Hideo Okawara's Mixed Signal Lecture Series

DSP-Based Testing – Fundamentals 17 Another Method in S/N Calculation

Verigy Japan

September 2009

Preface to the Series

ADC and DAC are the most typical mixed signal devices. In mixed signal testing, analog stimulus signal is generated by an arbitrary waveform generator (AWG) which employs a D/A converter inside, and analog signal is measured by a digitizer or a sampler which employs an A/D converter inside. The stimulus signal is created with mathematical method, and the measured signal is processed with mathematical method, extracting various parameters. It is based on digital signal processing (DSP) so that our test methodologies are often called DSP-based testing.

Test/application engineers in the mixed signal field should have thorough knowledge about DSP-based testing. FFT (Fast Fourier Transform) is the most powerful tool here. This corner will deliver a series of fundamental knowledge of DSP-based testing, especially FFT and its related topics. It will help test/application engineers comprehend what the DSP-based testing is and assorted techniques.

Editor's Note

For other articles in this series, please visit the Verigy web site at www.verigy.com/go/gosemi.

Another Method in S/N Calculation

S/N or signal to noise ratio is one of the most typical parameter in analog and mixed signal tests. When calculating S/N, the captured waveform is processed by an FFT based API and the frequency spectrum is analyzed. For instance, DSP_THD(), DSP_SND() or DSP_SPECTRUM() are mostly employed. For successful S/N calculation, the coherency condition is indispensable. It is described as follows;

$$\frac{Ft}{Fs} = \frac{M}{N} \qquad (1)$$

Ft:	test signal frequency
Fs:	sampling frequency
M:	number of test signal cycles (an integer number)
N:	number of data points (an integer number)

M and N have no common divisor.

When calculating S/N, there are two inconvenient cases.

Case 1: The coherency condition is strictly satisfied but the number of points N is not 2^n so that FFT cannot be applied. You have to calculate entire spectrum by using DFT that is time-consuming. Test time is the most important factor in production test.

Case 2: N is 2^n, but the coherency is not strictly satisfied for some reason so that the spectrum derived by FFT is too smeared to calculate S/N correctly.

How do you cope with these situations? This is the theme of this article.
S/N is the signal to noise ratio. In general, it is calculated in the frequency domain as mentioned above. The point of this month issue is that you can calculate S/N in the time domain. So the strategy is to separate the signal from the noise. Estimation of the fundamental signal is the key.

Case 1: When N is not 2^n

The yellow line in Figure 1(a) shows a noisy waveform. Exactly M=17 cycles of a sinusoidal waveform is captured in N=379 points of a data array. The reshuffled one cycle sinusoidal waveform is overlaid in the figure as well. M and N have no common divisors so that the coherency condition is strictly complied. However, N is not 2^n so that you cannot apply FFT to calculate the SNR of the data. Then you could employ your own DFT (Discrete Fourier Transform) routine to figure out entire spectrum components. However, full spectrum DFT is much slower than FFT so that the DFT approach is not appropriate for a production program.

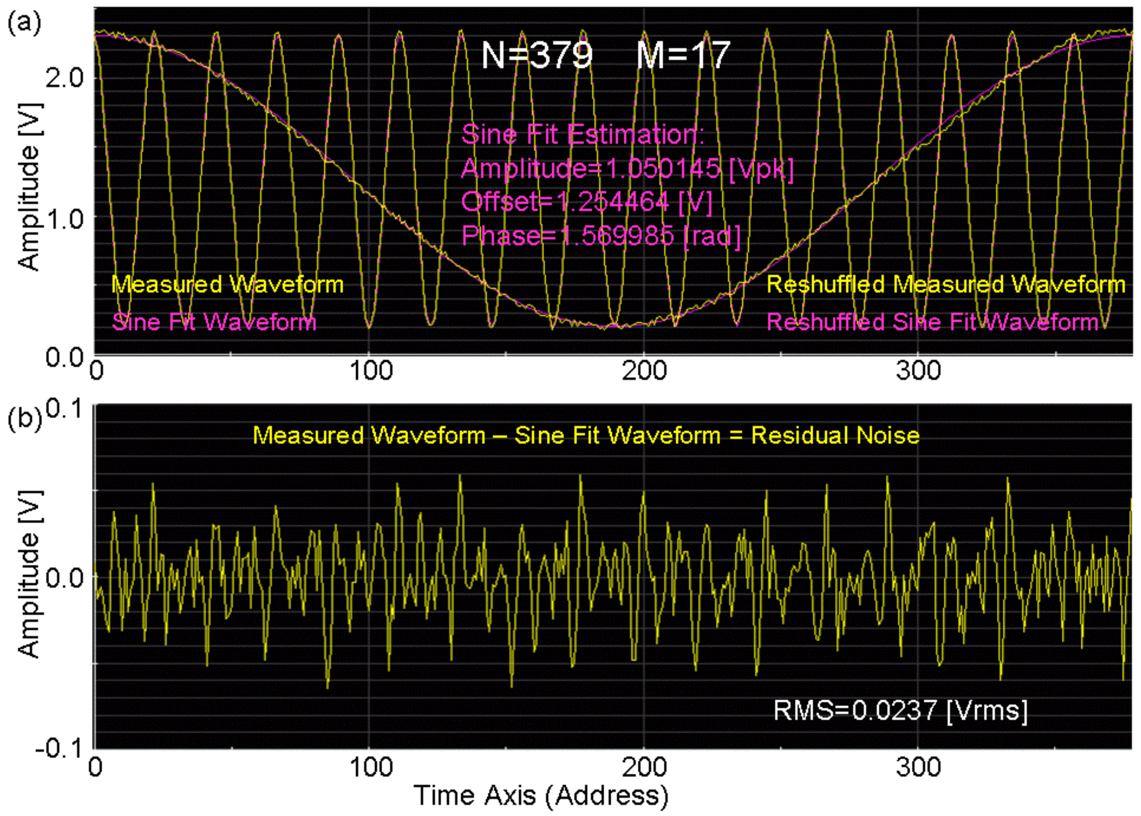

Figure 1: Measured and Sine Fit Waveforms and Residual Noise

In this case, you can utilize the sine fit “DSP_SINFIT()” to estimate the fundamental tone of the waveform. This API provides the best fit sine waveform array, amplitude, phase and offset. So you can subtract the ideal waveform from the original noisy waveform, extracting the residual noise. Figure 1(b) shows the residual noise calculated by this method. Now that you have the information of the fundamental waveform and the residual noise, you can easily calculate the SNR by computing the RMS values of them. As Figure 1 shows the signal level is estimated as 1.050145Vpk, so the RMS signal level is calculated as 0.74256465Vrms. On the other hand, the RMS residual noise is calculated as 0.0237Vrms by using an API “DSP_MEAN()”. Then the SNR is calculated as 29.9dB.

```
INT       Ndata,Ncycles;
DOUBLE    dA,dP,dO,dMean,dSigma,dRMSnoise,dRMSsignal,dSNR;
ARRAY_D   dVdgt,dWave,dNoise;
. . .

dVdgt=DGT(“AIN”).getWaveform();
. . .

Ndata=379;                 // Number of data
Ncycles=17;                // Number of cycles
dVdgt.resize(Ndata);       // 17 cycles of sine in 379 points data

DSP_SINFIT(dVdgt,dWave,&dA,&dP,&dO,Ncycles);   // Sine Fit API
DSP_SUB_VEC(dVdgt,dWave,dNoise);            // Ideal Sine Subtraction
DSP_MEAN(dNoise,&dMean,&dSigma);            // Std.Dev. Of Noise
dRMSnoise=sqrt((dSigma*dSigma)*(Ndata-1)/Ndata);
dRMSsignal=dA/sqrt(2.0);                       // Peak to RMS Conv.
dSNR=20.0*log10(dRMSsignal/dRMSnoise);         // SNR
```

List 1: SNR Calculation in Time Domain

List 1 describes the precise calculation process. Standard deviation is the same as RMS (Root Mean Square). But there is a very minute difference in definition between the statistics and the measurement. Line 16 is the correction of this difference. It is very small actually so that you may neglect it.

In order to validate the result, the waveform is processed by a full spectrum DFT discussed in the previous Newsletter article[1], and its spectrum is shown in Figure 2. By calculating the power of each line, the SNR becomes 29.9dB as well. So the time domain analysis and the frequency domain analysis derive the consistent results.

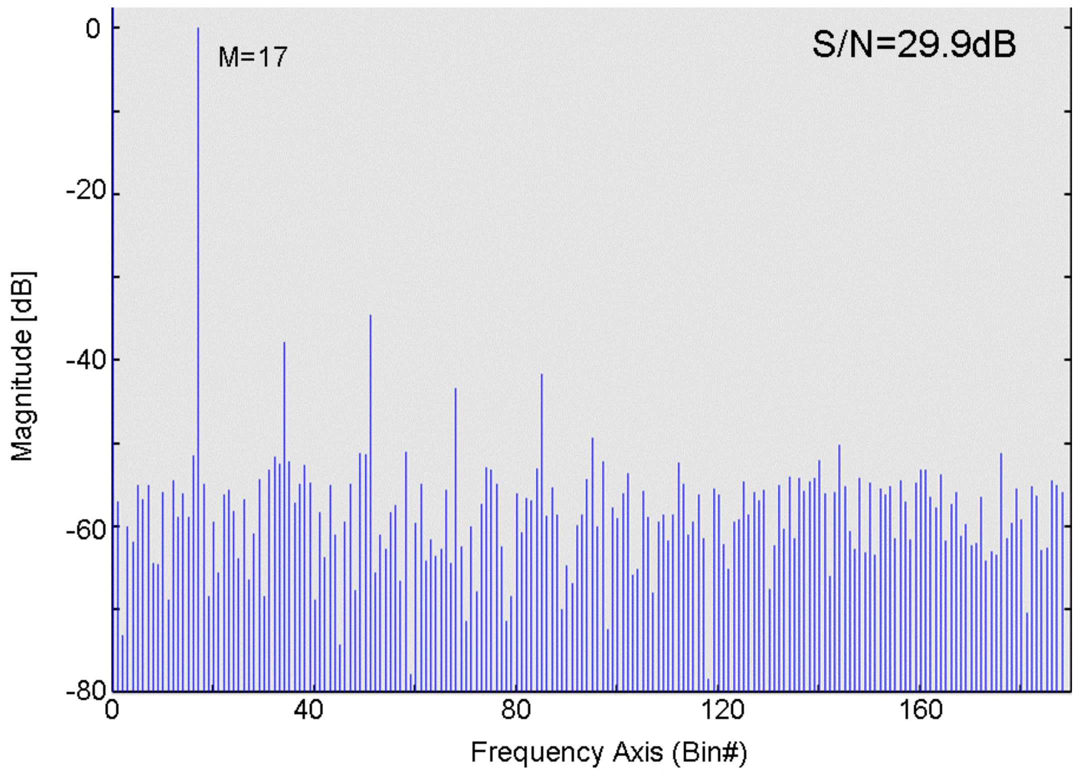

Figure 2: Spectrum by DFT (for Comparison)

Case 2: Unexpected Measurement Result

A 16-bit high-speed A/D converter is tested with a VCO based clock source on the DUT board. The VCO reference is driven by a tester digital pin. The coherency condition is strictly planned. Ft=60.00125MHz, Fs=40.96Msps, M=48001, and N=32768. Figure 3 shows the raw and re-constructed waveforms. The reconstructed waveform looks fine. However, when FFT is performed to the waveform, the frequency spectrum appears as Figure 4. The fundamental tone is smeared significantly so that the calculated S/N becomes much worse than expected. Probably the number of cycles M seems to have very minute offset than 48001 for some reason.

Practical Solution

In this case, you cannot directly calculate the S/N from the spectrum. However, there is a method to go through the problem.

[1] "DSP Based Testing – Fundamentals 5 Spectrum Analysis – DFT" Sept. 2008

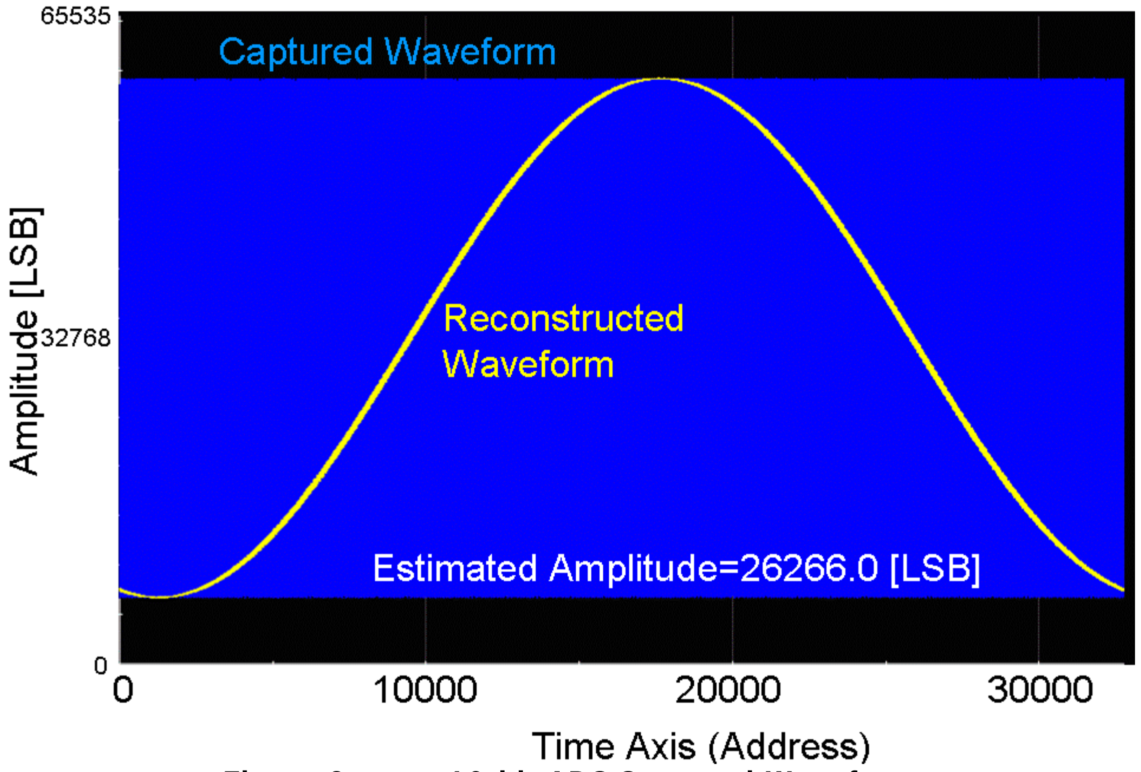

Figure 3: 16-bit ADC Captured Waveform

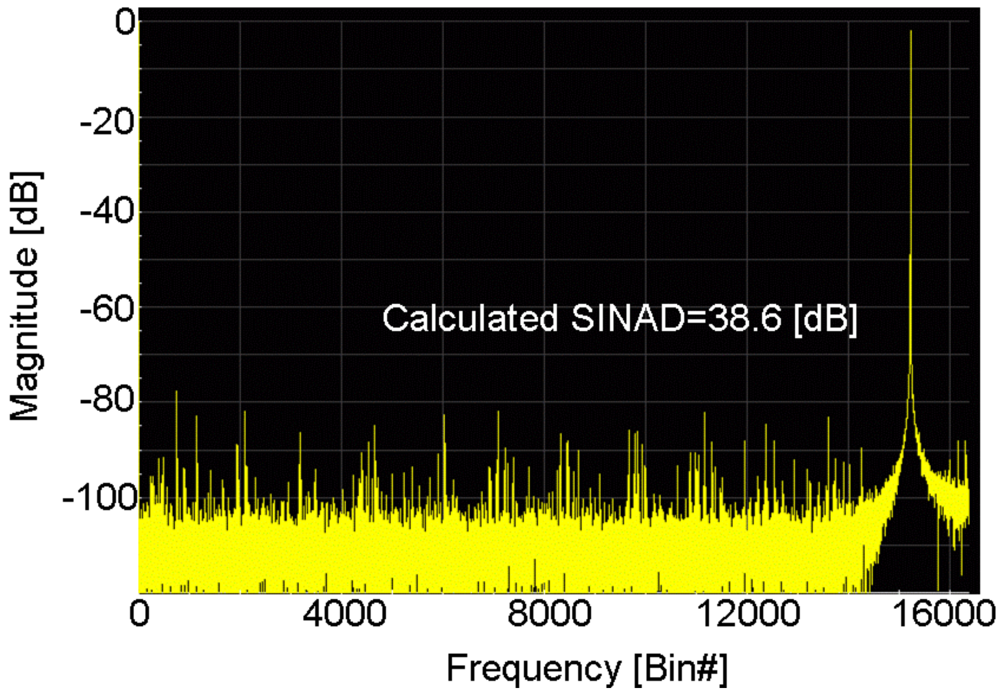

Figure 4: FFT Spectrum

If you could estimate the ideal fundamental tone by data processing, you can subtract the estimated ideal tone from the original waveform, extracting residual noise, and the S/N can be calculated as discussed in the previous section.

In the past Newsletter articles, you can find "Spectrum Estimation" [2] which describes about the Tabei & Ueda Method. By using this method, you can precisely estimate the frequency, amplitude and phase of the fundamental tone. After that, the procedure is exactly the same as in the case 1.

The example source code is shown in List 2. Lines from 14 to 30 are the application of the Tabei & Ueda Methdod for the fundamental tone estimation. Line 30 and later are the same procedure as the case of the sine fit.

```
01:   INT            i,Nadc,Ncycles,Nsp;
02:   DOUBLE         dA,dP,dR,dFx,dAx,dPx,dX,dY,dMean,dSigma,dRMS;
03:   COMPLEX        CX,CY,CZ;
04:   ARRAY_I        iADC;
05:   ARRAY_D        dADC,dNoise,dTemp,dSp;
06:   ARRAY_COMPLEX  CSp;
07:   . . .
08:   iADC=VECTOR("ADCdata").getVectors();               // ADC Data Upload
09:   Nadc=iADC.size();  Nsp=Nadc/2;
10:   . . .
11:   dA=32767.5;                                        // Full-scale of the ADC
12:   DSP_FFT(iADC,CSp,HANNING);                         // Hanning Window
13:   DSP_RECT_POL(CSp,dSp,dTemp);
14:   Ncycles=14000; dMax=0.0;                           // 14000 by the spectrum
15:   for (i=14000;i<Nsp;i++) if (dSp[i]>dMax) { dMax=dSp[i]; Ncycles=i; }
16:                                                      // dSp[Ncycles] is max.
17:   if (dSp[Ncycles-1]>dSp[Ncycles+1]) {               // Which is the 2nd Highest
18:      dR=dSp[Ncycles-1]/dSp[Ncycles];
19:      dFx=Ncycles+(1.0-2.0*dR)/(1.0+dR);              // Estimated Frequency
20:   } else {
21:      dR=dSp[Ncycles+1]/dSp[Ncycles];
22:      dFx=Ncycles-(1.0-2.0*dR)/(1.0+dR);              // Estimated Frequency
23:   }
24:   dX=dFx-Ncycles;
25:   dY=M_PI*dX;
26:   dAx= -dSp[Ncycles]*(dY/sin(dY))*(dX-1.0)*(dX+1.0);  // Estimated Amplitude
27:   CX=CSp[Ncycles];
28:   CY.real()= cos(dY); CY.imag()=-sin(dY);
29:   CZ=CX*CY;
30:   dPx=atan2(CZ.imag(),CZ.real());                    // Estimated Phase
31:   dP=2.0*M_PI*dFx/Nadc;
32:   dWave.resize(Nadc);
33:   for (i=0;i<Nadc;i++) dWave[i]=dAx*cos(dP*i+dPx);  // Ideal Waveform
34:   dADC.resize(Nadc);
35:   for (i=0;i<Nadc;i++) dADC[i]=(DOUBLE)iADC[i];     // Array DOUBLE
36:
37:   DSP_SUB_VEC(dADC,dWave,dNoise);                   // Subtract ideal from orig.
38:   DSP_MEAN(dNoise,&dMean,&dSigma);                  // Sigma Of Residual Noise
39:   dRMS=sqrt((dSigma*dSigma)*(Nadc-1)/Nadc);         // Compen. from Sigma to RMS
40:   dSNR=20.0*log10(dAx/sqrt(2.0)/dRMS);              // Signal to Noise [dB]
```

List 2: Source Code for S/N Calculation

By applying the source code in List 2, the residual noise is extracted as Figure 5 shows. Then the RMS noise is calculated as 13.96[LSB]. The estimated amplitude is 26266[LSB]. (See Figure 3)

[2] "Hideo Okawara's Mixed Signal Lecture Series DSP-Based Testing – Fundamentals 12 Spectrum Estimation" , pp.7-9, April 2009.

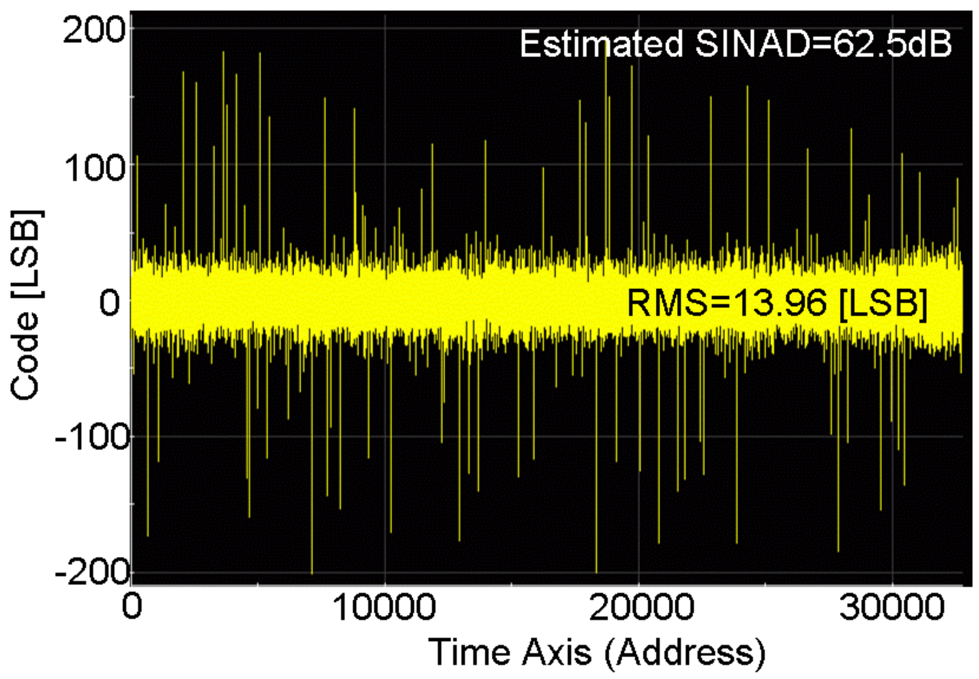

Figure 5: Residual Noise Waveform

Finally, the S/N is calculated as 62.5dB as Figure 6 where the residual noise spectrum is displayed, and the original smeared spectrum is overlaid as well.

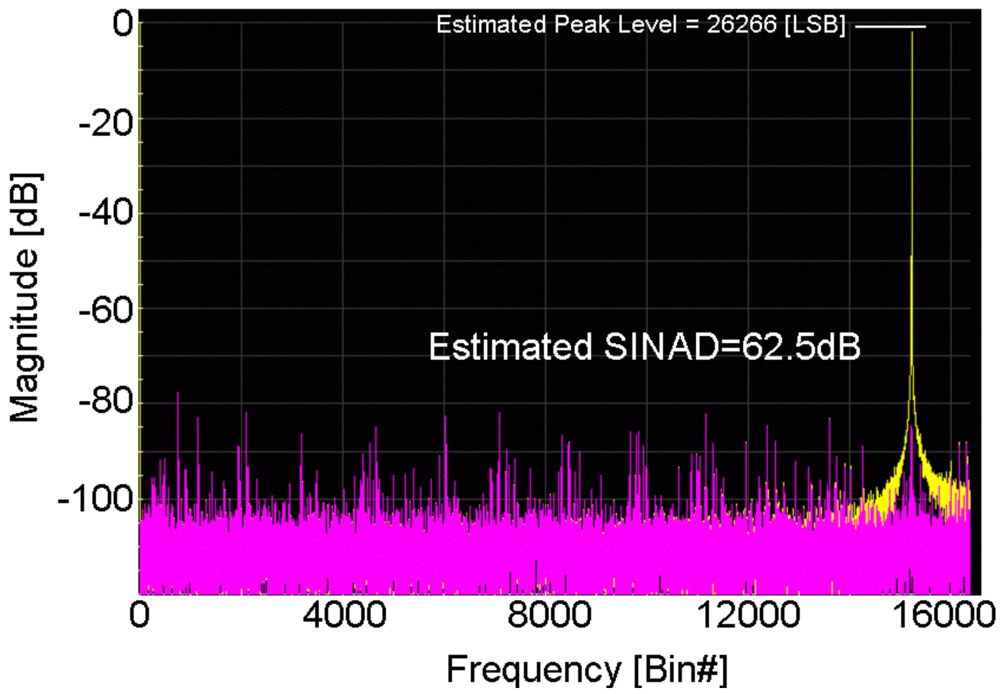

Figure 6: Composite Spectrum

Cause of the Problem

The test condition is strictly set up on the coherency condition. However, the measurement result is significantly smeared. The problem comes from less wait time in the measurement. A VCO feeds the sampling clock to the ADC. This VCO is a quartz crystal oscillator so that it takes time to settle the frequency stable when the pin driver activates it from the cold state. If the data capturing starts after enough wait time such as 1 second, you can get a beautiful spectrum.

However, it is not acceptable for a production test to wait for such a long time. You need to start as early as possible. In the situation like this, the practical solution would be useful.