mV + 100 mV)/2 = 107 mV.

Figure 7.7 highlights the problem with center code testing. Notice that the code centers fall very nearly on a straight line, while the code edges show much less linear behavior. The averaging process in the definition of code centers produces an artificially low DNL result compared to edge code testing. Because the code widths in Figure 7.7 alternate between wide and narrow codes, the averaging process effectively smooth's these variations out, leaving a transfer characteristic that looks like it has fairly evenly spaced steps. Because center code testing produces an artificially low DNL value, this technique should be avoided. The edge code method is a more discerning test, and is therefore the preferred means of translating the transfer curve of an ADC to the one-to-one mapping needed for INL and DNL measurements.

Figure 7.7. Code edges and code centers.



We can search for code edges in one of several different ways. Three common techniques are the step search or binary search method, the hardware servo method, and the histogram method. In the next section, we will see how each of these techniques is applied, and we will examine the strengths and weaknesses of each method. Since all the various ADC edge measurement techniques are slower than simply measuring an output voltage, ADC testing is generally much slower than DAC testing.

7.2.2 Step Search and Binary Search Methods

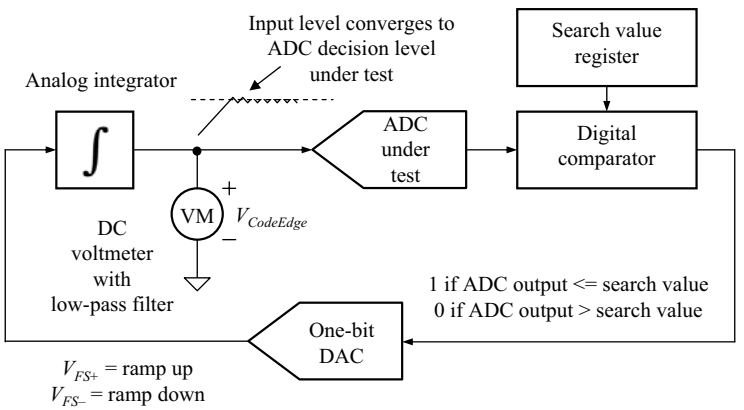
The most obvious method to find the edge between two ADC codes is to simply adjust the input voltage of the ADC up or down until the output codes are evenly divided between the first code and the second. To achieve repeatable results, we need to collect about 50 to 100 samples from the ADC so that we have a statistically significant number of conversions. The input voltage adjustment could be performed using a simple step search, but a faster method is to use a binary search to quickly find the input voltage corresponding to the ADC code edge. (Step searches and binary searches were discussed in Section 3.11.)

Binary searches are an acceptable production test method for comparators and slicer circuits, which are effectively one-bit ADCs. However, if we try to apply a binary search technique to multibit ADCs in production, we run into a major problem. If we use a binary search with, say, five iterations, we have to collect 100 samples for each iteration. This would result in a total of 500 collected samples *per code edge*. A *D*-bit ADC has $2^D - 1$ code edges. Therefore, the test time for most ADCs would be far too high. For example, a 10-bit ADC operating at a sampling rate of 100 kHz would require a total data collection time of 500 codes times $2^{10} - 1$ edges times the sample period (1/100 kHz). Thus the total collection time would be $500 \times 1023 \times 10\ \mu s = 5.115\ s$! Clearly, this is not a production-worthy solution.

7.2.3 Servo Method

A much better method for measuring code edges in production is the use of a servo circuit. Figure 7.8 shows a simplified block diagram of an ADC servo measurement setup. The output codes from the ADC are compared against a value programmed into the search value register. If the ADC output is greater than or equal to the expected value, the integrator ramps downward. If it is less than the expected value, the integrator ramps upward.

Figure 7.8 ADC servo test setup.



Eventually, the integrator finds the desired code edge and fluctuates back and forth across its transition level. The average voltage at the ADC input, $V_{CodeEdge}$, represents the lower edge of the code under test. This voltage can easily be measured using a DC voltmeter with a low-pass filtered input. The servo search process is repeated for each code edge in the ADC transfer curve.

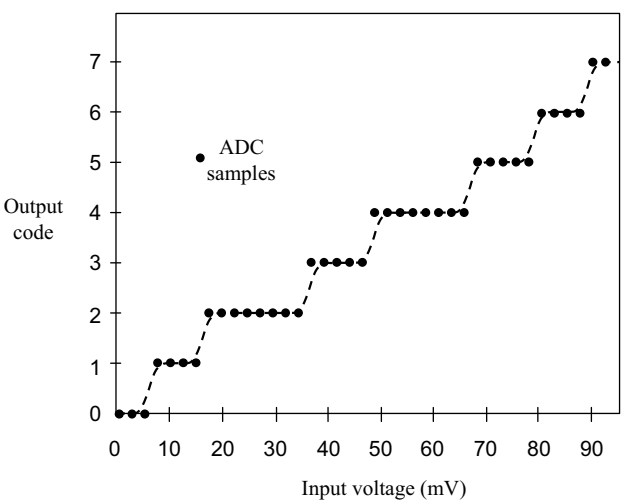
The servo method is actually a fast hardware version of the step search. Unlike the step search or binary search methods, the servo method does not perform averaging before moving from one input voltage to the next. The continuous up/down adjustment of the servo integrator coupled with the averaging process of the filtered voltmeter act together to remove the effects of the ADC's input noise. Because of its speed, the servo technique is generally more production-worthy than the step search or binary search methods.

Although the servo method is faster than the binary search method, it is also fairly slow compared with a more common production testing technique, the histogram method. Histogram testing requires an input signal with a known voltage distribution. There are two commonly used histogram methods: the linear ramp method and the sinusoidal method.

7.2.4 Linear Ramp Histogram Method

The simplest way to perform a histogram test is to apply a rising or falling linear ramp to the input of the ADC and collect samples from the ADC at a constant sampling rate. The ADC samples are captured as the input ramp slowly moves from one end of the ADC conversion range to the other. The ramp is set to rise or fall slowly enough that each ADC code is "hit" several times, as shown in Figure 7.9. The number of occurrences of each code is directly proportional to the width of the code. In other words, wide codes are hit more often than narrow codes. For example, if the voltage spacing between the upper and lower decision levels for code 2 are twice as wide as the spacing for code 1, then we expect code 2 to occur twice as often as code 1. The reason for this is that it takes the linear ramp input signal twice as long to sweep through code 2 as it takes to sweep through code 1. Of course, this method assumes that the ramp is perfectly linear and that the ADC sampling rate is constant throughout the entire ramp. This condition is easily maintained in mixed-signal ATE testers.

Figure 7.9. ADC samples from linear ramp histogram test.



The number of occurrences of each code is plotted as a histogram, as illustrated in Figure 7.10. Ideally, each code should be hit the same number of times, but this would only be true for a perfectly linear ADC. The histogram shows us which codes are hit more often, indicating that they are wider codes. For example, we can see from the histogram in Figure 7.10 that codes 2 and 4 are twice as wide as codes 1 and 6.

Let us denote the number of hits that occur for the ith code word of a D-bit ADC as $H(i)$ for $i = 0,1,\ldots,2^D - 1$. Next,let us define the average number of hits for each code word, excluding the number of hits included in the two end codes, as

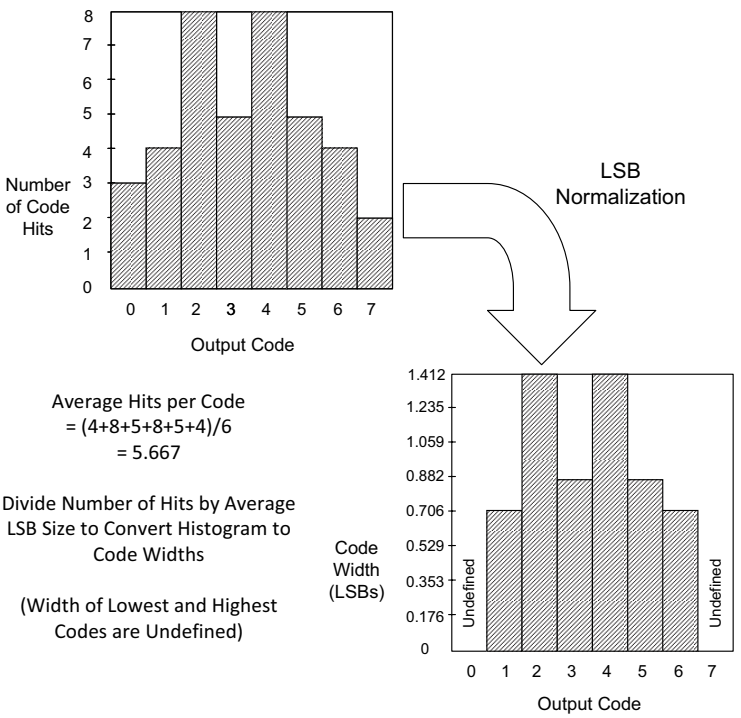
$$H_{Average} = \frac{1}{2^D - 2} \sum_{i=1}^{2^D-2} H(i) \tag{7.6}$$

Dividing $H(i)$ by $H_{Average}$, we obtain the width of each code word in units of LSBs as

$$\text{code width}(i) = \frac{H(i)}{H_{Average}}, \quad i = 1,2,\ldots,2^D - 2 \tag{7.7}$$

Excluding the highest and lowest code count is necessary, because these two codes do not have a defined code width. In effect, the end codes are infinitely wide. For example, code 0 in an unsigned binary ADC has no lower decision level, since there is no code corresponding to −1. In many practical situations, the input ramp signal extends beyond the upper and lower ranges of the

Figure 7.10. LSB normalization translates ADC code histogram into LSB code widths.



ADC resulting in an increase code count for these two code words. These meaningless hits should be ignored in the linear ramp histogram analysis.

7.2.5 Conversion from Histograms to Code Edge Transfer Curves

To calculate absolute or best-fit INL and DNL curves, we have to determine the absolute voltage for each decision level. Unfortunately, an LSB code width plot such as the one in Figure 7.10 tells us the width of each code in LSBs rather than volts. To convert the code width plot into voltage units, we need to measure the average LSB size of the ADC, in volts. This can be done using a binary search or servo method to find the upper and lower code edge voltages, V_{UE} and V_{LE}. In a D-bit ADC, there are $2^D - 2$ LSBs between these two code edges. Therefore, the average LSB size can be calculated as follows:

$$V_{AveCodeWidth} = V_{LSB} = \frac{V_{UE} - V_{LE}}{2^D - 2} \tag{7.8}$$

The code width plot can then be converted to volts by multiplying each value by the average code width, in volts

$$V_{CodeWidth}(i) = V_{LSB} \times \text{LSB code width}(i) \tag{7.9}$$

The following example will illustrate this approach.

EXAMPLE 7.2

A binary search method is used to find the transition between code 0 and code 1 of the ADC in Figure 7.9. The code edge is found to be 53 mV. A second binary search determines the code edge between codes 6 and 7 to be 2.77 V. What is the average LSB step size for this 3-bit ADC? Based on the data contained in the histogram of Figure 7.10, what is the width of each of the 8 codes, in volts?

Solution:

The average LSB size is equal to

$$V_{LSB} = \frac{2.77\ \text{V} - 0.053\text{V}}{2^3 - 2} = 452.8\ \text{mV}$$

Therefore, the code width for each code is:

Code 0: Undefined (infinite width)
Code 1: 0.706 LSBs × 452.8 mV = 319.68 mV
Code 2: 1.412 LSBs × 452.8 mV = 639.35 mV
Code 3: 0.882 LSBs × 452.8 mV = 399.37 mV
Code 4: 1.412 LSBs × 452.8 mV = 639.35 mV
Code 5: 0.882 LSBs × 452.8 mV = 399.37 mV
Code 6: 0.706 LSBs × 452.8 mV = 319.68 mV
Code 7: Undefined (infinite width)

If we wish to calculate the absolute voltage level of each code edge, we simply perform a running sum on the code widths expressed in volts, starting with the voltage V_{LE}, as follows

$$V_{CodeEdge}(i)=\begin{cases} V_{LE}, & i=0 \\ V_{LE}+\sum_{k=1}^{i} V_{CodeWidth}(k), & i=1,2,\ldots,2^D-2 \end{cases} \tag{7.10}$$

Alternatively, we can write a recursive equation for the code edges as follows

$$V_{CodeEdge}(i)=V_{CodeEdge}(i-1)+V_{LSB}\times V_{CodeWidth}(i), \quad i=1,2,\ldots,2^D-2 \tag{7.11}$$

where we begin with $V_{CodeEdge}(0) = V_{LE}$. The resulting code edge transfer curve is equivalent to a DAC output transfer curve, except that it will only have $2^D - 1$ values rather than 2D values.

EXAMPLE 7.3

Using the results of Example 7.2, reconstruct the 3-bit ADC transfer curve for each decision level.

Solution:
The transition from code 0 to code 1 was measured using a binary search. It was 53 mV. The other codes edges can be calculated using a running sum:

Code 0 to Code 1: 53 mV
Code 1 to Code 2: 53 mV + 319.68 mV = 372.68 mV
Code 2 to Code 3: 372.68 mV + 639.35 mV = 1011.9 mV
Code 3 to Code 4: 1011.9 mV + 399.37 mV = 1411.5 mV
Code 4 to Code 5: 1411.5 mV + 639.35 mV = 2050.8 mV
Code 5 to Code 6: 2050.8 mV + 399.37 mV = 2450.4 mV
Code 6 to Code 7: 2450.4 mV + 319.68 mV = 2770.0 mV

7.2.6 Accuracy Limitations of Histogram Testing

The accuracy of any code width or edge is inversely proportional to the average number of hits per code. Consider an input ramp to an ADC that extends over the ADC input range, V_{UE}–V_{LE}, with ramp duration T_R. If N_1 samples are collected from the ADC at a sampling rate of F_S over this time duration, then we can write

$$T_R=\frac{N_1}{F_S} \tag{7.12}$$

Furthermore, if N_1 samples are collected over the ADC input range then each sample represents the response to a voltage change ΔV given by

$$\Delta V=\frac{V_{UE}-V_{LE}}{N_1} \tag{7.13}$$

Exercises

7.4.	A linear histogram test was performed on an unsigned 4-bit ADC, resulting in the following distribution of code hits beginning with code 0 4, 5, 5, 7, 8, 4, 2, 4, 4, 3, 6, 3, 4, 6, 5 , 9 A binary search was performed on the first transition between codes 0 and 1 and found the code edge to be at 125 mV. A second binary search was performed and found the code edge between codes 14 and 15 to be 3.542 V. What is the average LSB size for this 4-bit ADC? Determine the width of each code, in volts.	ANS. LSB = 224.1 mV; code 0: undefined, code 1: 258.9 mV; code 2: 258.9 mV; code 3: 362.4 mV; code 4: 414.2 mV, code 5: 207.1 mV; code 6: 103.5 mV; code 7: 207.1 mV; code 8: 207.1 mV; code 9: 155.3 mV; code 10: 310.6 mV; code 11: 155.3 mV; code 12: 207.1 mV; code 13: 310.6 mV; code 14: 258.9 mV; code 15: undefined.
7.5.	For the distribution of code hits obtained for the 4-bit ADC listed in Exercise 7.4, determine the location of the code edges.	ANS. Beginning with code 0–1 transition: 0.1250 V, 0.3839 V, 0.6427 V, 1.0051 V, 1.4193 V, 1.6264 V, 1.7300 V, 1.9370 V, 2.1441 V, 2.2995 V, 2.6101 V, 2.7654 V, 2.9725 V, 3.2831 V, 3.5420 V.

Conversely, we can also express this voltage step or voltage resolution in terms of the average number of code hits $H_{Average}$ and the LSB step size by combining Eqs. (7.6) and (7.8) with (7.13) to arrive at

$$\Delta V = \frac{V_{LSB}}{H_{Average}} \tag{7.14}$$

By dividing each side of this equation by V_{LSB}, we obtain voltage resolution expressed in LSBs as

$$\frac{\Delta V}{V_{LSB}} = \pm\frac{1}{2}\frac{1}{H_{Average}} \quad [\text{LSB}] \tag{7.15}$$

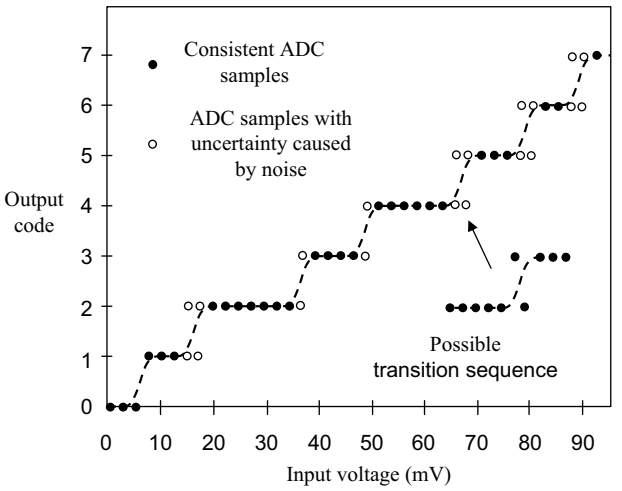
For example, if we measure an average of 5 hits per code, then the code width or code edge would, on average, have one-fifth of an LSB of resolution. If one LSB step size is equivalent to 452.8 mV, as in the last example, then the code width and edge would have a possible error of ±45.28 mV. To improve the accuracy of the histogram test, the average number of hits per code must be increased.

To understand how the average number of hits per code can be increased, consider combining Eqs. (7.12), (7.13), and (7.14), together with Eq. (7.8), to arrive at

$$H_{Average} = \frac{F_S}{2^D - 2} \times T_R \tag{7.16}$$

Clearly, a higher average number of hits per code is achiveable by using a longer ramp duration, a higher ADC sampling frequency, or a smaller ADC resolution. The latter two parameters are generally set by the DUT, so the test engineer really has only one option: Run the ramp very slowly.

Figure 7.11. Uncertainty caused by random noise.



This, in turn, drives up the time of the test. Nonetheless, for characterization this is an acceptable solution. Typically, code hits on the order of several hundreds is selected. The larger sample set also helps to improve the repeatibility of the test.

In production testing, however, we can only afford to collect a relatively small number of samples from each code, typically 16 or 32. Otherwise the test time becomes excessive. Therefore, even a perfect ADC will not produce a flat histogram in production testing because the limited number of samples collected gives rise to a limited code width resolution and repeatability. We can see that the samples in Figure 7.9 are spread too far apart to resolve small fractions of an LSB.

In addition to the accuracy limitation caused by limited resolution, we also face a repeatability limitation.[2] If we look carefully at Figure 7.9, we notice that several of the codes occur so close to a decision level that the ADC noise will cause the results to vary from one test execution to the next. This variability will happen *even if our input signal is exactly the same during each test execution*. Figure 7.11 illustrates the uncertainty in output codes caused by noise in the ADC circuits.

In many cases, we find that the raw data sequence from the ADC may zigzag up and down as the output codes near a transition from one code to the next. In Figure 7.11, for instance, we see that it is possible to achieve an ADC output sequence 4, 4, 4, 4, 4, 5, 4, 5, 5, 5 rather than the ideal sequence 4, 4, 4, 4, 4, 4, 5, 5, 5, 5. Unfortunately, this is the nature of histogram testing of ADCs. The results will be variable and somewhat unrepeatable unless we collect many samples per code. In histogram testing, as in many other tests, there is an inherent tradeoff between good repeatability and low test time. It is the test engineer's responsibility to balance the need for low test time with the need for acceptable accuracy and repeatability.

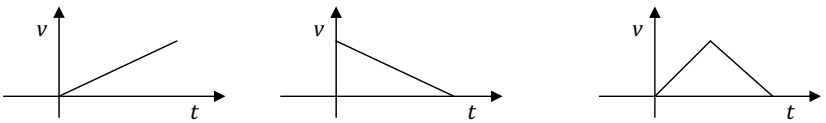
7.2.7 Rising Ramps Versus Falling Ramps

Most ADC architectures include one or more analog comparators in their design. Since comparators may be subject to hysteresis, we occasionally find a discrepancy between code edges measured using a rising ramp and code edges measured using a falling ramp. The most complete way to test an ADC is to test parameters such as INL and DNL using both a rising ramp and a falling

Exercises

7.6.	A 10-bit 5-V ADC operates with a sampling frequency of 1 MHz. If a linear histogram test is to be conducted on this ADC, what should be the minimum duration of the ramp signal so that the average code count is at least 10 hits? 100 hits?	ANS. T_R > 10.22 ms; T_R > 102.2 ms.
7.7.	A 10-bit 5-V ADC operates with a sampling frequency of 1 MHz. If a linear histogram test is to be conducted on this ADC with a ramp signal of 15-ms duration, estimate the voltage resolution of this test.	ANS. ΔV = 332.6 μV.
7.8.	A 10-bit 5-V ADC operates with a sampling frequency of 1 MHz. If a linear histogram test is to be conducted on this ADC with a ramp signal of 15-ms duration, how many samples need to be collected for processing?	ANS. N = 15,000.

Figure 7.12. Types of linear histogram inputs.



Up ramp Down ramp Triangle waveform

ramp. Both methods must produce a passing result before the ADC is considered good. However, the extra test doubles the test time; thus we prefer to use only one ramp. If characterization shows that we have a good match between the rising ramp and falling ramp, then we can drop back to a single test for production. Alternatively, if characterization shows that either the rising ramp or falling ramp always produces the worst-case results, then we can use only the worst-case test condition to save test time.

A compromise solution is to ramp the signal up at twice the normal rate and then ramp it down again (Figure 7.12). This triangle waveform approach tests both the falling and rising edge locations, averaging their results. It takes no longer than a single ramp technique, but it cancels the effects of hysteresis. A separate test could then be performed to verify that the ADC's hysteresis errors are within acceptable limits. The hysteresis test could be performed at only a few codes, saving test time compared to the two-pass ramp solution.

7.2.8 Sinusoidal Histogram Method

Sinusoidal histogram tests were originally used to compensate for the relatively poor linearity of early AWG instruments. Since it is easier to produce a pure sinusoidal waveform than to produce a perfectly linear ramp, early testers often relied on sinusoidal histogram testing for high-resolution ADCs. A second, more common reason to use the sinusoidal histogram method is that it allows better characterization of the dynamic performance of the ADC. The linear histogram technique is basically a static performance test. Because the input voltage is ramped slowly, the input level only changes by a fraction of an LSB from one ADC sample to the next. Sometimes we need to

test the ADC transition levels in a more dynamic, real-world situation. To do this, we can use a high-frequency sinusoidal input signal. Our goal is to make the ADC respond to the rapidly changing inputs of a sinusoid rather than the slowly varying voltages of a ramp. In theory, we could use a high-frequency triangle wave to achieve this result, but high-frequency linear triangles are much more difficult to produce than high-frequency sinusoids.

Ramp inputs have an even distribution of voltages over the entire ADC input range. Sinusoids, on the other hand, have an uneven distribution of voltages. A sine wave spends much more time near the upper and lower peak than at the center. As a result, we would expect to get more code hits at the upper and lower codes than at the center of the ADC's transfer curve, even when testing a perfect ADC. Fortunately, the distribution of voltage levels in a pure sinusoid is well defined; thus we can compensate for the uneven distribution of voltages inherent to sinusoidal waveforms.

Figure 7.13 shows a sinusoidal waveform that is quantized by a 4-bit ADC. Notice that there are only 15 decision levels in a 4-bit ADC and that the sine wave is programmed to exceed the upper and lower decision levels by a fairly wide margin. The reason we program the sine wave to exceed the ADC's full-scale range is that we have to make sure that the sine wave passes through all the codes if we want to get a histogram of all code widths. If we expand the time scale to view a quarter period of the waveform, we can see how the distribution of output codes is nonuniform due to the sinusoidal distribution of voltages, as shown in Figure 7.14. Clearly we get more code hits near the peaks of the sine wave than at the center, even for this simple example.

In order to understand the details of this test setup more clearly, consider the illustration shown in Figure 7.15. The diagram consists of three parts. At the center is the transfer characteristic of a 4-bit ADC, that is, 16 output code levels expressed as a function of the ADC input voltage. Below the ADC input voltage axis is a rotated graph of the ADC input voltage as a function of time. Here the input signal is a sine wave of amplitude peak and with DC offset, Offset, all

Figure 7.13. Sinusoidal input for 4-bit ADC.

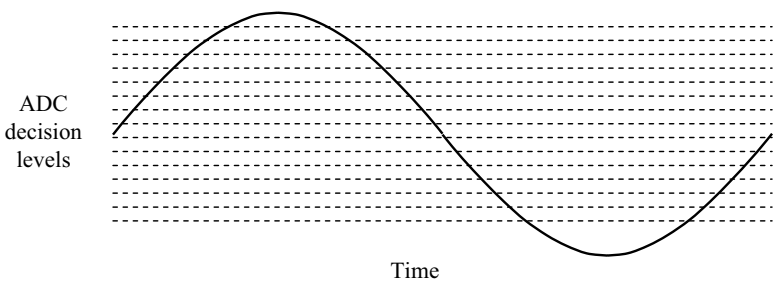


Figure 7.14. Close-up view of 4-bit ADC sinusoidal input.

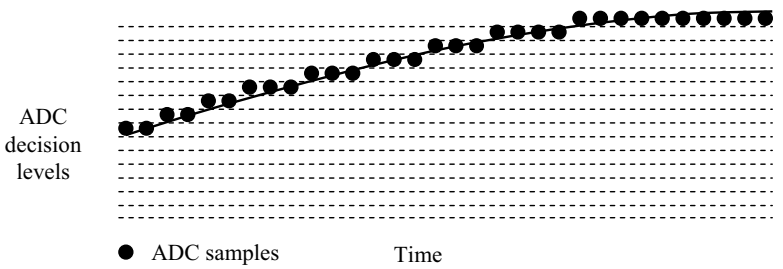
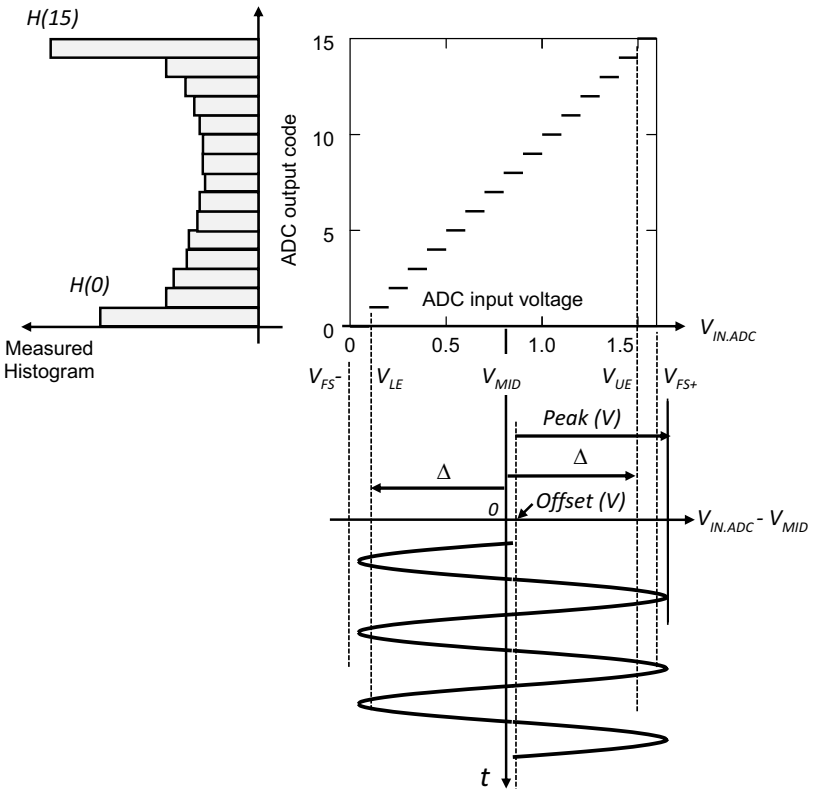


Figure 7.15. Sinusoidal histogram for an ideal ADC.



referenced to the middle level of the ADC input as defined by the distance between the upper and lower decision levels (V_{UE} and V_{LE}), that is,

$$V_{MID} = V_{LE} + \Delta \tag{7.17}$$

where

$$\Delta = \frac{V_{UE} - V_{LE}}{2} \tag{7.18}$$

Looking back at the expression for the LSB step size V_{LSB} in Eq. (7.8), we recognize that we can also write Δ as

$$\Delta = \left(2^{D-1} - 1\right)V_{LSB} \tag{7.19}$$

On the left-hand side of the ADC transfer curve, we have plot of a histogram of the ADC code levels (albeit rotated by 90 degrees). Here we see that the histogram exhibits a "bathtub"-like shape. If we try to use this histogram result the same way we use the linear ramp histogram results, the upper and lower codes would appear to be much wider than the middle codes. Clearly, we need to normalize our histogram to remove the effects of the sinusoidal waveform's nonuniform voltage distribution.

The normalization process is slightly complicated because we do not really know what the gain and offset of the ADC will be a priori. Additionally, we may not know the exact offset and amplitude of the sinusoidal input waveform. Fortunately, we have a piece of information at our disposal that tells us the level and offset of the signal as the ADC sees it.

The number of hits at the upper and lower codes in our histogram can be used to calculate the input signal's offset and amplitude. For example, in Figure 7.13, we can see that we will get more hits at the lower code than at the upper code. The lower codes will be hit more often because the sinusoid has a negative offset. The mismatch between these two numbers tells us the offset, while the number of total hits tells us the amplitude. Consider the pdf for the input voltage seen by the ADC is

$$f(v)=\begin{cases}\dfrac{1}{\pi\sqrt{\text{peak}^2-(v-\text{offset})^2}}, & -\text{peak}+\text{offset}\le v\le \text{peak}+\text{offset}\\ 0, & \text{otherwise}\end{cases} \tag{7.20}$$

The probability that the input signal is less than the lowest code decision level now defined by -Δ(i.e., relative to the ADC mid-level) is given by

$$P(V_{IN,ADC}<-\Delta)=\int_{-\text{Peak+Offset}}^{-\Delta}\frac{1}{\pi\sqrt{\text{peak}^2-(v-\text{offset})^2}}\,dv \tag{7.21}$$

which, after some calculus and algebra, reduces to

$$P(V_{IN,ADC}<-\Delta)=\frac{1}{\pi}\left[\sin^{-1}\left(\frac{-\Delta-\text{offset}}{\text{peak}}\right)+\frac{\pi}{2}\right] \tag{7.22}$$

Likewise, the probability that the input signal is larger than the highest code decision level +Δ is found in a similar manner as

$$P(\Delta<V_{IN,ADC})=\int_{\Delta}^{\text{peak+offset}}\frac{1}{\pi\sqrt{\text{peak}^2-(v-\text{offset})^2}}\,dv=\frac{1}{\pi}\left[\frac{\pi}{2}-\sin^{-1}\left(\frac{\Delta-\text{offset}}{\text{peak}}\right)\right] \tag{7.23}$$

If N samples are collected from the ADC output (including end code counts), then the expected number of code hits for code 0 and code 2^D-1 is simply given by the

$$\begin{aligned}H(0)&=N\times P(V_{IN,ADC}<-\Delta)=\frac{N}{\pi}\left[\sin^{-1}\left(\frac{-\Delta-\text{offset}}{\text{peak}}\right)+\frac{\pi}{2}\right]\\ H(2^D-1)&=N\times P(\Delta<V_{IN,ADC})=\frac{N}{\pi}\left[\frac{\pi}{2}-\sin^{-1}\left(\frac{\Delta-\text{offset}}{\text{peak}}\right)\right]\end{aligned} \tag{7.24}$$

Here we see we have two equations and two unknowns, which leads to the following solution:

$$\text{offset}=\left(\frac{C_2-C_1}{C_2+C_1}\right)\Delta=\left(\frac{C_2-C_1}{C_2+C_1}\right)(2^{D-1}-1)V_{LSB} \tag{7.25}$$

and

$$\text{peak} = \frac{2}{C_2 + C_1}\Delta = \frac{2}{C_2 + C_1}\left(2^{D-1} - 1\right)V_{LSB} \tag{7.26}$$

where

$$C_1 = \cos\left(\pi \frac{H\left(2^D - 1\right)}{N}\right) \quad \text{and} \quad C_2 = \cos\left(\pi \frac{H(0)}{N}\right)$$

We should note that N should be large enough that each ADC code is hit at least 16 times. The common rule of thumb is to collect at least 32 samples for each code in the ADC's transfer curve. For example, an 8-bit converter would require $2^8 \times 32 = 8192$ samples. Of course, some codes will be hit more often than 32 times and some will be hit less often than 32 times due to the curved nature of the sinusoidal input.

Once we know the values of peak and offset, we can calculate the ideal sine wave distribution of code hits, denoted $H_{sinewave}$, that we would expect from a perfectly linear ADC excited by a sinusoid. The equation for the ith code count, once again, excluding the upper and lower code counts, is

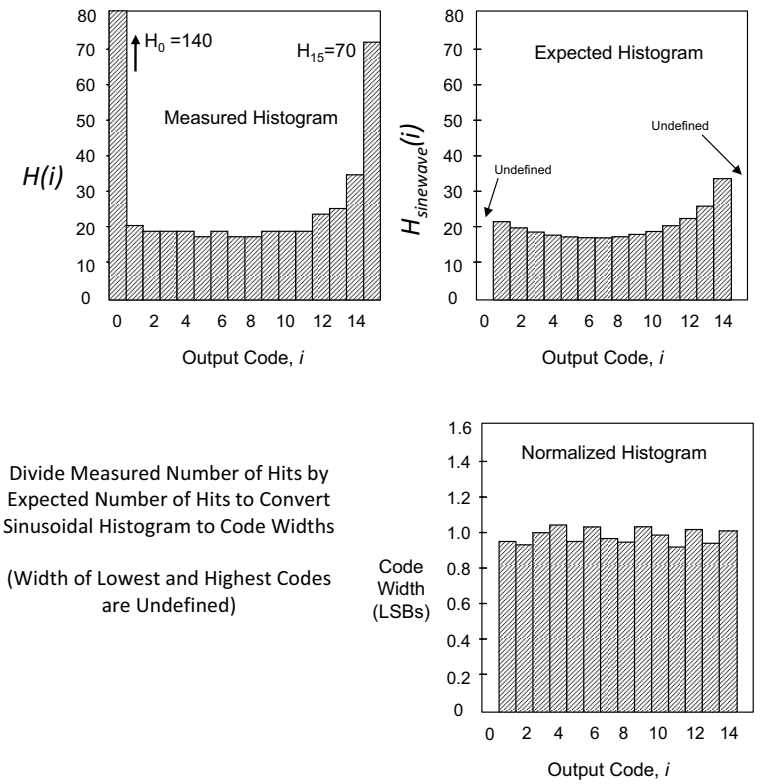
( ) — ( ) ( )

$$\text{LSB code width } i = \frac{H\ i}{H_{sinewave}\ i}, \quad i = 1, 2, \ldots, 2^D - 2 \tag{7.28}$$

Figure 7.16 illustrates the sinusoidal histogram normalization process for an idealized 4-bit ADC. Once we have calculated the normalized histogram, we are ready to convert the code widths into a code edge plot, using the same steps as we used for the linear ramp histogram method.

This example is based on an ideal ADC with equal code widths. Even with this idealized simulation, the normalized histogram does not result in equal code width measurements. This simulated example was based on a sample size of 32 samples per ADC code (16 ADC codes × 32 samples per code = 512 collected samples). As we can see in Figure 7.16, many of the codes were hit fewer than 20 times in this simulation. Like the linear ramp histogram method, the number of hits per code limits the measurement resolution of a sinusoidal histogram. If we had collected hundreds of samples for each code in this 4-bit ADC example, the results would have been much closer to a flat histogram. Also, the repeatability of code width measurements will improve with a larger sample size. Unfortunately, a larger sample size requires a longer test time. Again, we are faced with a tradeoff between low test time and high accuracy. We'll explore this in greater detail shortly. Let us first look at an example.

Figure 7.16. Sinusoidal histogram for an ideal 4-bit ADC (simulated).



EXAMPLE 7.4

The distribution of code hits for an two's complement 4-bit ADC excited by a sinusoidal signal beginning with code –8 is as follows

170, 61, 55, 48, 44, 42, 40, 39, 39, 40, 41, 42, 45, 50, 72, 196

A binary search was performed on the first transition between codes –8 and –7 and found the code edge to be at 330 mV. A second binary search was performed and found the code edge between codes 6 and 7 to be 4.561 V. What is the average LSB size for this 4-bit ADC? What is the mid-level of the ADC input? What is the offset and amplitude of the sinusoidal signal seen by the ADC relative to its mid-level? What is expected or ideal sinusoidal distribution of code hits corresponding to this input signal? Determine the width of each code, as well as the code edges (all in volts). Plot the transfer characteristics of this ADC.

Solution:
According to Eq. (7.8) we find the average LSB step size is

$$V_{LSB} = \frac{4.561\,\text{V} - 0.330\text{V}}{2^4 - 2} = 302.2\ \text{mV}$$

Next, we compute the sinusoidal signal seen by the ADC relative to the input mid-level according to the following

$$V_{MID} = V_{LE} + \Delta = 0.330 + 2.115 = 2.445 \text{ V}$$

where

$$\Delta = \frac{V_{UE} - V_{LE}}{2} = \frac{4.561 - 0.330}{2} = 2.115 \text{ V}$$

The parameters of the sine wave seen by the ADC relative to mid-level is then found from the code hits data such that

$$C_1 = \cos\left(\pi \frac{H\left(2^D - 1\right)}{N}\right) = \cos\left(\frac{196}{1024}\pi\right) = 0.8245,$$
$$C_2 = \cos\left(\pi \frac{H(0)}{N}\right) = \cos\left(\frac{170}{1024}\pi\right) = 0.8670$$

leading to

$$peak = \frac{2}{C_2 + C_1}\Delta = \frac{2}{0.8670 + 0.8245} \times 2.115 = 2.5011 \text{ V}$$

$$offset = \left(\frac{C_2 - C_1}{C_2 + C_1}\right)\Delta = \left(\frac{0.8670 - 0.8245}{0.8670 + 0.8245}\right) \times 2.115 = 0.05309 \text{ V}$$

Next, the expected number of code hits for an ideal ADC is found from Eq. (7.27), resulting in the following list of code hits beginning code –7 and ending with code 6:

$H_{sinewave}$ = 67.42, 54.33, 47.78, 44.02, 41.67, 40.23, 39.56, 39.44, 39.88, 41.09, 43.10, 46.25, 51.56, 61.50

Subsequently, the width of each code (-7 to 6) expressed in LSBs is found from Eq. (7.28) to be

code width = 0.9048, 1.012, 1.005, 0.9995, 1.008, 0.9943, 0.9858, 0.9888, 1.003, 0.9978, 0.9745, 0.9730, 0.9697, 1.171

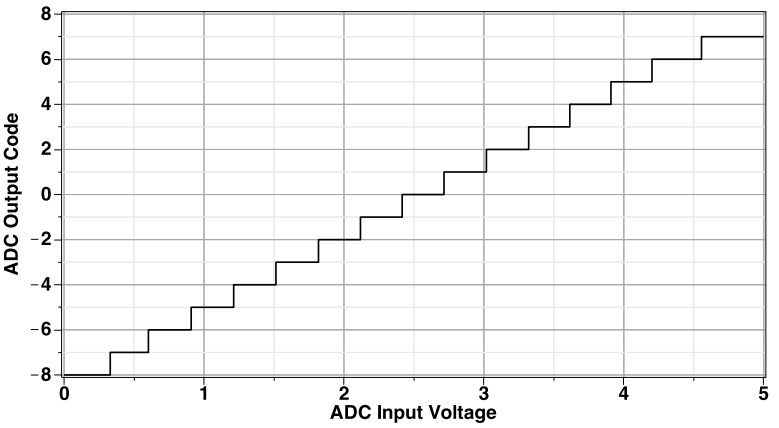
The code width of each code (-7 to 6) is scaled by V_{LSB} to obtain the code width in volts:

code width = 0.2734, 0.3058, 0.3037, 0.3020, 0.3046, 0.3005, 0.2979, 0.2988, 0.3031, 0.3015, 0.2945, 0.2940, 0.2930, 0.3539

Finally, the code edges in are found through application of Eq. (7.11) using the above code widths in volts:

code edges = 0.330, 0.6034, 0.9092, 1.213, 1.515, 1.820, 2.120, 2.418, 2.717, 3.020, 3.322, 3.616, 3.910, 4.203, 4.557

A plot of transfer characteristics corresponding to this ADC is shown below:



Exercises

7.9. The distribution of code hits for an unsigned 4-bit ADC excited by a sinusoidal signal beginning with code 0 is as follows

137, 80, 52, 60, 40, 51, 36, 48, 36, 48, 37, 52, 42, 64, 80, 160

A binary search was performed on the first transition between codes 0 and 1 and found the code edge to be at –4.921 V. A second binary search was performed and found the code edge between codes 14 and 15 to be 4.952 V. What is the offset and amplitude of the input sinusoidal signal? What is expected or ideal sinusoidal distribution of code hits? Finally, what is the distribution of code widths (in volts) for this ADC?

ANS. Offset = 0.0849 V; peak = 5.500 V.

Ideal sinusoidal distribution (code 1 to 14): 80.75, 60.56, 51.95, 47.19, 44.36, 42.70, 41.90, 41.81, 42.43, 43.86, 46.37, 50.55, 57.90, 73.60.

Code widths (code 1 to 14): 0.7003, 0.6060, 0.8159, 0.5980, 0.8103, 0.5948, 0.8082, 0.6070, 0.7983, 0.5949, 0.7912, 0.5864, 0.7800, 0.7694.

To gain a better idea of the distribution of codes and its impact on the performance of the sinusoidal histogram test, consider the minimum and maximum number of code hits corresponding to an ideal D-bit ADC excited by a peak-to-peak sine wave signal equal to the full-scale range of the ADC with frequency F_T as follows:

$$\begin{aligned}\text{maximum number of code hits} &= \frac{1}{\pi}\left(\frac{F_S}{F_T}\right)\left[\frac{\pi}{2} - \sin^{-1}\left(1 - \frac{1}{2^{D-1}}\right)\right] \\ \text{minimum number of code hits} &= \frac{1}{2\pi}\left(\frac{F_S}{F_T}\right)\sin^{-1}\left(\frac{1}{2^{D-1}}\right)\end{aligned} \tag{7.29}$$

While not explicit in the above equation, one can show that the total number of samples collected from the ADC output when excited by a single cycle of the sine wave input is given by

$$N = \frac{F_S}{F_T} \tag{7.30}$$

Like before, both F_S and D are parameters of the DUT, leaving F_T as the only parameter that the test engineer can use to optimize the test (e.g., minimize test time). The lower the test frequency, F_T, the greater the number of minimum code hits and, in turn, the longer the test time. The resolution of the measurement is bounded by the largest step size that takes place during the sampling process. Each step change in the input signal level can be described by

$$\Delta V[n] = \frac{V_{FSR}}{2}\left\{\sin\left[2\pi\frac{F_T}{F_S}(n+1)\right] - \sin\left[2\pi\frac{F_T}{F_S}n\right]\right\} \tag{7.31}$$

The largest step change occurs around the zero crossing point of the sine wave, resulting in the sinusoidal historgam method having a worst-case voltage resolution of

$$\max\{\Delta V\} = \frac{V_{FSR}}{2}\sin\left(2\pi\frac{F_T}{F_S}\right) \tag{7.32}$$

If we assume the full-scale range is equivalent to $2DV_{LSB}$, Eq. (7.32) can be rewrtitten as

$$\max\{\Delta V\} = 2^{D-1}V_{LSB}\sin\left(2\pi\frac{F_T}{F_S}\right) \tag{7.33}$$

Furthermore, because F_T is typically much smaller than F_S, we can approximate the worst-case voltage resolution of the sinusoidal histogram test as

$$\max\{\Delta V\} = \pm 2^{D-1}\pi V_{LSB}\frac{F_T}{F_S} \tag{7.34}$$

Clearly, the lower the test frequency, the higher the resolution but longer the test time.

Exercises

7.10. A 10-bit 5-V ADC operates with a sampling frequency of 1 MHz. If a sinusoidal histogram test is to be conducted on this ADC, what should be the maximum frequency of the input signal so that the average minimum code count is at least 10 hits? 100 hits?

ANS. F_T = 1865.1 Hz; F_T = 186.51 Hz.

7.11. A 10-bit 5-V ADC operates with a sampling frequency of 1 MHz. If a sinusoidal histogram test is to be conducted on this ADC with a test time of no more than 15 ms, estimate the worst-case voltage resolution for this test.

ANS. ΔV = ±1.047 mV; ΔV = ±0.1072 LSB.

7.12. A 10-bit 5-V ADC operates with a sampling frequency of 1 MHz. If a sinusoidal histogram test is to be conducted on this ADC with an input frequency of 100 Hz, how many samples need to be collected for processing? How long will it take to collect these samples?

ANS. N = 10,000; test time = 10 ms.

7.3 DC TESTS AND TRANSFER CURVE TESTS

7.31. DC Gain and Offset

Once we have produced a code edge transfer curve for an ADC, we can test the ADC much as we would test a DAC. Since a code edge transfer curve is a one-to-one mapping function, we can apply all the same DC and transfer curve tests outlined in Chapter 6, "DAC Testing." There are a few minor differences to consider. For example, a *D*-bit ADC has one fewer code edge than an *D*-bit DAC has outputs. A more important difference is that the ideal ADC transfer curve may be ambiguously defined. The test engineer should realize that there are several ways to define the ideal performance of an ADC.

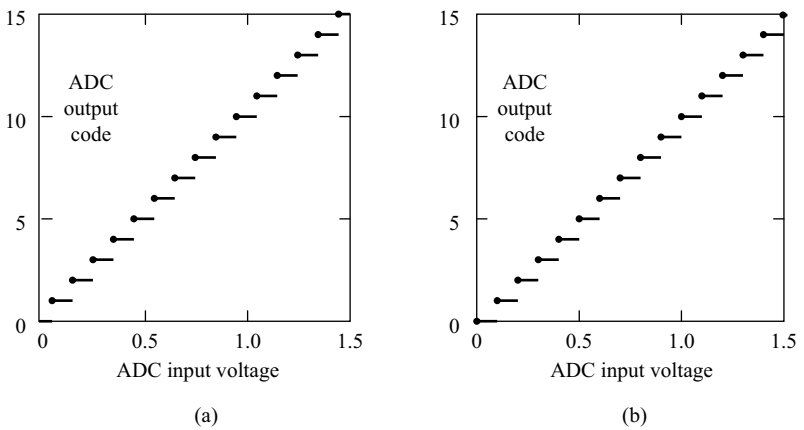
Figure 7.17 shows two alternate definitions of an 8-bit ADC's ideal performance. The first alternative is to define the ideal location of the first code edge, V_{LE}, at a voltage corresponding to +½ LSB above the V_{FS-} level. The second alternative is to define the ideal location of the first code edge at a voltage corresponding to +1 LSB above V_{FS-}.

When measuring DC offsets and other absolute voltage levels, it is very important that we understand exactly what the ideal transfer curve is supposed to be. Otherwise, we may introduce errors of ±½ LSB. Unfortunately, there is little consistency from one ADC data sheet to the next as to the intended ideal performance. This is another issue that the test engineer must clarify before writing the test program.

Once the ideal curve has been established, DC gain and offset can be measured in a manner similar to DAC DC gain and offset. The gain and offset are measured by calculating the slope and offset of the best-fit line. If the converter is defined using the definition illustrated in Figure 7.17a, we have to remember that the ideal line would have an offset of +½ LSB.

Unfortunately, there are many other ways to define gain and offset. In some data sheets, the offset is defined simply as the offset of the first code edge from its ideal position and the gain is defined as the ratio of the actual voltage range divided by the ideal voltage range from V_{FS-} to V_{FS+}. Other definitions abound; thus, the test engineer is responsible for determining the correct methodology for each ADC to be tested. Of course, ambiguities in the data sheet should be clarified to prevent correlation headaches caused by misunderstandings in data sheet definitions.

Figure 7.17. Alternate definitions of ADC transfer curves.



7.3.2 INL and DNL

Except for the fact that an ADC code edge transfer curve has one fewer value than an equivalent DAC curve, we can calculate ADC INL and DNL exactly the same way as DAC INL and DNL. If we use the histogram method, we can take a shortcut in measuring INL and DNL. Specifically, once the code widths are known, the endpoint DNL expressed in units of LSBs can be determined by subtracting one LSB from each code width as follows

$$\text{endpoint } DNL(i) = \text{LSB code width}(i) - 1, \quad i = 1,2,\ldots,2^D - 2 \tag{7.35}$$

Subsequently, as described in Chapter 6, the DNL curve can then be integrated using a running sum to calculate the endpoint INL curve in units of LSBs according to the following

$$\text{endpoint } INL(i) = \begin{cases} 0 & i = 1 \\ \sum_{k=1}^{i-1} \text{Endpoint } DNL(k), & i = 2,3,\ldots,2^D - 2 \\ 0 & i = 2^D - 1 \end{cases} \tag{7.36}$$

Using this shortcut method, we never even have to compute the absolute voltage level for each code edge, unless we need that information for a separate test, such as gain or offset.

As with DAC INL and DNL testing, a best-fit approach is the preferred method for calculating ADC INL and DNL. As discussed in Chapter 6, "DAC Testing," best-fit INL and DNL testing results in a more meaningful, repeatable reference line than endpoint testing, since the best-fit reference line is less dependent on any individual code's edge location. We can convert an endpoint INL curve to a best-fit INL curve by first calculating the best-fit line for the endpoint INL curve, for example,

$$\text{best-fit endpoint } INL[i] = \text{gain} \times i + \text{offset}, \quad i = 1,...,2^D - 1$$

Subtracting the best-fit line from the endpoint INL curve yields the best-fit INL curve, that is,

$$\text{best-fit } INL[i] = \text{endpoint } INL[i] - \text{best-fit endpoint } INL[i], \quad i = 1,\ldots,2^D - 1 \tag{7.37}$$

The best-fit DNL curve is then calculated by taking the discrete time first derivative of the best-fit INL curve according to

$$\text{best-fit } DNL(i) = \text{best-fit } INL(i) - \text{best-fit } INL(i-1), \quad i = 1,2,\ldots,2^D - 2 \tag{7.38}$$

Notice that the histogram method captures an endpoint DNL curve and then integrates the DNL curve to calculate endpoint INL. This is unlike the DAC methodology and the ADC servo/search methodologies, which start with a measurement of absolute voltage levels to measure INL and then calculate the DNL through discrete time first derivatives. The following example will illustrate this method.

EXAMPLE 7.5

A linear histogram test was performed on an unsigned 4-bit ADC resulting in the following distribution of code hits beginning with code 0:

4, 5, 5, 7, 8, 4, 2, 4, 4, 3, 6, 3, 4, 6, 5, 9

Determine the best-fit DNL and INL characteristicsof this ADC.

Solution:

We begin by first finding the endpoint DNL characteristics for this ADC. As the average code hit is 4.714, we find that the code width (in LSBs) beginning with code 1 and ending with code 14 is:

Code Widths:

[0, undefined], [1, 1.061], [2, 1.061], [3, 1.485], [4, 1.697], [5, 0.8485], [6, 0.4243], [7, 0.8485], [8, 0.8485], [9, 0.6364], [10, 1.273], [11, 0.6364], [12, 0.8485], [13, 1.273], [14, 1.061], [15, undefined]

Subsequently, using Eq. (7.35), we find that the endpoint DNL characteristics beginning with the 0 to 1 code transition and ending with the 13th to 14th code transition is:

Endpoint DNL:

[1, 0.061], [2, 0.061], [3, 0.485], [4, 0.697], [5, –0.1515], [6, –0.5757], [7, –0.1515], [8, –0.1515], [9, –0.3636], [10, 0.273], [11, –0.3636], [12, –0.1515], [13, 0.273], [14, 0.061]

Integrating the DNL function according to Eq. (7.36), we find the endpoint INL characteristics beginning with code 1 and ending with code 15 as follows:

Endpoint INL:

[1, 0], [2, 0.061], [3, 0.122], [4, 0.607], [5, 1.304], [6, 1.152], [7, 0.5763], [8, 0.4248], [9, 0.2733], [10, –0.0903], [11, 0.1827], [12, –0.1809], [13, –0.3324], [14, –0.0594], [15, 0]

Using the regression analysis equations of Chapter 6, we find that the gain and offset of the best-fit line parameters associated with the endpoint INL curve is –0.04393 and 0.5769, respectively. The best-fit line corresponding to the endpoint INL is then given by the expression

$$\text{best-fit endpoint } INL[i] = -0.04393 \times i + 0.5769, \quad i = 1,\ldots,15$$

Evaluating this function for n from 1 to 15, and it subtracting from the endpoint INL data set, that is,

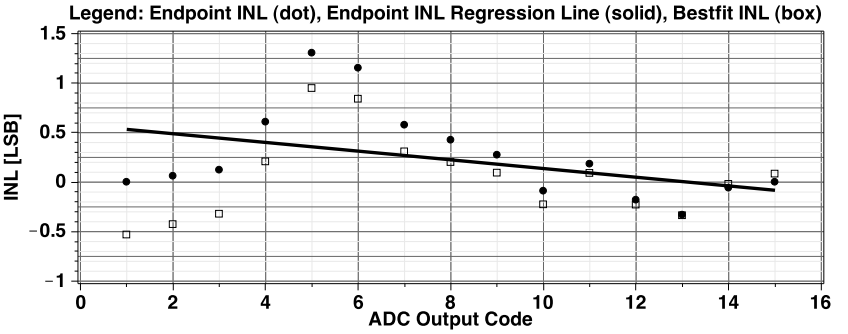
$$\text{best-fit } INL[i] = \text{endpoint } INL[i] - \text{best-fit endpoint } INL[i], \quad i = 1,\ldots,15$$

we obtain the following set of best-fit INL points:

Best-Fit INL:

[1, –0.5330], [2, –0.4280], [3, –0.3231], [4, 0.2058], [5, 0.9467], [6, 0.8387], [7, 0.3069], [8, 0.1993], [9, 0.0918], [10, –0.2279], [11, 0.0890], [12, –0.2306], [13, –0.3382], [14, –0.0213], [15, 0.0821]

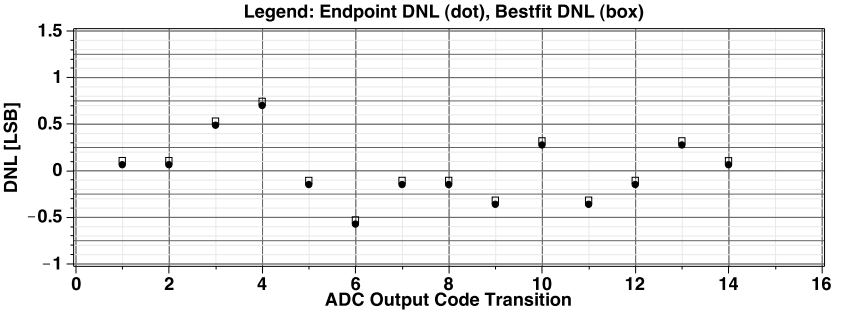
Below is a plot of three sets of data corresponding to the INL characteristics of the ADC: endpoint INL, the regression line for the endpoint INL, and the corresponding best-fit INL. As is clearly evident, the endpoint INL and best-fit INL are different.



Finally, we compute the best-fit DNL characteristics of the ADC by differentiating the best-fit INL curve using the first-order difference operation given in Eq. (7.38):

Best-Fit DNL:

[1, 0.1050], [2, 0.1049], [3, 0.5289], [4, 0.7409], [5, −0.1080], [6, −0.5318], [7, −0.1076], [8, −0.1075], [9, −0.3197], [10, 0.3169], [11, −0.3196], [12, −0.1076], [13, 0.3169], [14, 0.1034]



Above, we see from above that the DNL is very similar for both the endpoint and best-fit reference lines.

7.3.3 Monotonicity and Missing Codes

One final difference between ADC testing and DAC testing relates to differences in their weaknesses. For example, a DAC may be nonmonotonic, while an ADC will usually be monotonic *if it is tested statically*. For an ADC to be nonmonotonic, one or more of its code widths has to be

Exercises

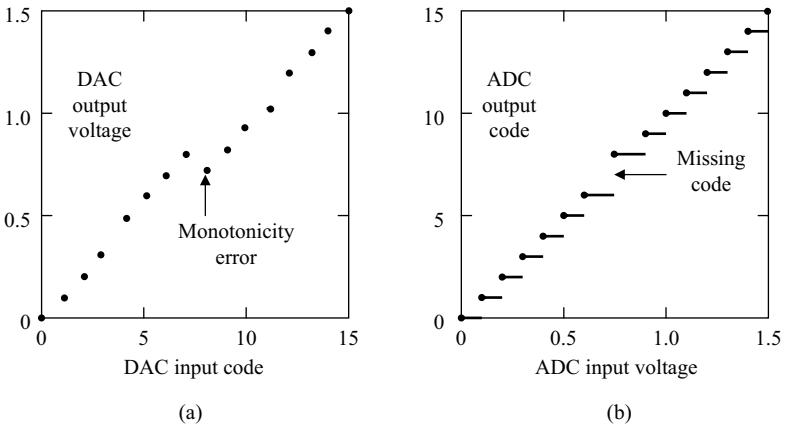
7.13. A linear histogram test was performed on an unsigned binary 4-bit ADC resulting in the following distribution of code hits beginning with code 0: 4, 5, 5, 7, 8, 7, 9, 5, 6, 3, 6, 7, 9, 6, 5, 9 Determine the endpoint DNL curve for this ADC.	ANS. DNL for code transitions from 1 to 14: –0.2045, –0.2045, 0.1136, 0.2727, 0.1136, 0.4318, –0.2045, –0.0455, –0.5227, –0.0455, 0.1136, 0.4318, –0.0455, –0.2045.
7.14. For the code distribution described in Exercise 7.13, determine the endpoint INL curve for this 4-bit ADC.	ANS. INL atcode edge 1 to 15:0, –0.2045, –0.4091, –0.2955, –0.0227, 0.0909, 0.5227, 0.3182, 0.2727, –0.2500, –0.2955, –0.1818, 0.2500, 0.2045, 0.
7.15. Compute the best-fit INL characteristic of the ADC described in Exercise 7.13.	ANS. INL atcode edge 1 to 15: 0.09521, –0.1254, –0.3460, –0.2480, 0.00905, 0.1071, 0.5231, 0.3025, 0.2410, –0.2977, –0.3592, –0.2612, 0.1548, 0.0933, –0.1286.
7.16. Compute the best-fit DNL characteristic of the ADC described in Exercise 7.13.	ANS. DNL for code transitions from 1 to 14: –0.2206, –0.2206, 0.0980, 0.2570, 0.09805, 0.4160, –0.2206, –0.0615, –0.5387, –0.0615, 0.0980, 0.4160, –0.0615, –0.2219.

negative. (One example of this is an ADC whose DC reference voltage is somehow drastically perturbed as the input voltage varies. However, this failure mechanism is quite rare.) Nevertheless, an ADC can *appear* to be nonmonotonic when its input is changing rapidly.[3]

For this reason, we do not typically test ADCs for monotonicity when we use slowly changing inputs (as in search or linear ramp INL and DNL tests). However, when testing ADCs with rapidly changing inputs, the ADC may behave as if it were nonmonotonic due to slew rate limitations in its comparator(s). These monotonicity errors show up as signal-to-noise ratio failures in some ADCs and as sparkling in others. (Sparkling is a dynamic failure mode discussed in Section 7.4.3.)

Unlike DACs, ADCs are often tested for missing codes. A missing code is one whose voltage width is zero. This means that the missing code can never be hit, regardless of the ADC's input voltage. A missing code appears as a missing step on an ADC transfer curve, as illustrated in Figure 7.18. Since DACs always produce a voltage for each input code, DACs cannot have missing codes. Although a true missing code is one that has zero width, missing codes are often defined as any code having a code width smaller than some specified value, such as 1/10 LSB. Technically, a code having a width of 1/10 LSB is not missing, but the chances of it being hit are low enough that it is considered to be missing from the ADC transfer curve.

Figure 7.18. (a) Monotonicity errors in DACs and (b) missing codes in ADCs.



7.4 DYNAMIC ADC TESTS

7.4.1 Conversion Time, Recovery Time, and Sampling Frequency

DACs have many dynamic tests such as settling time, rise and fall time, overshoot, and undershoot. ADCs do not exhibit these same features, since they do not have an analog output. Instead, an ADC may have any or all the following timing specifications: maximum sampling frequency, maximum conversion time, and minimum recovery time. There are many ways to design ADCs and ADC digital interfaces. Let us look at a few of the common interfacing strategies.

One common interface scheme is shown in Figure 7.19. The ADC begins a conversion cycle when the CONVERT signal is asserted high. After the conversion cycle is completed, the ADC asserts a DATA_READY signal that indicates the conversion is complete. Then the data are read from the ADC using a READ signal.

Maximum conversion time is the maximum amount of time it takes an ADC to produce a digital output after the CONVERT signal is asserted. The ADC is guaranteed to produce a valid output within the maximum conversion time. It is tempting to say that an ADC's maximum sampling frequency is simply the inverse of the maximum conversion time. In many cases this is true. Some ADCs require a minimum recovery time, which is the minimum amount of time the system must wait before asserting the next CONVERT signal. The maximum sampling frequency is therefore given by the equation

$$F_{max} = \frac{1}{T_{convert} + T_{recovery}} \tag{7.39}$$

We typically test $T_{convert}$ by measuring the period of time from the CONVERT signal's active edge to the DATA_READY signal's active edge. We have to verify that the $T_{convert}$ time is less than or equal to the maximum conversion time specification. For this measurement, we can use a time measurement system (TMS) instrument, or we can sometimes use the tester's digital pattern compare function if we can tolerate a less accurate pass/fail test. We can verify the F_{max} specification (and thus the $T_{recovery}$ specification) by simply operating the converter at its maximum sampling rate, F_{max}, and verifying that it passes all its dynamic performance specifications at this frequency.

Figure 7.19. ADC sample timing.

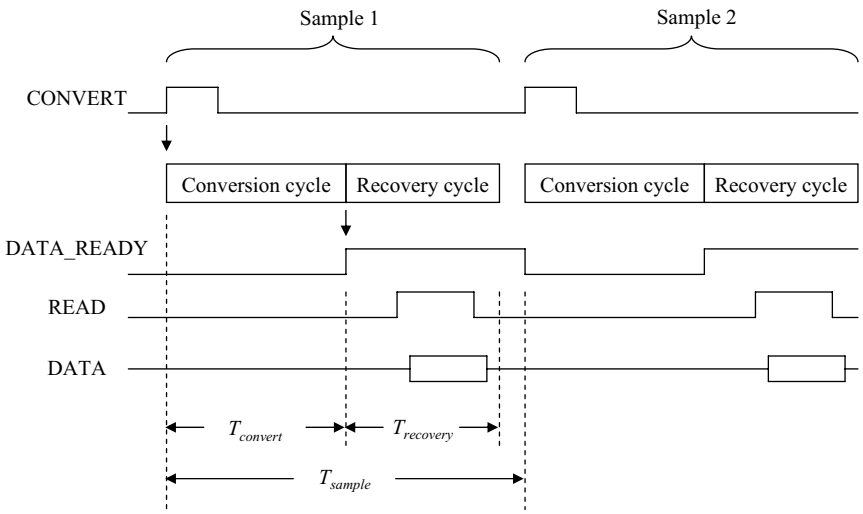
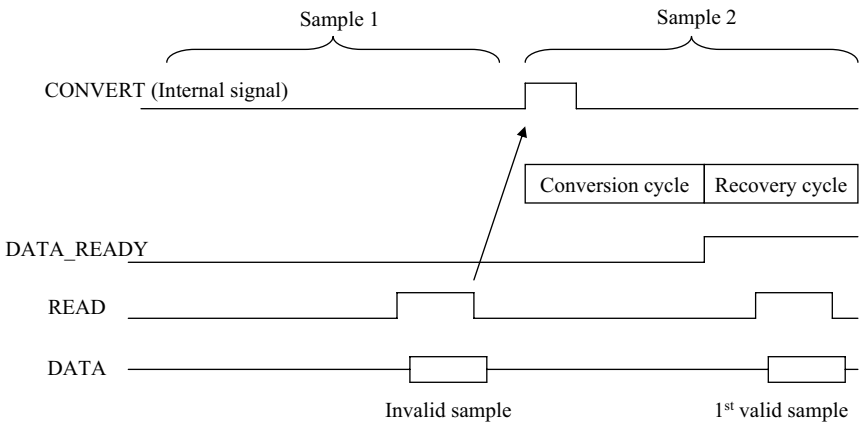


Figure 7.20. ADC conversion cycles with internally generated CONVERT signal.



In many ADC designs, the CONVERT signal is generated automatically after the ADC output data is read, as shown in Figure 7.20. This type of converter requires no externally supplied CONVERT signal. The first sample read from the ADC must therefore be discarded, since no conversion is performed until after the first READ pulse initiates the first conversion cycle.

Sometimes ADCs simply perform continuous conversions at a constant sampling rate. The CONVERT signal is generated at a fixed frequency derived from the device master clock. This architecture is very common in ADC channels such as those in a cellular telephone voice band interface or multimedia audio device. The continuous conversions can usually be disabled by a register bit or other control mechanism to minimize power consumption when conversions are not needed. These devices sometimes generate a DATA_READY signal that must be used to synchronize the tester with an asynchronous data stream. DUT-defined timing can be a difficult situation

to deal with, since ATE testers are not designed to operate in a slave mode with the DUT driving digital timing.

Clearly, there are many ways to design ADCs. The test engineer has to deal with many different permutations of interfacing possibilities, each with its own testing requirements.

7.4.2 Aperture Jitter

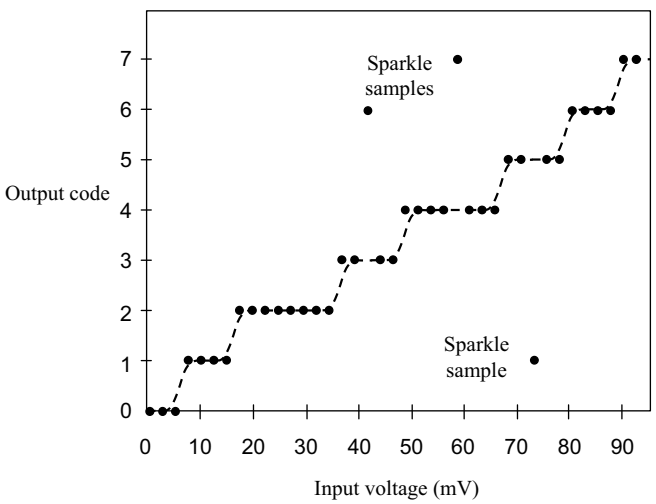
Noise associated with sampling clock can introduce error into the data conversion process. Typically, clock noise or what is also referred to as aperture jitter is guaranteed by acceptable signal-to-noise ratio (SNR) performance. It may or may not be tested in production, depending on the required sampling rate of the ADC. Very high-frequency ADCs typically must be tested for aperture jitter in production.

7.4.3 Sparkling

Sparkling is a phenomenon that happens most often in high-speed flash converters, such as those described ahead in Section 7.5.3, due to digital timing race conditions. It is the tendency for an ADC to occasionally produce a conversion that has a larger than expected offset from the expected value. We can think of a sparkle sample as one that is a statistical outlier from the Gaussian distribution in Figure 7.6. Sparkling shows up in a time-domain plot as sudden variations from the expected values. It got its name from early flash ADC applications, in which the sample outliers produced white sparkles on a video display. Sparkling is specified as a maximum acceptable deviation from the expected conversion result. For example, we might see a specification that states sparkling will be less than 2 LSBs, meaning that we will never see a sample that is more than 2 LSBs from the expected value (excluding gain and offset errors, of course). Sparkling should not be confused with noise-induced errors such as those illustrated in Figure 7.11.

Test methodologies for sparkling vary, mainly in the choice of input signal. We might look for sparkling in our ramp histogram raw data, such as that shown in Figure 7.21. We might also apply a very high-frequency sine wave to the ADC and look for time-domain spikes in the collected samples.

Figure 7.21. Sparkling in a linear ramp histogram sample set.



Since it is a random digital failure process, sparkling often produces intermittent test results. Sparkling is generally caused by a weakness in the ADC design that must be eliminated through good design margin rather than being screened out by exhaustive testing. Nevertheless, ADC sparkling tests are often added to a test program as a quick sanity check, making use of samples collected for one of the required parametric tests.

7.5 TESTS FOR COMMON ADC APPLICATIONS

7.5.1 DC Measurements

Like DACs, ADCs can be used for a variety of purposes. The ADC's application often determines its required parameters. For example, an ADC may be used to measure absolute voltage levels, as in a DC voltmeter or battery monitor. In this type of application, we do not usually care about transmission parameters like signal-to-noise ratio. We will typically only need to know the DC gain, DC offset, INL, DNL, and worst-case absolute voltage errors in decision levels, relative to the ideal decision levels. Idle channel noise will sometimes be specified, to ensure that results obtained from the ADC are not unrepeatable due to excessive noise.

Successive approximation ADCs and integrating ADCs are the most common converter type used for DC measurements. Sigma-delta designs[4,5] are seldom used due to their inherent tendency to produce self-tones with certain DC inputs.

7.5.2 Audio Digitization

Audio digitization is a very common application for ADCs, especially high-resolution ADCs. When the resolution exceeds 12 or 13 bits, it becomes very expensive to perform transfer curve tests such as INL and DNL because of the large number of code edges that must be measured. Fortunately, transmission parameters such as frequency response, signal to distortion ratio, idle channel noise, and so on, are more meaningful measures of audio digitizer performance. These sampled channel tests are much less time-consuming to measure than INL and DNL, especially when testing ADCs with 16 or more bits of resolution. Sigma–delta ADCs[5] have become the most common architecture for audio digitization application.

As previously mentioned, self-tones are a potential source of trouble when sigma-delta ADCs are used in audio digitization applications. Because of the way the human mind processes sound, very low-amplitude self-tones are much easier to hear than white noise at equivalent signal levels. It is impractical to test self-tones at every possible DC input level. Self-tones should at least be tested with the analog input tied to ground or $V_{DD}/2$ (or whatever voltage represents the converter's midscale input level). When characterization indicates that a particular ADC design is not prone to self-tone generation, then this test is often eliminated in production.

7.5.3 Data Transmission

Data transmission applications differ from audio applications mainly in terms of the sampling rates and the frequency range of the transmitted signals. Data transmission ADCs, such as those found in modems, hard disk drive read channels, and cellular telephone intermediate frequency (IF) sections, often digitize signals that are well above the audio band. These applications typically require lower-resolution ADCs, but may require much higher sampling rates. Aperture jitter is often a prime concern for these applications, especially if the signal frequency band extends past a few tens of megahertz. Excessive aperture jitter can introduce apparent noise in the digitized signal, ruining the performance of the ADC.

Signal-to-noise ratio, group delay distortion, and other transmission parameters are often specified in data transmission applications. Also, data transmission specifications such as error vector magnitude (EVM), phase trajectory error (PTE), and bit error rate (BER) may also need to be tested. Some of these parameters are so numerous that we cannot possibly cover them in this book. The test engineer will have to learn about these and other application-specific testing requirements by studying the relevant standards documents. ATE vendors can also be a tremendous source of expertise when learning about new testing requirements and methodologies.

Most ADC architectures are well suited for low-frequency data transmission applications (with the exception of integrating converters). High-frequency applications may require fast successive approximation ADCs, semiflash ADCs, or even full-flash ADCs, depending on the required sampling rates.

7.5.4 Video Digitization

NTSC video signal digitization is another key application for high-speed ADCs. These applications require the faster ADC types (flash, semiflash, or pipelined successive approximation ADCs). The test list for these types of converters usually includes transmission parameters as well as differential gain and differential phase measurements. Like other high-speed applications, aperture jitter is a key performance specification for video digitization applications. Sparkling is particularly noticeable in video applications, thus, this potential weakness should be thoroughly characterized and/or tested in production.

7.6 SUMMARY

ADC testing is very closely related to DAC testing. Many of the DC and intrinsic tests defined in this chapter are very similar to those performed on DACs. The most important difference is that the ADC code edge transfer curve is harder and much more time-consuming to measure than the DAC transfer curve. However, once the many-to-one statistical mapping of an ADC has been converted to a one-to-one code edge transfer curve, the DC and transfer curve tests are very similar in nature to those encountered in DAC testing. This chapter by no means represents an exhaustive list of all possible ADC types and testing methodologies. There is a seemingly endless variety of ADC architectures and methods for defining their performance. Hopefully, this chapter will provide a solid starting point for the beginning test engineer.

PROBLEMS

7.1. If V is normally distributed with zero mean and a standard deviation of 50 mV, find $P(V <$ 40 mV). Repeat for $P(V > 10$ mV). Repeat for $P(-10$ mV $< V < 40$ mV).

7.2. If V is normally distributed with mean 10 mV and standard deviation 50 mV, find $P(V <$ 40 mV). Repeat for $P(V > 10$ mV). Repeat for $P(-10$ mV $< V < 40$ mV).

7.3. If V is normally distributed with zero mean and standard deviation 200 mV, what is the value of ΔV such that $P(V < \Delta V) = 0.6$?

7.4. An ADC input is set to 3.340 V DC. The noise of the ADC and DC signal source is characterized to be 15 mV RMS and is assumed to be perfectly Gaussian. The transition between code 324 and 325 occurs at 3.350 V DC for this particular ADC; therefore the value 324 is the expected output from the ADC. What is the probability that the ADC will produce code 325 instead of 324? If we collected 400 samples from the output of the ADC, how many would we expect to be code 324 and how many would be code 325?

7.5. An ADC input is set to 1.000 V DC. The transition between code 65 and 66 occurs at 1.025 V DC for this particular ADC. If 200 samples of the ADC output are collected and 176

of them are code 65 and the remaining code 66, what is the RMS value of the noise at the input of this particular ADC?

7.6. An ADC input is set to 2.000 V DC. The noise of the ADC and DC signal source is characterized to be 10 mV RMS and is assumed to be perfectly Gaussian. The transition between code 115 and 116 occurs at 1.990 V DC and the transition between code 116 and 117 occurs at 2.005 V DC for this particular ADC. If 500 samples of the ADC output are collected, how many do we expect to be code 115, code 116, and code 117?

7.7. A linear histogram test was performed on an unsigned binary 3-bit ADC, resulting in the following distribution of code hits beginning with code 0:

5, 6, 4, 6, 7, 7, 5, 6

A binary search was performed on the first transition between codes 0 and 1 and found the code edge to be at 10 mV. A second binary search was performed and found the code edge between codes 6 and 7 to be 1.25 V. What is the average LSB size for this 3-bit ADC? Determine the width of each code, in volts. Also, determine the location of the code edges. Plot the transfer curve for this ADC.

7.8. A linear histogram test was performed on a two's complementary 4-bit ADC resulting in the following distribution of code hits beginning with code −8:

12, 15, 13, 12, 10, 12, 12, 14, 14, 13, 15, 19, 16, 14, 20, 19

A binary search was performed on the first transition between codes −8 and −7 and found the code edge to be at 75 mV. A second binary search was performed and found the code edge between codes 6 and 7 to be 4.56 V. What is the average LSB size for this 4-bit ADC? Determine the width of each code, in volts. Also, determine the location of the code edges. Plot the transfer curve for this ADC.

7.9. A linear histogram test was performed on an unsigned binary 3-bit ADC, resulting in the following distribution of code hits beginning with code 0:

6, 6, 5, 6, 4, 6, 5, 6

A binary search was performed on the first transition between codes 0 and 1 and found the code edge to be at 32 mV. A second binary search was performed and found the code edge between codes 6 and 7 to be 3.125 V. What is the average LSB size for this 3-bit ADC? What is the measurement accuracy of this test, in volts?

7.10. A 12-bit ADC operates with a sampling frequency of 25 MHz. If a linear histogram test is to be conducted on this ADC, what should be the minimum duration of the ramp signal so that the average code count is at least 6 hits? What about for 100 hits?

7.11. A 10-bit ADC with a 10-V full-scale range operates with a sampling frequency of 60 MHz. If a linear histogram test is to be conducted on this ADC with a ramp signal of 100-μs duration, estimate the voltage resolution of this test. How many samples need to be collected for this test?

7.12. A sinusoidal histogram test was performed on an unsigned binary 4-bit ADC, resulting in the following distribution of code hits beginning with code 0:

137, 81, 60, 52, 47, 44, 43, 42, 42, 42, 44, 46, 50, 57, 72, 166

A binary search was performed on the first transition between codes 0 and 1 and found the code edge to be at 14 mV. A second binary search was performed and found the code edge between codes 14 and 15 to be 0.95 V. What is the average LSB size for this 4-bit ADC? Determine the width of each code, in volts. Also, determine the location of the code edges. Plot the transfer curve for this ADC.

7.13. A sinusoidal histogram test was performed on an two's complementary binary 4-bit ADC resulting in the following distribution of code hits beginning with code −8:

251, 163, 104, 118, 80, 99, 71, 93, 70, 94, 72, 101, 82, 124, 163, 315

A binary search was performed on the first transition between codes −8 and −7 and found the code edge to be at 20 mV. A second binary search was performed and found the code edge between codes 6 and 7 to be 9.94 V. What is the average LSB size for this 4-bit ADC? Determine the width of each code, in volts. Also, determine the location of the code edges. Plot the transfer curve for this ADC.

7.14. A sinusoidal histogram test was performed on an unsigned binary 4-bit ADC, resulting in the following distribution of code hits beginning with code 0:

137, 81, 60, 52, 47, 44, 43, 42, 42, 42, 44, 46, 50, 57, 72, 166

A binary search was performed on the first transition between codes 0 and 1 and found the code edge to be at 14 mV. A second binary search was performed and found the code edge between codes 14 and 15 to be 0.95 V. What was the input DC bias level relative to $V_{SS} = 0$ V used to perform this test? What is the minimum and maximum input signal level applied to the ADC relative to VSS?

7.15. A sinusoidal histogram test was performed on an two's complementary binary 4-bit ADC resulting in the following distribution of code hits beginning with code −8:

251, 163, 104, 118, 80, 99, 71, 93, 70, 94, 72, 101, 82, 124, 163, 315

A binary search was performed on the first transition between codes −8 and −7 and found the code edge to be at 20 mV. A second binary search was performed and found the code edge between codes 6 and 7 to be 9.94 V. What was the input DC bias level relative to $V_{SS} = 0$ V used to perform this test? What is the minimum and maximum input signal level applied to the ADC relative to V_{SS}?

7.16. A 12-bit ADC operates with a sampling frequency of 25 MHz. If a sinusoidal histogram test is to be conducted on this ADC, what should be the maximum frequency of the input signal so that the average minimum code count is at least 10 hits? 100 hits? How many samples will be collected? How long will it take to collect these samples?

7.17. A 10-bit 5-V ADC operates with a sampling frequency of 25 MHz. If a sinusoidal histogram test is to be conducted on this ADC with a test time of no more than 15 ms, estimate the worst-case voltage resolution for this test. How many samples need to be collected for this test?

7.18. A linear histogram test was performed on a two's complementary 4-bit ADC, resulting in the following distribution of code hits beginning with code −8:

20, 15, 14, 12, 11, 12, 12, 14, 14, 13, 15, 16, 16, 14, 20, 23

Determine the endpoint DNL and INL curves for this ADC. Compare these results to those obtained with a best-fit reference line.

7.19. Determine the endpoint DNL and INL curves for the histogram data provided in Problem 7.8. Compare these results to those obtained with a best-fit reference line.

7.20. Determine the endpoint DNL and INL curves for the histogram data provided in Problem 7.9. Compare these results to those obtained with a best-fit reference line.

REFERENCES

1. M. J. Kiemele, S. R. Schmidt, and R. J. Berdine, *Basic Statistics, Tools for Continuous Improvement*, 4th edition, Air Academy Press, Colorado Springs, CO, pp. 9–71, 1997, ISBN 1880156067.
2. M. Mahoney, *Tutorial DSP-Based Testing of Analog and Mixed-Signal Circuits*, The Computer Society of the IEEE, Washington, D.C., p. 137, 1987, ISBN 0818607858.

3. Reference 2, pp. 147–154.
4. J. C. Candy and G. C. Temes, *Oversampling Delta-Sigma Data Converters: Theory, Design, and Simulation*, IEEE Press, New York, 1992, ISBN 0879422858.
5. S. R. Norsworthy, R. Schreier and G. C. Temes, *Delta-Sigma Data Converters: Theory, Design, and Simulation*, IEEE Press, New York, 1996, ISBN 0780310454.