ECE283 Lab3

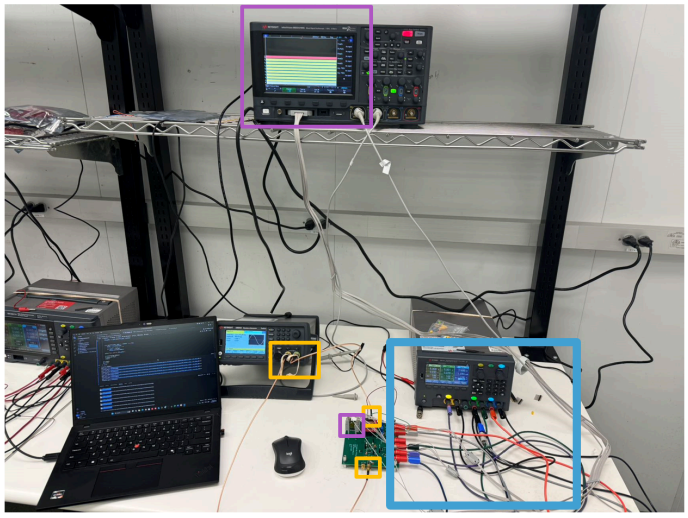
Chengming Li, Mingjie Ma

PID A59026442,A69028795

Email: chl248@ucsd.edu

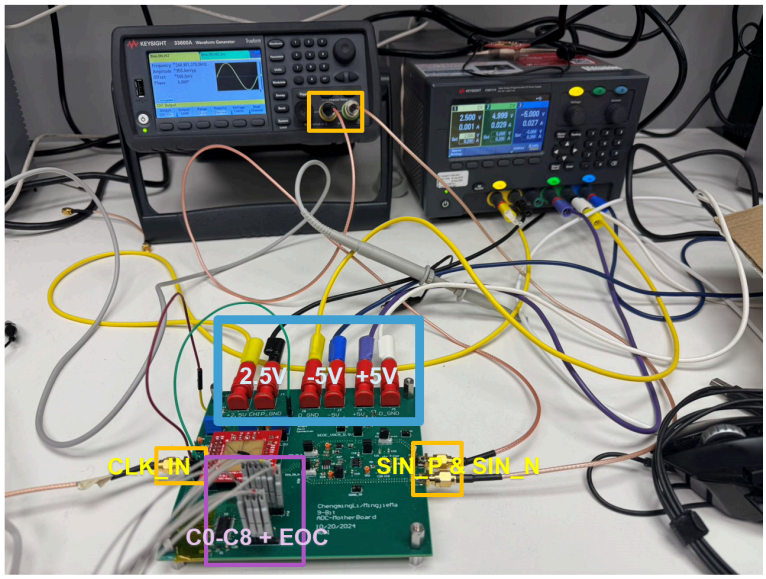
mim029@ucsd.edu

Photograph of your measurement setup, including motherboard with annotations.



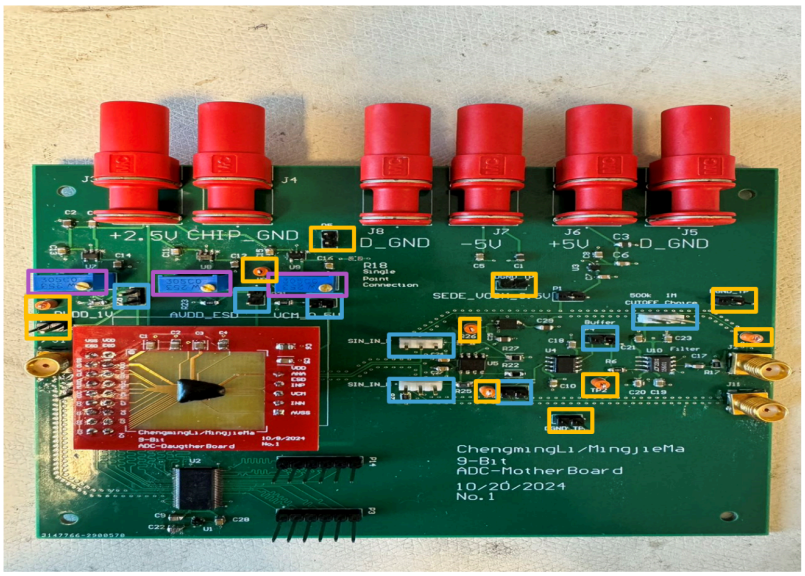
1. Power Supply
    1. +5V, -5V, and 2.5V
2. Function Generator
    1. CLK & SIN INPUT
3. Oscilloscope
    1. Digital Analyzer
    2. Digital Output
4. 4-Layer Stack
    1. Signal
    2. DGND + AGND
    3. +5V
    4. -5V

Fig.1 Measurement Setup & Automation



1. Power Supply
    1. +5V, -5V, and 2.5V
2. Function Generator
    1. CLK & SIN INPUT
3. Oscilloscope
    1. Digital Analyzer
    2. Digital Output

Fig.2 Measurement Setup Zoom IN



1. Modularized the Design
    1. By having jumper between each IC
    2. For debugging
    3. SE-DE / ByPass Options
2. Testability
    1. Jumper for GND for easy probeing
    2. Grab & Probe
3. Configurability
    1. Potentiometer for tweaking the perfect Vcm, AVDD, AVDDESD
4. 4-Layer Stack
    1. Signal
    2. DGND + AGND
    3. +5V
    4. -5V

Fig.3 MotherBoard

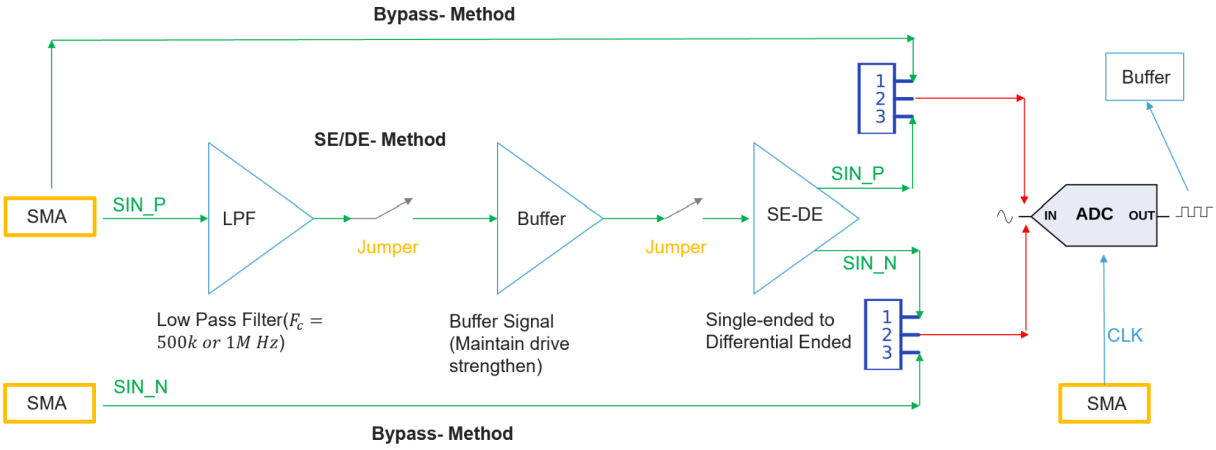
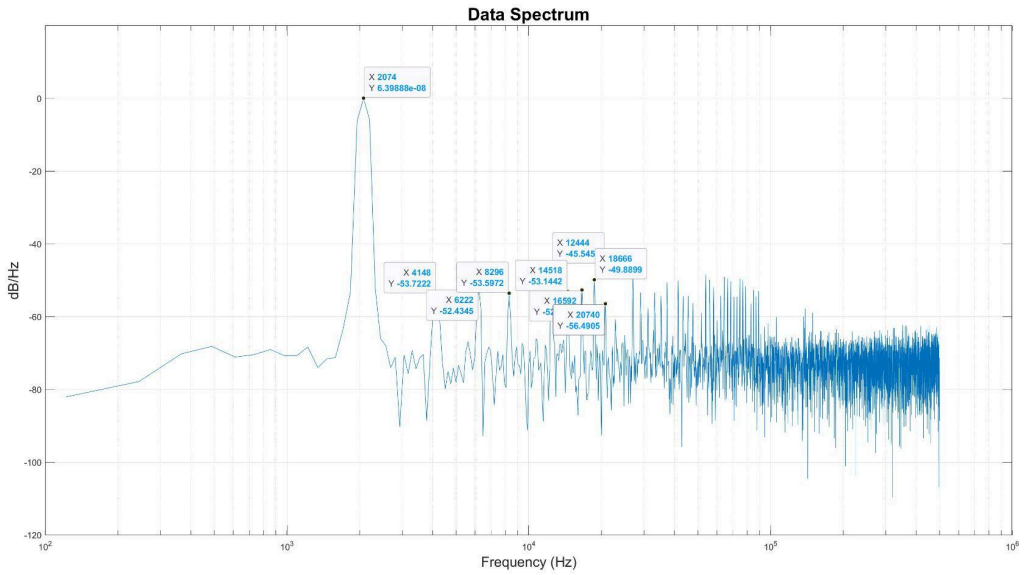


Fig.4 MotherBoard-HighLevel Block Diagram

Measured spectrum with a 0 dBFS input at 2 kHz

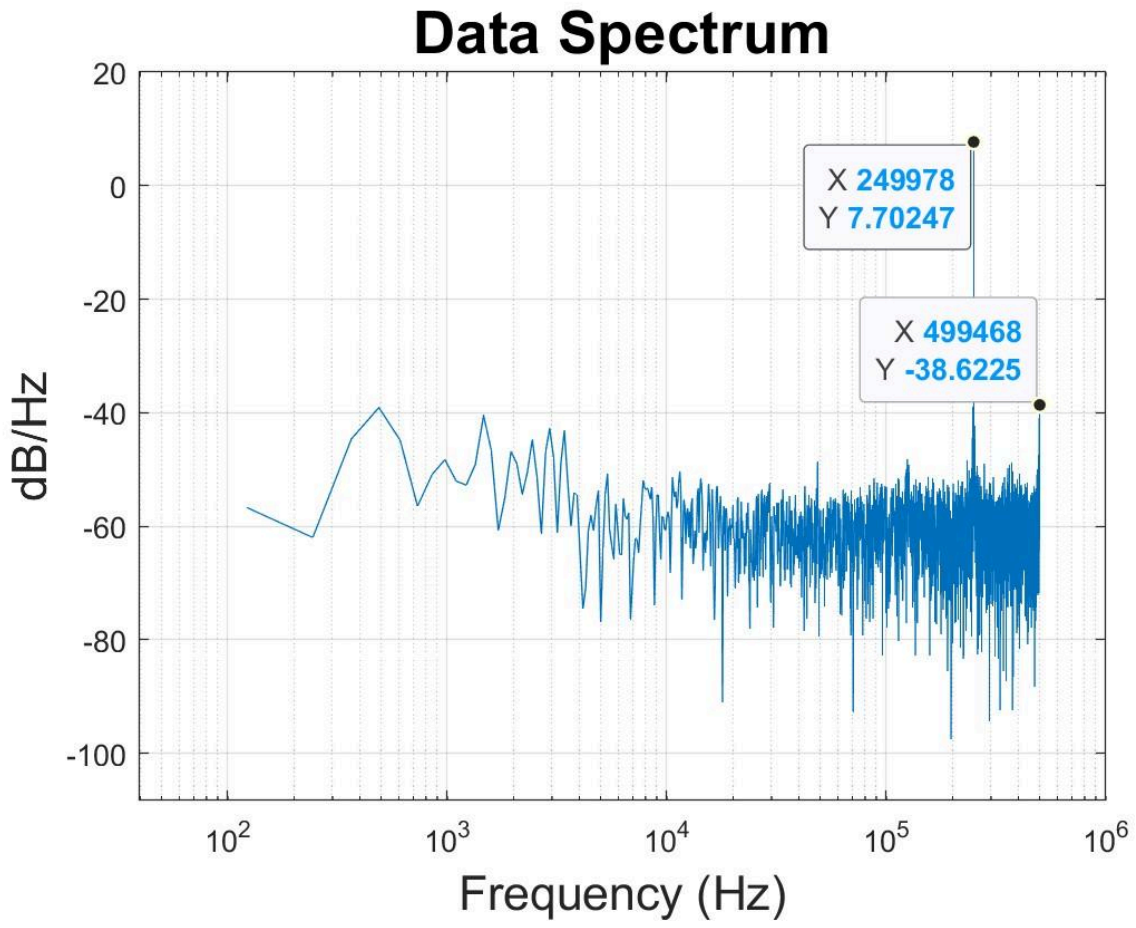


SNR = 33.924 dB, SNDR = 32.811 dB, SFDR = 52.445 dB, HD2 = 53.740 and HD3 = 52.445

This is a much lower than the simulation, as 2KHz, the simulation result should be around 50dB The result is related to how good the chip is. i.e. very low harmonics distortion and low noise floor.

- The above result show relative same magnitude of HD2 and HD3, which doesn't like what a differential input should behave
- And lots of very high harmonics coming from the clock coupling or might be the fluctuation on the chip VDD
- We guess we probably need to put decoupling caps on the daughter board to actually filter out the noise on chip VDD

Measured spectrum with a 0 dBFS input at 250 kHz

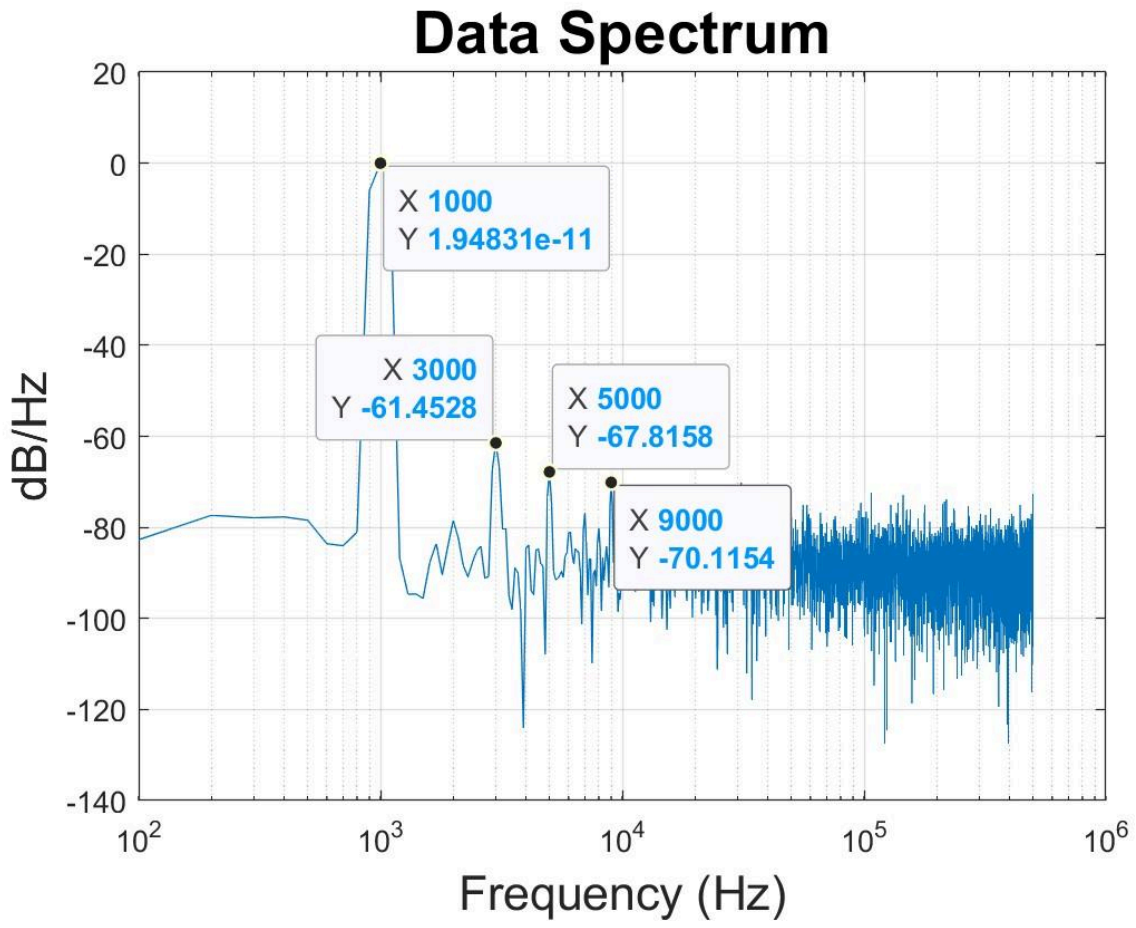


From TI Source

SNR = 30.600 dB, SNDR = -7.255 dB, SFDR = 40.562 dB, HD2 = 58.906 and HD3 = 40.562

- This also doesn't behave correctly.
- As the clock input frequency is 999.424e3 Hz. ideally, the 2nd harmonics should be hidden from the spectrum because the 2nd harmonics is right on the nyquist rate.
- And again, the noise floor is too high, this makes our ADC to be **noise-limited**
- And the 2nd harmonics is also way too high in a differential-input system

Measured spectrum with a 0 dBFS input at 1 kHz



SNR = 50.680 dB, SNDR = 50.228 dB, SFDR = 61.450 dB, HD2 = 78.362 and HD3 = 61.450

- This is the best performance we retrieved from our ADC at 1kHz
- It behave correctly as in a differential input system
- We only see the very high odd harmonics showing up, but not the even harmonics component in the spectrum.
- This spectrum also has a very low noise floor near -80 dB/Hz.
- But, this data has not been able to reproduce from chip to chip.
- We suspect there are some other ICs that are broken on the motherboard after we collected this dataset. And that's why we see unusual high noise floor on the previous two pages

Measured SNR, SFDR, and SNDR as a function of frequency (1-100 kHz, 10 points/decade).

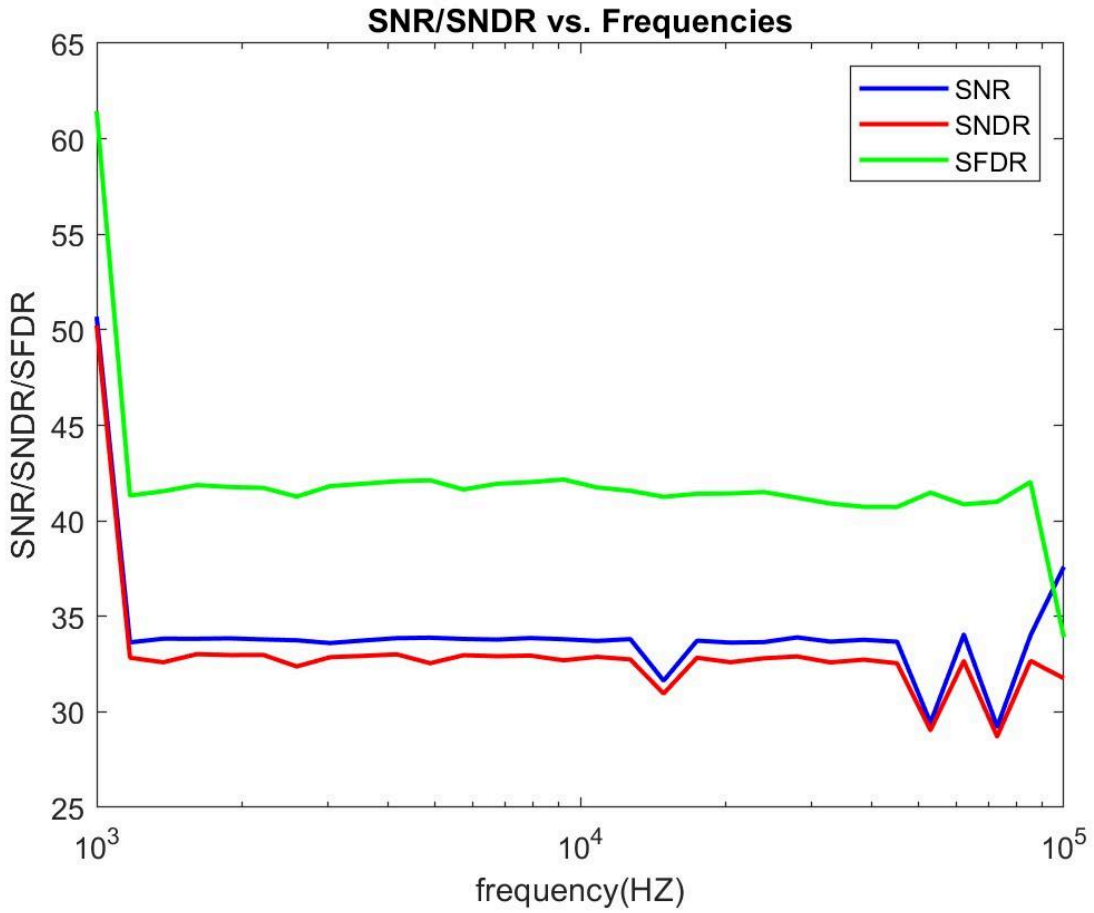
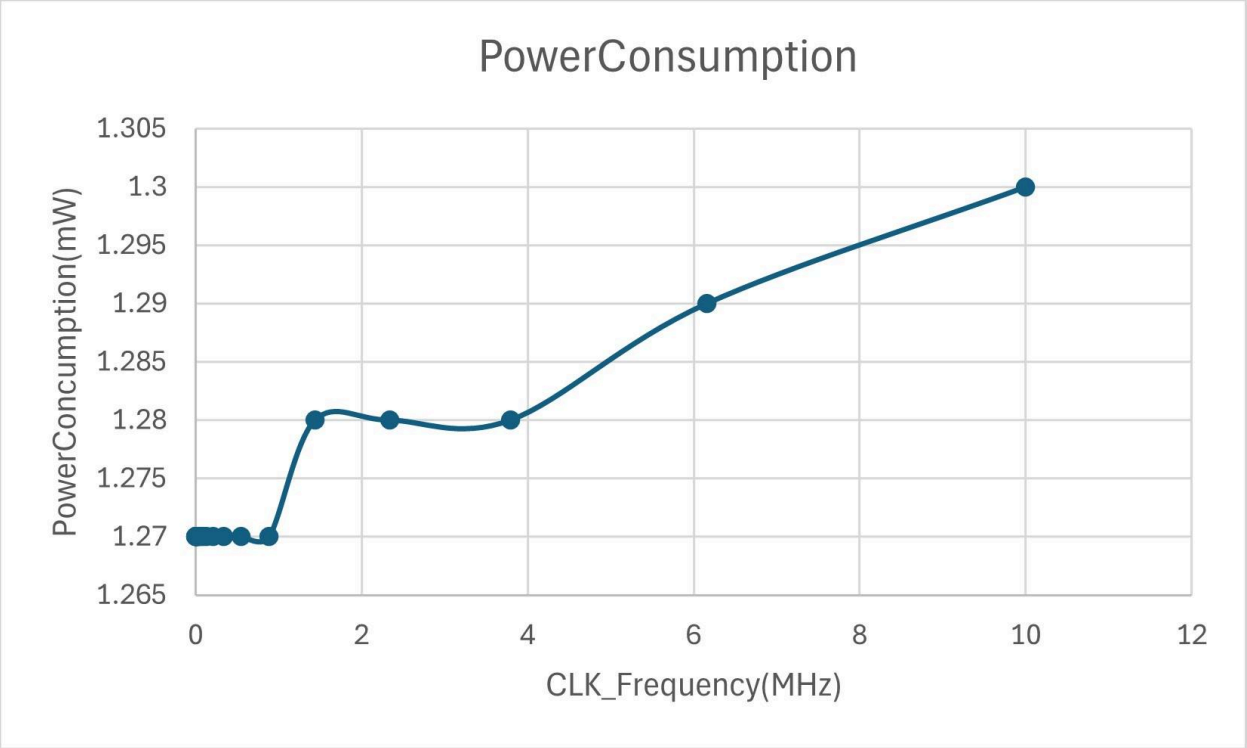


Fig. SND, SFDR, and SNDR as a function of frequency

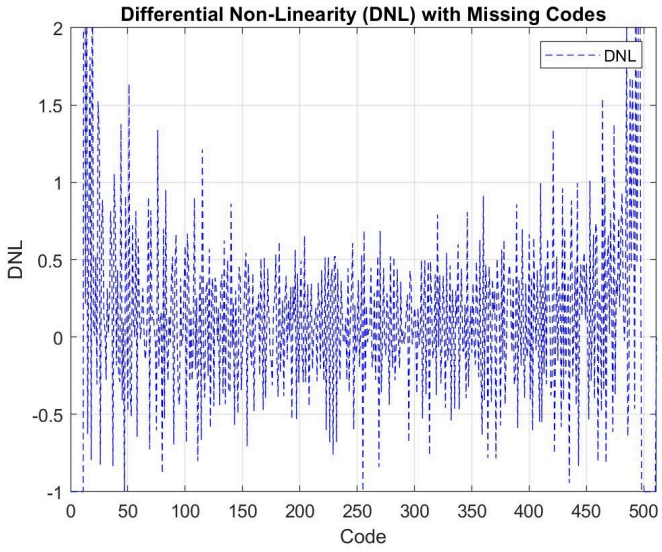
- Same dataset is used as in the previous page.
- We doubt the reason we see this big drop after 1kHz is because we didn't match : f_in/f_sample = (# *integer cycles*)/(# FFT_points) exactly either in the automation or in the code.
- But the result seems pretty similar across frequency
- Maybe this is our ADC's actual performance.

Measured power consumption vs. sampling frequency (1 kHz – 10 MHz)

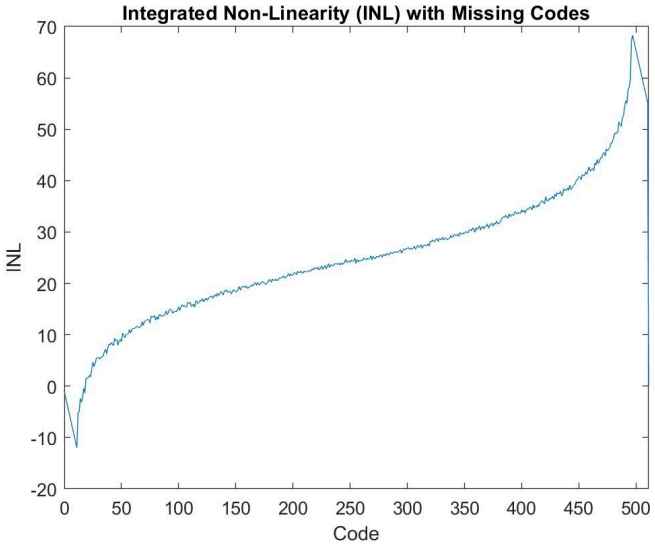


- Our power consumption is higher than other groups in the class because our chip-VDD is actually powering up the entire mega group's Chip. And so, this is not so easy to actually tell the actual power consumption of our chip is.
- But the behavior is expected. As the clk frequency increases, the Power consumption goes up.
- At low CLK frequency, the power consumption remains constant.

Measured INL and DNL

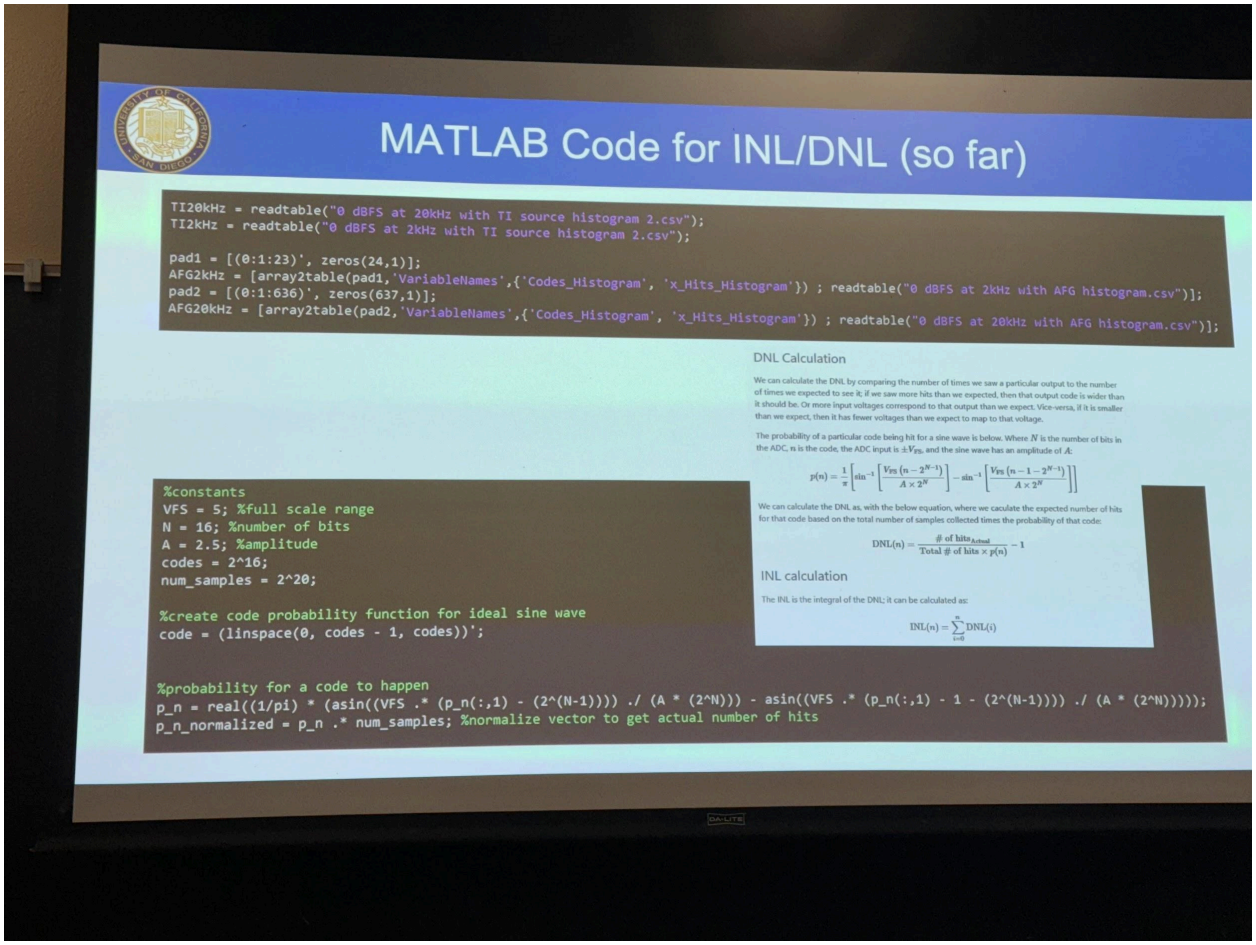


DNL



INL

DNL and INL Formula is provided by TA:Darshan. Formula can be seen velow



Basically, the INL and DNL data is converted from the Codes# and # of hits, which is exported from our ADC's code. And following the formula, gives the above results.

1. Codes # and # of hits from ADC's code
2. Import into Matlab
3. Data processing according to the formula provided above
4. Plot the resulting INL and DNL according to each code
5. Notes: the result for INL and DNL is about correct. As the input signal is a sine wave, we should expect the low code and high code to exceed 1LSB, because the slope is changing very slowly. And so more code are captured in these two places
6. The ideal measurement method to test the DNL and INL should be a ramp signal. So that every code can be captured equally with the same slope.
7. The incorrect input signal as SIN also cause abnormal behavior is seen in the INL data

MATLAB/Python/LabVIEW code

Keysight Connection Expert & Keysight Command Expert is used to conduct all the instrument control code

Python_Instrument_Code

1. Function Generator(KS33600A_Control.py)
    1. Freq_coupling', 'FuncGenConnect', 'Load__setup', 'PWM_setup', 'Polarity_invert', 'Sin_setup', 'Sync_phase', 'UnitVpp_setup', 'Voltage_coupling', 'output_OFF', 'output_ON
2. PowerSupply(KSE36311A_Control.py)
    1. 'Channel_Select', 'Current_Setup', 'Output_OFF', 'Output_ON', 'PowerSupply_Connect', 'Voltage_Setup'
3. Oscilloscope(KSMSOX3104G.py)
    1. 'DigitalDisplay_ON', 'Digitalizer_ON_OFF', 'Oscilloscope_Connect', 'Oscilloscope_RUN', 'Oscilloscope_SetBUS', 'Oscilloscope_Single', 'Oscilloscope_Stop', 'Oscilloscope_TimeBase', 'Oscilloscope_Trigger_Dchan', 'Oscilloscope_Trigger_External', 'Oscilloscope_WGen_Square', 'Oscilloscope_WGen_Square_OFF', 'Oscilloscope_WGen_Square_ON', 'Save_waveform', 'SetDigital_Threshold
4. Automation(Lab4_KS-SEDE-EthenBench.ipynb)

Naming is due to there so many different sets of instruments and with different instrument ID. I used the people's name to distinguish which set of instruments is.

GitHub Link:
https://github.com/stevenli518/ECE266_CMOSCircuitLab/tree/main/Lab7_ICTesting/InstrumentControlCode

Matlab_Data-processing_Code

1. MatLabCode\ADC_Data\Board2\11-25-Python-Sweep-1MHz-50%-1020mV/Good_Sampling_Script.mlx
2. FFT plot, SNR, SNDR, SFDR, INL, and DNL are included.

GitHub Link:
https://github.com/stevenli518/ECE266_CMOSCircuitLab/tree/main/Lab7_ICTesting/MatLabCode/ADC_Data/Board2/11-25-Python-Sweep-1MHz-50%25-1020mV

Comments and conclusions

1. How my GPIB code works:
    a. Files: KS33600A_Control.py, KSE36311A_Control.py, KSMSOX3104G.py, Lab4_KS-SEDE-EthenBench.ipynb
    b. Instrument control code are all written in the library format, which increase the re-useability in the test-automation
    c. Heavily relied on the Keysight command expert to develop the code, instead of reading the datasheet.
    d. I use python library, pyvisa
    e. It has the function “open_resource”, and one input as the address of the instrument. The address can be get from function “list_resources”
    f. Then I write a control library for both the keysight function generator by using the keysight command expert to retrieve all the command we need
    g. Remote-control setup the following parameters:
        i. Voltage High/Low
        ii. Waveform Shape -Sine
        iii. Offset
        iv. Amplitude
        v. Frequency
        vi. Phase
        vii. Output Channel
        viii. Output load -HI Z
        ix. Output turn ON/OFF
    h. Use my library can simply sweep the frequency each time during the “for” loop
2. My test automation follows below documentation:

a.

SE-DE Test Setup

1. Power Supply
    a. 1 channel is used as 5V, 0.2A
    b. 1 channel is used as -5V, 0.2A
    c. 1 channel is used as 2.5V, 0.2A
2. Function Generator
    a. SIN Input - Channel 1
        i. High-Z Load
        ii. 1Vpp
        iii. 0 V offset
        iv. Frequency at 1KHz as initial
    b. Clock Input – Channel2
        i. High-Z Load
        ii. 2.5 Vpp
        iii. 1.25 Offset
        iv. Frequency at 1MHz
        v. Duty Cycle 50%
    c. Sync Both Channels every time the parameters are changed
3. Oscilloscope
    a. Time Base 1ms/div
    b. Digital Bus1
        i. Digital Code 0-8 (C0-8)
        ii. Digital Code 11 - End of Code (EOC)
        iii. Trigger Level 1.8V
        iv. Turns off Others
    c. Trigger Source
        i. EOC, D11 Rising Edge
        ii. Possibly CLK from Function Generator OR CLK from buffer
    d. Collect data
        i. Stop or Single
        ii. MSB First
        iii. ASCII format
        iv. Maximum Points

b.

Bypass Setup

1. Power Supply
    a. 1 channel is used as 5V, 0.2A
    b. 1 channel is used as -5V, 0.2A
    c. 1 channel is used as 2.5V, 0.2A
2. Function Generator
    a. SIN Input - Channel 1
        i. High-Z Load
        ii. 1Vpp
        iii. 0 V offset
        iv. Frequency at 1KHz as initial
    b. SIN Input Inverted - Channel 2
        i. Polarity Inverted
        ii. High-Z Load
        iii. 1Vpp
        iv. 0 V offset
        v. Frequency at 1KHz as initial
    c. Sync Both Channels every time the parameters are changed
3. Oscilloscope
    a. WaveGen used as CLK
        i. Load – Infinity
        ii. Frequency – 1M
        iii. Amplitude 2.5V
        iv. Offset – 1.25V
        v. Duty Cycle 50%
    b. Time Base 1ms/div
    c. Digital Bus1
        i. Digital Code 0-8 (C0-8)
        ii. Digital Code 11 - End of Code (EOC)
        iii. Trigger Level 1.8V
        iv. Turns off Others
    d. Trigger Source
        i. EOC, D11 Rising Edge
        ii. Possibly CLK from Function Generator OR CLK from buffer
    e. Collect data
        i. Stop or Single
        ii. MSB First
        iii. ASCII format
        iv. Maximum Points

3. How my data processing code works:
    a. Files:Good_Sampling_Script.mlx

b. Read-table from a .csv file, which saved all the digital code data coming from the python automation script
c. And convert the data to binary
d. Filtering out the data by only capturing the rising edge of EOC. i.e. previous EOC = 0, and current EOC = 1
e. Feed the after_filtering out into SNR/SNDR/SFDR formulas from Prof.Hall
f. Additional SNR is computed using Matlab built in function snr()
g. SFDR is computed as Signal Power / (2nd or 3rd) Harmonics power
h. SFDR is either 2nd or 3rd, whoever is the highest
i. Using INL and DNL formulas provided by TA Darshan
j. Plot the data accordingly.

4. Comments:
    a. Hugh fluctuation on chip'sVDD, as 600mVpp. We suspect this is the mainly reason that our ADC didn't perform properly. We should stack up more decoupling capacitors on the daughter if more time is allowed
    b. Coherent sampling: f_in/f_sample = (# *integer cycles*)/(# FFT_points) didn't match exactly either in the test automation, or in the code. Because, after filtering out the data, it has a different sampling frequency than the frequency used in the instrument.
    c. We should wire-bond all the pads out, instead of leaving them floating inside the chip. This may cause bad performance in our ADC
    d. On our daughter board, the substrate is floating, this is another reason that may worsen our ADC's performance.