TESTING AND CHARACTERIZING ADCs

GEORGIA THEANO PAPADAKIS

Cern Summer Student Program 2010
Department:BE
Group:CO

CERN-EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
1211 GENEVA-SWITZERLAND

September 3, 2010

This document's purpose is to present a precise method for testing AD Converters,to present some of the most important non linearities and noise and distortion-related specification and to highlight all the points that need attention. Also,a Python code is added at the end of the document that measures important quantities related to ADCs.

1.Basic Parameters

An ADC is a device that converts an analog signal into a digital one. There are three basic parameters and some extra information that need to be presented here. First of all,the transfer function of an ideal ADC is shown in the figure below;

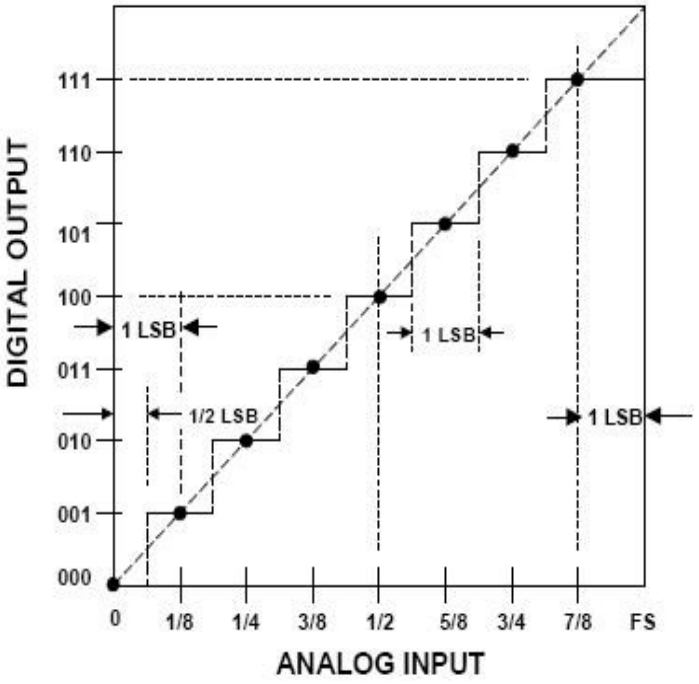

FIG.1-The ideal transfer function of a bipolar ADC

The three basic parameters are;

- The number of bits M. An ADC with M number of bits has 2^M digital values to represent an analog signal. So,in the figure above,the analog - input axes contains 2^M set of values and the digital - output axes contain 2^M values.
- FSR stands for full-scale range and is the range of analog values that the device can represent.
- LSB stands for least significant bit and is the minimum change in voltage required to guarantee a change in the output code level and is given by the formula:

$$LSB=\frac{FSR}{2^M} \tag{1}$$

2.Testing ADCs

Testing ADCs is a procedure that should be divided in two parts;Static Testing and Dynamic Testing. These are presented below.

2.1.Static Testing

The static testing procedure's purpose is to measure the non-linearities of the transfer function of an ADC. It contains the measurements of two basic parameters;DNL and INL.

2.1.1. DNL

- DNL (differential non linearity)error is defined as the difference between an actual step width of the transfer function of the ADC and the ideal value of 1LSB.For an ideal ADC, in which the differential nonlinearity coincides with DNL = 0LSB, each analog step equals 1LSB and the transition values are spaced exactly 1LSB apart.

A DNL error specification of less than or equal to 1LSB guarantees a monotonic transfer function with no missing codes. The theoretical formula of measuring DNL is shown below,and it refers to every step of the transfer function of the ADC.

$$DNL_j = \frac{V_{j+1} - V_j}{V_{LSB}} - 1 \qquad \text{where } 0 < j < 2^N - 2 \qquad (2)$$

The meaning of DNL error can be shown below,in the figure below.

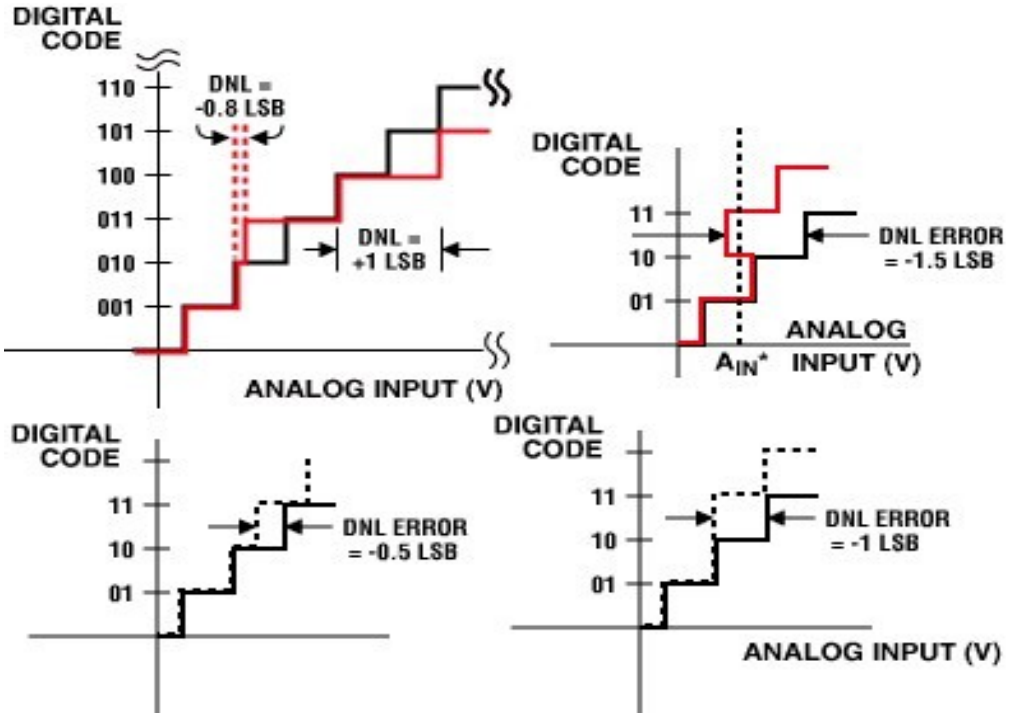

FIG.2- DNL error

2.1.2. INL

- INL(integral non linearity) is described as the deviation, in LSB or percent of full-scale range, of an actual transfer function from a straight line. This can be shown in the following figure.

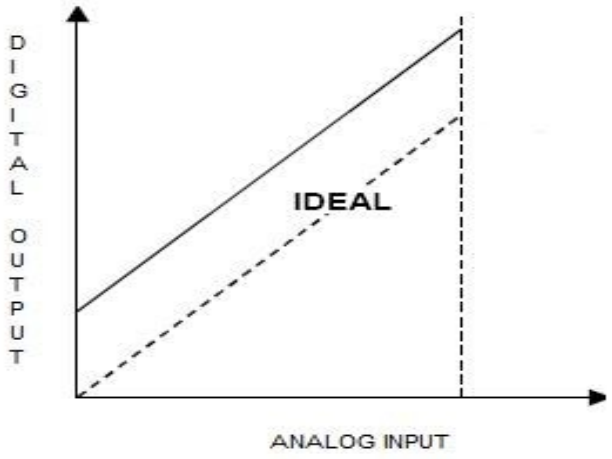

FIG.3-The deviation of the transfer function

The INL - error magnitude depends directly on the position chosen for this straight line. The theoretical formula of measuring INL is shown below.

$$INL = \left| \left(\frac{V_D - V_{ZERO}}{V_{LSB-IDEAL}} - D \right) \right| \quad \text{, where } 0 < D < 2^N - 1 \qquad (3)$$

2.2.Dynamic Testing

In order to test the dynamic performance of the ADCs,there are specific noise and distortion-related parameters that need to be measured. These parameters are:

2.2.1.SNR

- SNR (signal to noise ratio) shows how much a signal has been corrupted by noise. For an ADC with N-number of bits as a resolution,the theoretical formula of the SNR is shown below .

$$SNR_{THEORETICAL} = 6.02*N+1.76 \text{ (dB)} \qquad (4)$$

and the relation between the signal's power and the noise is shown by the following formula :

$$SNR = S\text{-}NOISE\ FLOOR - 10*\log\{(M/2\} \quad \text{(dB)} \qquad (5)$$

where M is the number of samples and the factor 10*log{(M/2} represents the process gain. fs is the sampling frequency.

2.2.2. THD

- Another useful,distortion related specification that must be measured is the THD (total harmonic distortion).THD is defined as the ratio of the signal to the root-sum-square (rss) of a specified number harmonics of the fundamental signal. IEEE Std. 1241-2000 (Reference 9) suggests that the first 10 harmonics be included. A THD rating < 1% is desired. The way that THD is going to measured is by the following formula:

$$THD = 20\log\left(\sqrt{10^{[V_2/20]2} + 10^{[V_3/20]2} + .. + 10^{[V_{10}/20]2}}\right) = 10\log\left(10^{[V_2/20]2} + 10^{[V_3/20]2} + ... + 10^{[V_n/20]2}\right) \quad (6)$$

where V2 ,V3,...V10 are expressed in dB.

By having THD and SNR values defined,then there are two more specifications that need to be measured.

2.2.3. SINAD

- SINAD stands for Signal-to-noise and distortion ratio. It is a measure of the quality of a signal from a communications device, defined as:

$$SINAD = \frac{P_{signal} + P_{noise} + P_{distortion}}{P_{noise} + P_{distortion}} \quad (7)$$

2.2.4. ENOB

- ENOB is the last parameter that should be calculated. It stands for effective number of bits and it represents the quality of a digitized signal. The resolution of an ADC is specified by the number of bits used to represent the analog value, in principle giving 2^N signal levels for a N bit signal. However, all real signals contain a certain amount of noise. If the converter is able to represent signal levels below the system noise floor, the lower bits of the digitized signal only represent system noise and do not contain useful information. ENOB specifies the number of bits in the digitized signal above the noise floor. An often used definition for ENOB is:

$$ENOB = \frac{SINAD - 1.76 + factor}{6.02} \quad (8)$$

where “factors “ is :

$$factor = 20\log\left(\frac{full\ scale\ amplitude}{real\ amplitude}\right) \quad (9)$$

and is a correction factor that normalizes the ENOB value to full-scale range,regardless of the actual signal amplitude. Full scale amplitude is half the FSR for a bipolar ADC.

3.The Actual Method

A practical way to test an ADC ,in order to calculate all the mentioned quantities, is by supplying it with a full scale amplitude sine wave input, and use the digitized samples of it (that have values from 0 to 2^M-2 for a bipolar ADC) to measure non-linearities and specific noise and distortion quantities. These digitized samples are going to be used both for static and dynamic testing.

Two very important notes about the sampling procedure are the following;

Firstly,the number of samples should always be a power of two. Also,it should always guarantee an integer number of sine wave periods. In addition, the sine wave' s frequency and the sampling frequency should not be correlated in any way.

3.1.The Actual Method for Static Testing : The Histogram Testing

This specific method for calculating DNL and INL involves collecting a very large number of digitized samples over a period of time, for a well-defined input signal(sine wave in this case) with a known probability density function .The ADC transfer function is then determined by a statistical analysis of the samples . The large number of digitized samples are collected and the number of occurrences of each code are tallied. It is really important to get a very large number of samples:much larger than than the number of possible digital values that the ADC can represent(2^M-2 for a bipolar ADC). There is a suggesting formula for the number of samples:

$$M_T = \pi \times 2^{N-1} \times (Z_{\alpha/2})^2 / \beta^2 \quad (10)$$

As it can be seen,M_T depends on the resolution of the ADC (N is the number of bits), the desired confidence level of the measurement, and the size of the DNL error.

The confidence level is the probability value (1-α) associated with a confidence interval. The confidence level is usually expressed in percent. The confidence level values used in the M_T calculation originate from Table 2 of the IEEE Standard 1241 Draft, printed on May 12th, 1997 - Standards for Terminology and Test Methods for Analog-to-Digital Converters Prepared by the Analog-to-Digital Converter

Subcommittee of the Waveform Measurements and Analysis Committee of the IEEE Instrumentation and Measurement Society. That table is shown below. Also,β depends on the DNL error that the user is expecting to get,expressed as a fraction of the LSB.

u	$Z_{u/2}$	$Z_{4,u/2}$	$Z_{8,u/2}$	$Z_{12,u/2}$	$Z_{16,u/2}$	$Z_{20,u/2}$	$Z_{24,u/2}$
0.2	1.28	2.46	3.33	4.04	4.64	5.19	5.68
0.1	1.64	2.72	3.53	4.21	4.80	5.33	5.81
0.05	1.96	2.95	3.72	4.37	4.94	5.46	5.93
0.02	2.33	3.22	3.95	4.57	5.12	5.62	6.08
0.01	2.58	3.42	4.11	4.71	5.25	5.74	6.19
0.005	2.81	3.60	4.27	4.85	5.38	5.85	6.30
0.002	3.09	3.84	4.47	5.03	5.54	6.01	6.44
0.001	3.29	4.00	4.62	5.16	5.66	6.12	6.54

(Table 2 of the IEEE Standard 1241)

For instance, if α = 1% then the confidence level is equal to 1 - 0.01 = 0.99 or 99% confidence level. So, a 10-bit ADC with a DNL error (β) of 0.1LSB and a 99% confidence level, which means that α=0.005 ,requires at least M_T=1270100 samples to be recorded.

The frequency of the waveform should be low enough such that the ADC does not make ac-related errors and it must not be an exact divisor of the sampling frequency. This is very important in order not to have clustering of values caused by the sub-harmonics of the sine wave.

The probability density function of the sine wave,for an N-bit resolution ADC,can be calculated by the following formula,

$$p(n)=\frac{1}{\pi}[\sin^{-1}\frac{2V_{FS}(n-2^{N-1})}{A2^N}-\sin^{-1}\frac{2V_{FS}(n-1-2^{N-1})}{A2^N}] \quad (11)$$

where V_{FS} is the analog range of half FSR and A is the estimated amplitude of the sine wave input. The plot of p(n) is shown in the figure below;

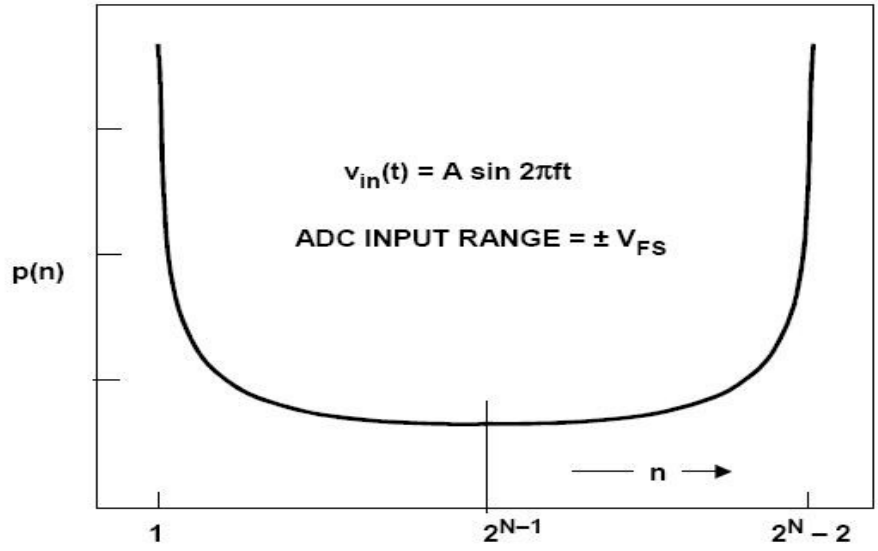

FIG 4-probability density function of a sine wave

A is given by the following formula:

$$A_{ESTIMATE} = \frac{V_{FS}}{\sin_{-1} \frac{M_T}{M_T + h(0) + h(2^{(N)} - 1)} \frac{\pi}{2}} \qquad (12)$$

The actual amplitude of the sine wave input should be chosen such that the ADC is slightly overdriven at both ends of its range.

If h(n)ACTUAL is the number of occurrences per bin,then the practical way to measure DNL is by the formula:

$$DNL(n) = \frac{h(n)_{ACTUAL}}{p(n) M_T} \qquad (13)$$

It is important for one to notice that V_{FS} and V_{FSR} (or FSR) are not the same quantities. For a bipolar ADC,FSR is the full scale range that equals to 2 V_{FS}.

Once all the DNL values are calculated(one for each step of the transfer function),the INL can be estimated by the following formula:

$$INL_j = \sum_{n=1}^{j} DNL_j \quad , \text{ from n=1 to j} \qquad (14)$$

or by detecting the largest value of DNL error that represents the maximum deviation, in LSB or percent of full-scale range, of an actual transfer function from a straight line.

A small summary of the method is:

- Calculating p(n) from eq. (11)
- Making the histogram by counting the number of occurrences per bin (h(n))
- Calculating DNL(n) by using eq.(13)
- Calculating INL(n) by detecting the largest value of DNL error.

It should be mentioned that these procedure could be used by using any other signal as an input,which means that it isn't mandatory to use the sine wave,but any other function with a known propability density function could be used.The reasons why the sine wave is used are more related to the dynamic testing of ADC;the sine wave has a unique property;it's fourier transform is a dirac function that can be very easily distinguished.This will be further understood in the following pages.

3.2.The Actual Method for Dynamic Testing

This specific method for the dynamic testing requires the frequency of the input sine wave signal not to be a sub-harmonic of the sampling frequency. Also the number of samples should indulge the condition:

$$fin/\ fs = Mc\ /M \qquad (15)$$

where fin is the frequency of the sine wave, fs is the sampling frequency,Mc is the number of cycles in the spectral window and M is the total number of samples .It should

be noted that Mc should be a prime number and M should be a power of two. The condition in equation (15) guaranties a Coherent Sampling and it is mandatory in order to avoid a situation where the energy of the fundamental and its harmonics leak into adjacent bins as shown in this figure:

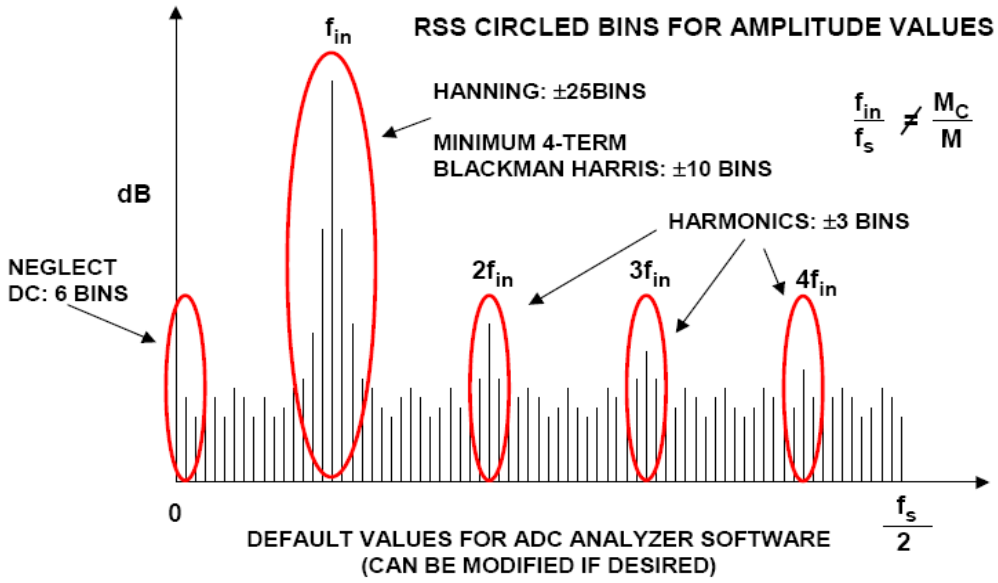

FIG.5-The FFT of the signal

It should be mentioned that in order to plot the data in a meaningful way, the magnitude data must be converted to decibels (dB). This can be done using the formula:

$$dB = 10\log\left(\frac{Magnitude^2}{Magnitude_{Max}^2}\right) = 20\log\left(\frac{Magnitude}{Magnitude_{Max}}\right) \qquad (16)$$

It should be mentioned that “Magnitude” represent values of power,not amplitude,hence the square as the multiplication by 20.

For the Dynamic-testing,it is reasonable to work in frequency domain. So the Fourier Transform of the signal should be calculated out of the digitized samples of the sine wave. The main idea for using the DFT of the sampled sine wave is coming from Parseval' s theorem; the sum (or integral) of the square of a function is equal to the sum (or integral) of the square of its transform.

But the integral of the square of a function,in this application is translated as the power of the signal. So, by measuring the rss values of all the noise and distortion bins from the FFT plot, the values of noise and distortion are obtained. By detecting the highest peak of the FFT plot,the power of the output signal(sine wave) is obtained.

An important note is that,for the plot of the FFT to be more clear,a window function should be used. It should be multiplied with the digitized sine wave .One could use Hamming,Triangular,Bartlett or Blackman function as a window function.

This procedure and the influence of the window function to the signal is shown in the figure below. It makes the important parts of the FFT clear,in order to discriminate the signal,noise and distortion from each other more easily.

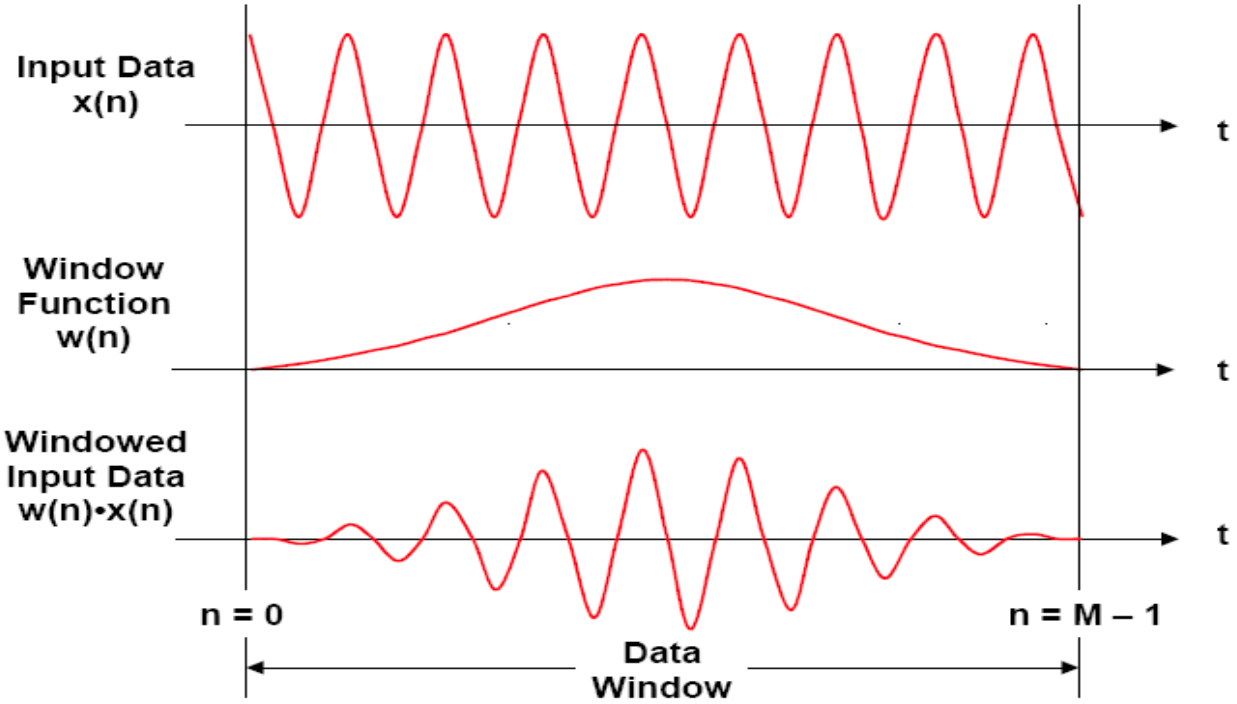

FIG.6-The use of the window function

The formula that should be used to calculate SNR ,after the mentioned procedures are integrated,is the formula (5).To calculate THD eq.(6) is used and to calculate SINAD the following equation should be used.

$$SINAD = 20\log\left(\sqrt{10^{[-SNR/20]^2} + 10^{[-THD/20]^2}}\right) \quad (17)$$

To calculate ENOB,formulas (8) and (9) should be used.

A small summary of the method is;

- Calculating the DFT of the digitized sine wave.
- Calculating the height of the input frequency's bin which stands for S in eq. (5)
- Calculating the height of all the local maxima of the plot. These are the V_2 ,..., V_{10} ,so they are the distortion components of eq.(6)
- Calculating the height of all the other components of this FFT signal and taking the rss value of those. This stands for NOISE FLOOR in eq. (5)
- Calculating the process gain from the formula: 10*log(M/2)
- Calculating SNR from eq.(5)
- Comparing that value with the theoretical value of SNR taken from eq. (4)
- CAUTION: when calculating the rss value of all the frequency-components except the sine wave' s frequency bin,the sub-harmonic frequency bins should not be included(since SNR is a noise-related specification and not a distortion specification)
- Calculating SINAD and ENOB from formulas (17),(8),(9).

4.Algorithms for Testing ADCs.

In order to correspond the described method in terms of software and programming,here are described the details of the algorithms developed for static and dynamic testing of ADCs.

4.1.Static Testing Algorithm Details

The following algorithm is simple; a vector called “samples” is used in order to append in it the sine wave values that are generated by a small routine called “makesinewaveforunipolar(n,sampl_rate,fr,ph,amp,fullsr)”or“makesinewaveforbipolar(n,sampl_rate,fr,ph,amp,fullsr)” .To digitize the samples,there is another small routine called “adc_transferunipolar(v, fsr, bits)” or “adc_transferbipolar(v, fsr, bits,vfs)”.In that routine,vector ”analog” has elements that are integer multiplications of LSB and vector “digital” contains the corresponding digital values .By appending the digitized values, that this routine produces,to a new vector,called “dig_samples”,a histogram is made in order to calculate the number of occurrences per bin,meaning the number of times that a specific digital value was observed. Another vector is also used,in which the theoretical values of the probability density function (p(n) in formula (11)) per bin are appended. p(n) should always look like the figure 4. Then, formulas (13) and (14) are used to calculate two vector of DNL and INL values. The algorithm also calculates the maximum deviation of the transfer function,which is another way to measure INL.
The algorithm described above is the following:

```
from math import *
from matplotlib import pyplot as plt
from numpy import *
#**************CREATING THE DISTORTED SINEWAVE**********************#

def make_sinewaveforunipolar (n,sampl_rate,fr,ph,amp,fullsr):
    #produces the values of the sinewave#
     b=[]
     maj=[]
     for i in range (n):
         b.append(fullsr/2.0+amp*sin(2.0*pi*fr*i/sampl_rate+ph*2.0*pi))
       #b.append(fullsr/2+amp*sin(2*pi*fr*i/sampl_rate+ph*2*pi)
    +0.2*amp*sin(4.44*pi*fr*i/sampl_rate+ph*2*pi)
    +0.09*amp*sin(6.66*pi*fr*i/sampl_rate+ph*2*pi))
     return b

def make_sinewaveforbipolar (n,sampl_rate,fr,ph,amp,fullsr):
    #produces the values of the sinewave#
    b=[]
    for i in range (n):
```

```
        b.append(amp*sin( 2.0*pi*fr*i/sampl_rate+ph*2.0*pi))
       #b.append(amp*sin(2.0*pi*fr*i/sampl_rate+ph*2.0*pi)
       +0.02*amp*sin(4.44*pi*fr*i/sampl_rate+ph*2*pi)
       +0.09*amp*sin(6.66*pi*fr*i/sampl_rate+ph*2*pi))
    return b

#**********ANALOG TO DIGITAL**********#

def adc_transferunipolar(v, fsr, bits):

    #MAKING THE TRANSFER FUNCTION FOR UNIPOLAR ADC
    a=2.0**bits
    lsb=float(fsr/a)
    digital=[]
    analog=[]
    for i in range (int(a)):
        b=lsb*i
        analog.append(b)
        digital.append(i)

    #AND NOW THE CORRESPONDENCE!!!
     i=0
    if (v>=analog[0]) and (v<=analog[int(math.pow(2,bits)-1)]):
       while i<=a-1:
           if (v>=analog[i]) and (v<analog[i+1]):
               d=(digital[i])
           i=i+1
    elif (v>analog[int(math.pow(2,bits)-1)]) :
        d=( digital[int(math.pow(2,bits)-1)])
    else:
        d=(digital[0])

    return d

   def adc_transferbipolar(v, fsr, bits,vfs):

      #MAKING THE TRANSFER FUNCTION FOR BIPOLAR ADC
      a=2.0**bits
      lsb=float(fsr/a)
      digital=[]
      analog=[]
      for i in range (int(a)):
          b=-vfs+lsb*i
```

```
        analog.append(b)
        digital.append(i)

        #AND NOW THE CORRESPONDENCE!!!
       i=0
        if (v>=analog[0]) and (v<=analog[int(math.pow(2,bits)-1)]):
           while i<=a-1:
               if (v>=analog[i]) and (v<analog[i+1]):
                   d=(digital[i])
               i=i+1
        elif (v>analog[int(math.pow(2,bits)-1)]) :
            d=( digital[int(math.pow(2,bits)-1)])
        else:
            d=(digital[0])

        return d
#************************MAKING THE HISTOGRAM************************#

def histo(bits,samples):
      samples=array(samples)
      c=[]
      for i in range (2**bits):
          c.append(0)
      i=0


      for j in range (2**bits):
          for i in range(samples.size):
              if (samples[i]==j):
                  c[j]=c[j]+1
      return c

#*******************************THE INPUTS *******************************#

b=input("give number of bits :")
fsr=input("give me fsr(volts): ")
a=input("please give amplitude(volts):")
f=input("please give the sinewave's frequency(Hertz):")
p=input("please give phase:")
sr=input("please give sampling rate(Hertz):")
s=input("please give number of samples:")
bipolar=input("if the ADC is bipolar then press 1,else press 0 :")
```

```
############################## EXECUTION ###################################
fs=fsr/2
samples=[]                                  #making the sinewave's samples
if (bipolar==0):
   samples=make_sinewaveforunipolar(s,sr,f,p,a,fsr)
elif (bipolar==1):
   samples=make_sinewaveforbipolar(s,sr,f,p,a,fsr)

samples=array(samples)
dig_samples=[]                               #the convertion
if (bipolar==0):
   for i in range (s):
      dig_samples.append(adc_transferunipolar(samples[i], fsr, b))
elif (bipolar==1):
    for i in range (s):
       dig_samples.append(adc_transferbipolar(samples[i], fsr, b, fs))
dig_samples=array(dig_samples)
#to be sure that the frequency of the sinewave and the sampling frequency are not
#correlated,i need dig_samples to have a lot of different from each other values
e=[]
e=histo(b,dig_samples)
#print ("the number of occurencies per bin are "),e[:]
print ("                                ")
e=array(e)
ala=0                       #this variable shows the number of different values
for i in range (e.size):
         if (e[i]!=0):
                 ala=ala+1
h_first=e[0]
last=e.size
h_last=e[last-1]

#****************CALCULATING THE ESTIMATED AMPLITUDE**************#
lsb=fsr/(2.0**b)
print("the lsb is :"),lsb
print ("                                ")
g=(math.pi/2)
help1=float(s+h_first+h_last)
help2=float((s/help1))
w=float((g*(help2)))
k=float(math.sin(float(w)))
amp=float(fs/k)
print("the  estimated amplitude is "),amp
print ("                                ")
```

```
#**********CALCULATING THE p(n)**********#

p=[]
twobits = 2.0**b
twobits1 = 2.0**(b-1)
pirecip  = 1.0/math.pi
for i in range(1,int(twobits)+1):
       p.append(pirecip*( math.asin(2*fs*(i  -twobits1)/(amp*twobits)) - math.asin(2*fs*(i-1-
twobits1)/(amp*twobits)) ))
p=array(p)
print ("                                ")

#**********CALCULATING h(n)theoretical**********#
h=[]
for i in range(p.size):
   h.append(p[i]*s)
#print("                                ")
h=array(h)
#print("the theoretical number of samples is :"),sum(h)
#**********CALCULATING DNL ERROR**********#
dnl=[]
help4=0.0
help5=0.0
for i in range (0,2**b):
      help4=(e[i]/h[i])
      help5=help4-1.0
      dnl.append(help5)
dnl=array(dnl)
alala=0
for i in range ((2**b)-2):
        if (dnl[i]!=-1.0):
                alala=alala+1   #this variable calculates how many different values dnl
#vector has
print ("T H E  DNL  E R R O R S   A R E  :"),dnl[:]
#**********CALCULATING INL ERROR**********#
inl=0
for i in range (2**b-2):
      inl=inl+dnl[i]
##########################################################################
rang=range(p.size)
plt.plot(rang,h,rang,e)
plt.show()
```

4.1.1.Static Testing Algorithm Results

Below,there is an execution of my Static Testing algorithm. As an input I used a sine wave with 0.314Hz frequency,5Volts amplitude and the sampling rate of the ADC is 1000Hz .

```
give number of bits :8
give me fsr(volts): 10
please give amplitude(volts):5
please give the sinewave's frequency(Hertz):0.3.1415164576

please give phase:0
please give sampling rate(Hertz):1000
please give number of samples:100000
if the ADC is bipolar then press 1,else press 0 :1
the number of occurencies per bin are  3927, 1634, 1252, 1061,
933,...,  1095,1295, 1682, 4056
the lsb is : 0.0390625
   >>> dnl
  array([  2.55229473e-01,     1.48077630e-01,     1.07063086e-01,
1.13477963e-01,   1.42539189e-01,
         1.87508411e-01])
>>The   maximum   deviation   of   the   transfer   function   is   :
0.255229473128 LSBs. Which means : 0.0996990129405 %FSR.
```

The algorithm also shows, as an output,the plot of both the ideal histogram,which is the p(n) multiplied by the number of samples M_T, and the histogram made out of the digitized samples. For the input described above,this plot is shown below.

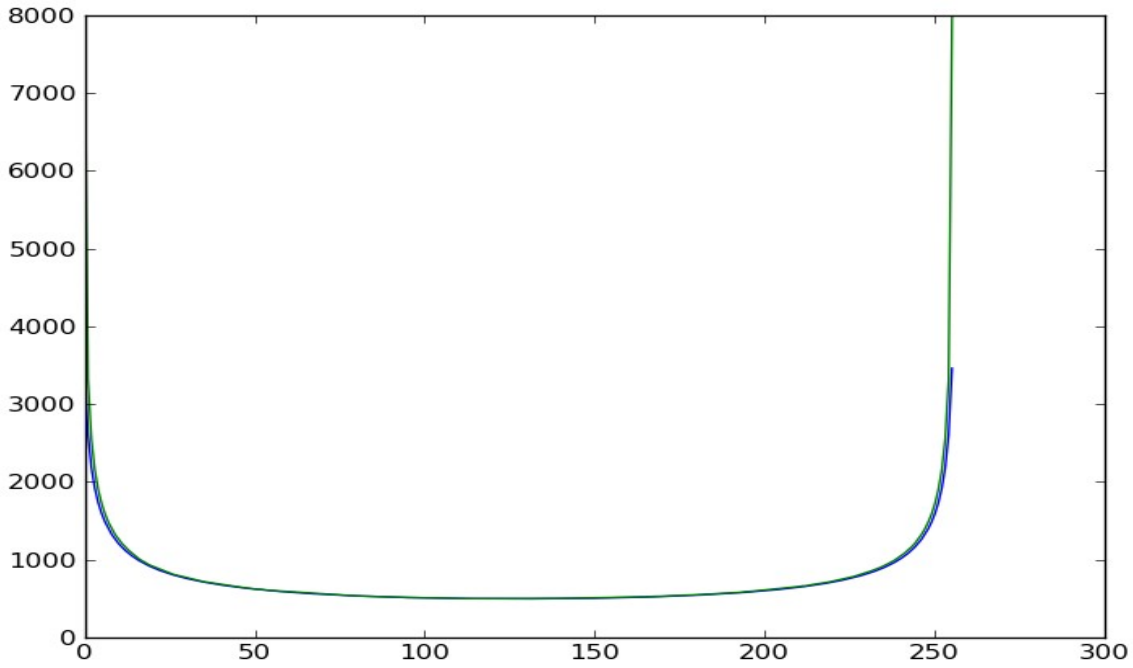

FIG.7-The theoretical and the calculated number of occurrences per bin

In this example,as it should,DNL and INL are calculated in LSBs. INL is also calculated as a percentage of FSR .The value of INL is in the correct order of magnitude and,also,the value of it is quite satisfying. A really important note is that it is to be expected to have DNL and INL errors even in the case that a pure sine wave is used as an input because,as it can be seen in the routine "adc_transferunipolar(v, fsr, bits)" or "adc_transferbipolar(v, fsr, bits,vfs)",the way the "software ADC" works is exactly the same as the way that a real ADC works where the non-linearities are caused by the inside – imperfections of the ADC. Meaning that it isn't easy to obtain all the different digital values that the ADC can represent:that would need a huge number of samples or a very high sampling frequency,but that doesn't always guarantee a finite executing time .But this is mandatory so a compromise is made. An interesting observation is that, as the sampling frequency and the number of bits increase,the DNL and,consequently,the INL errors decrease (as they should).

Note: In the routines def make_sinewaveforunipolar (n,sampl_rate,fr,ph,amp,fullsr) and def make_sinewaveforbipolar (n,sampl_rate,fr,ph,amp,fullsr) there are two commented lines that generate a distorted sine wave instead of a pure sine wave .This lines were not used in the execution of the algorithm because a pure sine wave is required for the static testing of ADCs. Those lines are there just to give the opportunity to a user to test the algorithm. It should be noted that,in order to produce a distorted sine wave,the signal added to the pure sine wave should be something like that :

signal added to the pure sine wave= a2 sin 2π2.22222 f t+ a3 sin 2π3.3333 f t
+ a4 sin 2π4.444f t+...

so that the distortion is totally random.

4.2. Dynamic Testing Algorithm Details

The algorithm used for the dynamic testing of ADCs is shown below. The algorithm contains a small loop for the opening of a file that contains the digitized sine wave samples and stores them into a vector. In order to calculate the noise and distortion-related quantities,the Fourier Transform of the digitized samples is calculated,and all the quantities are converted into dB units. Firstly,a vector is used in which the values of the digitized samples are appended. The window function used is the Bartlett function (but Hamming,Triangular or Blackmann could be also used). It is multiplied with the digitized sine wave and then the FFT of that multiplication is calculated and plotted.

The first part of the algorithm apart from the opening of the files,has to do with the detection of the harmonics and the calculation of the distortion. More specifically ,two vectors are used;one to append the results of the FFT of the multiplication of the window function and the digitized-sine wave and one to keep the corresponding frequencies. The condition used in order to recognize a value as a peak or not is double: firstly,the algorithm searches for local maxima of the FFT. Then,it recognizes them as distortion only if the corresponding to those peaks frequency values are close to a value that equals to an integer multiple of the sine wave' s frequency.”How”close depends on the number of samples. The distortion was then measured as the rss value of the magnitudes of those detected peaks. This is how THD could be calculated by using equation (6).

Following,the noise is now easy to measure: any component of the signal that is smaller than the mean value of the distortion is recognized as noise. So,the rss value of those components of the initial vector is the noise. Of course,the pure sine wave 's component is excluded. This is how SNR is being calculated by using equation (5).

The calculation of ENOB and SINAD was than easy to do by using equations (17),(8), (9).

The algorithm described above is the following:

```
from math  import *
from numpy import *
from matplotlib import pyplot as plt

#************************** OPEN THE FILE ******************************#
num_of_bits=input("GIVE ME THE NUMBER OF BITS :")
real_amplit=input("GIVE ME THE INPUT AMPLITUDE :")
```

```
full_scale_amplitude=input("GIVE ME THE FULLSCALE AMPLITUDE :")
name=raw_input("GIVE ME THE NAME OF THE FILE:")
f=open(name)
a=[]
for line in f:
    value=int(line)
    a.append(value)

#***************************MAKING THE FFT*****************************#
a=a-mean(a)
a=array(a)
asize=a.size
x=log2(asize)
print ("NUMBER OF SAMPLES ARE(-SIZE OF a IS) : "),asize
print ("THE SIZE OF a EQUALS TO:  2 TO THE "),x
b=abs(fft.fft(a))
maxb=max(b)
d=b/maxb
e=log10(d)
f=20*e
h=fft.fftshift(f)
m=h.size

#***********I 'LL MAKE A COUPLE:VECTOR h AND VECTOR freq **************#
freq=[]
for i in range (m):
    freq.append(-50000000+i*100000000/m)

#***** I'LL MAKE ANOTHER COUPLE:Vector hhalfh AND VECTOR freq_shift*****#
hhalf=[]
freq_shift=[]
for i in range(m/2,m):
    freq_shift.append(freq[i])
    hhalf.append(h[i])
hhalf=array(hhalf)
freq_shift=array(freq_shift)
ser=hhalf.size
print ("SIZE OF hhalf IS:"),ser

#***SEARCH FOR THE POSITION OF THE MAXMUM OF hhalf=>EXPECTING THE
#************************INPUT FREQUENCY*****************************#
for i in range (ser):
        maximum=hhalf[i]
        for j in range (ser):
        if hhalf[j]>maximum:
```

```
            maximum=hhalf[j]
            sinefreq=freq_shift[j]
print("the frequency of the sinewave is "),sinefreq
print("maximum of hhalf is "),maximum
#*****************************WINDOW FUNCTION************************#
nbart=asize/2
d=hamming(nbart)
print d[:]

gerog=a[0:(asize/2)]
gerog=array(gerog)
d=array(d)
mult=d*gerog
mult=array(mult)
mult1=abs(fft.fft(mult))
maxmult1=max(mult1)
mult2=mult1/maxmult1
mult3=log10(mult2)
mult4=20*mult3
mult5=fft.fftshift(mult4)
multsize=mult5.size

freq_shift1=[]
for i in range (gerog.size):
          freq_shift1.append(-50000000+i*100000000/nbart)
for i in range (gerog.size):
            if freq_shift1[i]==0:
                        flag=i
fourievector=mult5[flag:multsize]
freqvector=freq_shift1[flag:multsize]


# **********CALCULATION OF THE DISTORTION WHEN THERE IS ALSO*******
#******PROCESS GAIN SO THE HARMONICS ARE A LITTLE SPREAD**********
#**THE ARRAYS THAT ARE GOING TO BE USED ARE: fourievector and freqvector **

position=[]
distortion=[]
for i in range (fourievector.size):
#I TRY TO FIND THE POSITION OF THE HARMONICS BY SEARCHING IN freq_shift
#VECTOR FOR  FREQUENCIES CLOSE TO THE:f=n*sinefreq
        if (freqvector[i]%sinefreq==0) :#if you find a harmonic
                if x%2==0 :
```

```
    magnitude=[]
    helpfreq=[]
    if i>=2**(((x-2)/2)-1) and i<=fourievector.size-2**(((x-2)/2)-1)-1:
          for j in range (int(-2**(((x-2)/2)-1)),int(2**(((x-2)/2)-1))):
             magnitude.append(fourievector[i+j])
             helpfreq.append(freqvector[i+j])
  elif  i<2**(((x-2)/2)-1):
               for j in range (0,int(i+2**(((x-2)/2)-1))):
                  magnitude.append(fourievector[j])
                  helpfreq.append(freqvector[j])
  else:
               for j in range (int(i-2**(((x-2)/2)-1)),fourievector.size):
                  magnitude.append(fourievector[j])
                  helpfreq.append(freqvector[j])
           magnitude=array(magnitude)
           helpfreq=array(helpfreq)
           maxmag=max(magnitude)
           sizemag=magnitude.size
           for w in range(sizemag):
               if magnitude[w]>=maxmag:
                  maxmag=magnitude[w]
                  position.append(helpfreq[w])
                  distortion.append(magnitude[w])

 else:
          magnitude=[]
          helpfreq=[]
          for j in range(int(-2**(((x-3)/2)-1)),int(2**(((x-3)/2)-1))):
              magnitude.append(fourievector[i+j])
              helpfreq.append(freqvector[i+j])
          magnitude=array(magnitude)
          helpfreq=array(helpfreq)
          maxmag=max(magnitude)
          sizemag=magnitude.size
          for w in range(sizemag):
             if magnitude[w]>=maxmag:
                maxmag=magnitude[w]
                position.append(helpfreq[w])
                distortion.append(magnitude[w])
```

#I ASSUMED THAT WHEN THERE IS A SIGNAL LEACAGE,ONLY THE #HIGHEST #PEAK IS A HARMONIC AND THE REST OF THE PEAKS ARE NOTHING,NOT #EVEN NOISE

```
distortion=array(distortion)
distsize=distortion.size
position=array(position)

meandistortion=mean(distortion)
realdistortion=[]
realposition=[]
noise1=[]
for i in range(distsize):
    if (distortion[i]>=meandistortion) and (position[i]>sinefreq):


        realdistortion.append(distortion[i])
        realposition.append(position[i])

#************ C A L C U L A T I O N  O F  N O I S E ==> SNR ******************#

for i in range (ser):
    if hhalf[i]<meandistortion:
        noise1.append(hhalf[i])
noise1=array(noise1)
noisesize=noise1.size
noise2=[]
for i in range(noisesize):
    if noise1[i]>-140.0:
        noise2.append(noise1[i])

print ("THE NOISE FLOOR IS :"),mean(noise2)
snr=maximum-mean(noise2)-10*log10(asize/2)              # there is also the process
gain:10*log10(asize/2) #
snr_theoret=6.02*num_of_bits+1.76
print("*************** SNR *************************")
print ("SNR IS (quick and dirty) :" ),snr
print ("SNRtheoretical IS :"),snr_theoret

#*********** C A L C U L A T I O N  O F  D I S T O R T I O N ==> THD ***********#

print("******************* THD*****************************")
print("the REAL position of distortion is: "),realposition[:]

print("the REAL distortion is: "),realdistortion[:]
realposition=array(realposition)
realdistortion=array(realdistortion)
thd1=[]
```

```
for i in range (realdistortion.size):
    thd1.append(realdistortion[i]/20.0)
thd1=array(thd1)
thd2=thd1*2
thd2=array(thd2)
thd3=10**(thd2)
thd4=0
for i in range (realdistortion.size-1):
    if thd3[i]!=1 :
        thd4=thd4+thd3[i]


thd=10*log10(thd4)
print ("THD IS :"),thd

#******************C A L C U L A T I O N  O F  SINAD **********************#

print("***********SINAD **********************")
sinad1=-(snr)/10.0
sinad2=10**(sinad1)
sinad3=-abs(thd)/10.0
sinad4=10**(sinad3)
sinad5=sinad2+sinad4
sinad=-10*log10(sinad5)
print("SINAD IS :"),sinad


#******************** C A L C U L A T I O N  O F  ENOB **********************#
print("********** ENOB **********************")
factor=20*log10(full_scale_amplitude/real_amplit)
enob=(sinad-1.76+factor)/6.02
print ("ENOB IS :"),enob


# *********************************PLOTTING ******************************#
freqvector=array(freqvector)
fourievector=array(fourievector)
plt.plot(freqvector,fourievector)
plt.title("The FFT of the signal")
plt.xlabel("frequency")
plt.ylabel("dB")
plt.show()
```

4.2.1.Dynamic Testing Algorithm's Results

Below there is an execution of the Dynamic Testing Algorithm presented above.

```
GIVE ME THE NUMBER OF BITS :14
GIVE ME THE INPUT AMPLITUDE :5
GIVE ME THE FULLSCALE AMPLITUDE :5
GIVE ME THE NAME OF THE FILE:samples_2.dat
NUMBER OF SAMPLES ARE(-SIZE OF a IS) :  8192
THE SIZE OF a EQUALS TO:  2 TO THE  13.0

SIZE OF hhalf IS: 4096
the frequency of the sinewave is  1562500
maximum of hhalf is  -9.64327466553e-16
[  0.08               0.08000054    0.08000217  ...,    0.08000217
0.08000054  0.08       ]
THE NOISE FLOOR IS : -105.588459196


************************* SNR *******************************
SNR IS (quick and dirty) : 69.4648597164
SNRtheoretical IS : 86.04
************************* THD *******************************
the  REAL  position  of  distortion  is:    [3125000,  4687500,
10937500]
the     REAL     distortion     is:          [-87.859262900247728,
-81.114456373310162, -89.413998243431095]
THD IS : -80.2808572929
************************* SINAD *****************************
SINAD IS : 69.1190942732
************************* ENOB ******************************
ENOB IS : 11.1892183178
```

The samples used for this execution are real samples taken from FMC ADC 100M 14b 4cha ADC. As it can be seen from those results,there is a difference between the theoretical value of SNR,calculated using equation (4) and the real one calculated using equation (5).All of the values presented above are reasonable,although not very satisfying,according to specifications found on ADC datasheets. The algorithm also produces a plot of the digitized sine wave and a plot of the FFT of it and these are shown below.

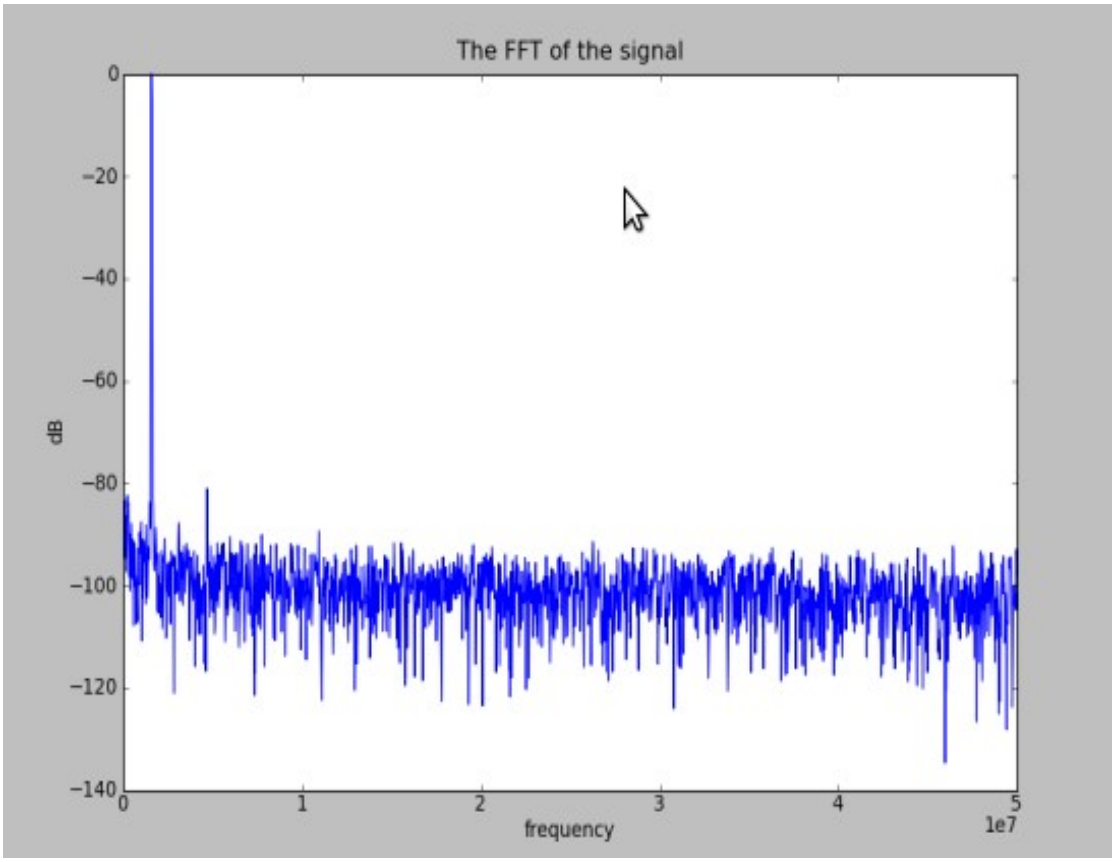

FIG.8-The Fourier Transform of the Sine Wave

The highest peak in figure 8 represents the sine wave' s frequency component. Some of the other peaks represent the distortion caused by the harmonics and the rest of the signal represents the noise floor.

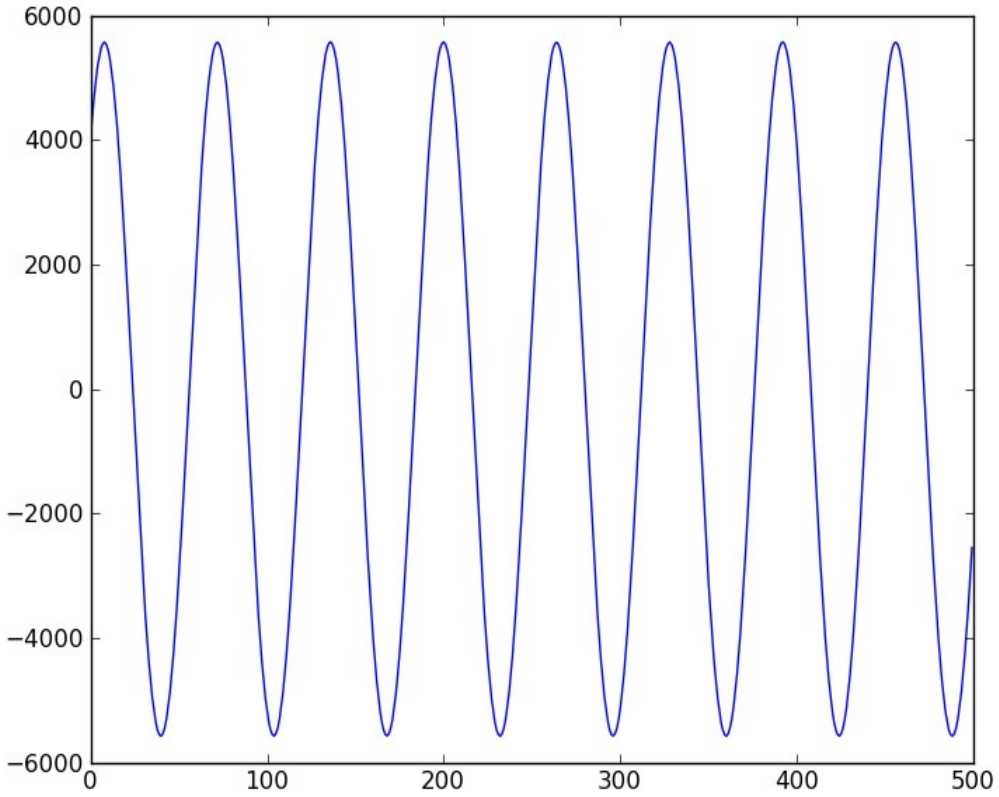

FIG.9-The Digitized Sine Wave

Notes :

1. Formula (11) was modified comparing to the ANALOG DEVICES document. A factor of 2 was multiplied by the argument of the $\sin^{-1}$ of the new argument is never larger than 1.
2. Formula (6) was also modified comparing to the ANALOG-DIGITAL CONVERSION document.

References

-ANALOG-DIGITAL CONVERSION-TESTING DATA CONVERTES

-ANALOG DEVICES : MT -003 TUTORIAL

-Advanced A/D and D/A Conversion Techniques and Their Applications -7th European Workshop on ADC Modelling and Testing

-IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters

-Waveform Recorder Testing: IEEE Standard 1057 and You [Thomas E. Linnenbrick]

-IEEE TRASNACTIONS ON INSTRUMENTATION AND MEASUREMENT ,VOL 57,NO 2,FEBRUARY 2008- Preliminary Considerations on ADC Standard Harmonization [Sergio Rapuano ,Member ,IEEE]

-Digital Signal Processing (4th Edition) :John G. Proakis ,Dimitris K. Manolakis

-Beginning Python (From Novice to Professional) : Magnus Lie Hetland

-http://www.maxim-ic.com/app-notes/index.mvp/id/2085

-http://www.maxim-ic.com/app-notes/index.mvp/id/729