

Theory, Design, and Testing of Nyquist Data Converters

Germano Nicollini

Company Fellow

AMG R&D

E-mail: germano.nicollini@st.com

ACKNOWLEDGMENTS

Many thanks to **Pierangelo Confalonieri**, **Marco Zamprogno**, **Riccardo Martignone**, **Federico Guanziroli**, and **Alberto Minuti** for the several discussions about theory and design of Nyquist data converters and the many successful silicon realizations. Thank also to **Paolo Pesenti** for discussions about ADC metastability.



ACKNOWLEDGMENTS

Thank to **Daniele Devecchi** for the HW/SW implementations of many data converter validations.



Theory and Design of Nyquist Data Converters

- Data converter theory
 - Data converter non-idealities
- Data converter design
 - Basic data converter (DAC) typologies
 - INL/DNL: Effect of short-range mismatches
 - INL/DNL: Effect of long-range mismatches
 - INL/DNL: Effect of output impedance in current steering data converters
 - DACs with main and sub arrays
 - Self-calibration
 - Redundancy in ADCs
 - Metastability in ADCs
 - Performance limitations in ADCs
 - Dynamic performance limitations in DACs
 - Power-supply decoupling

Validation of Nyquist Data Converters

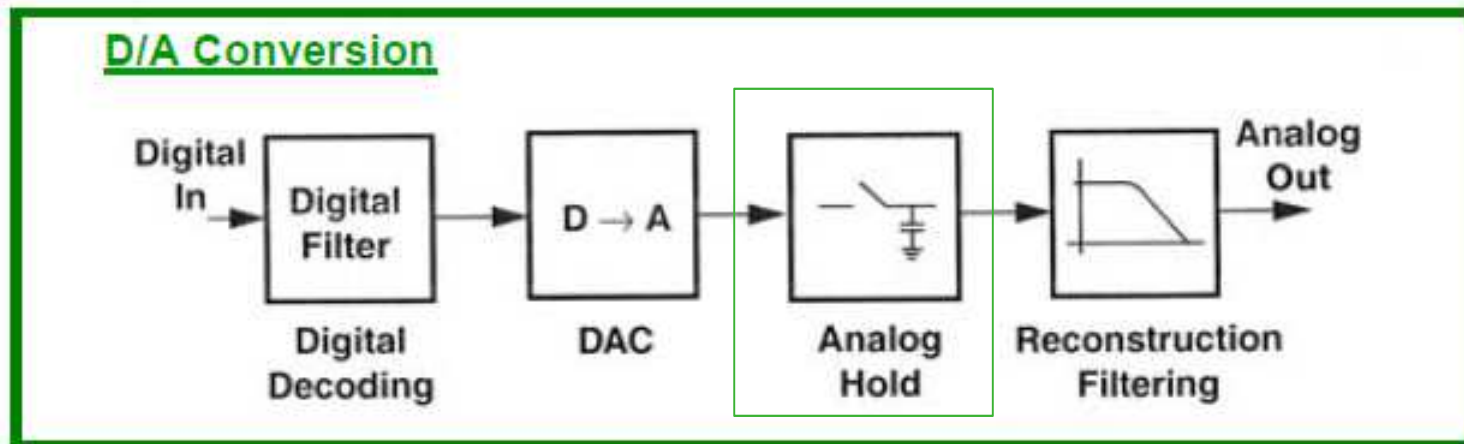
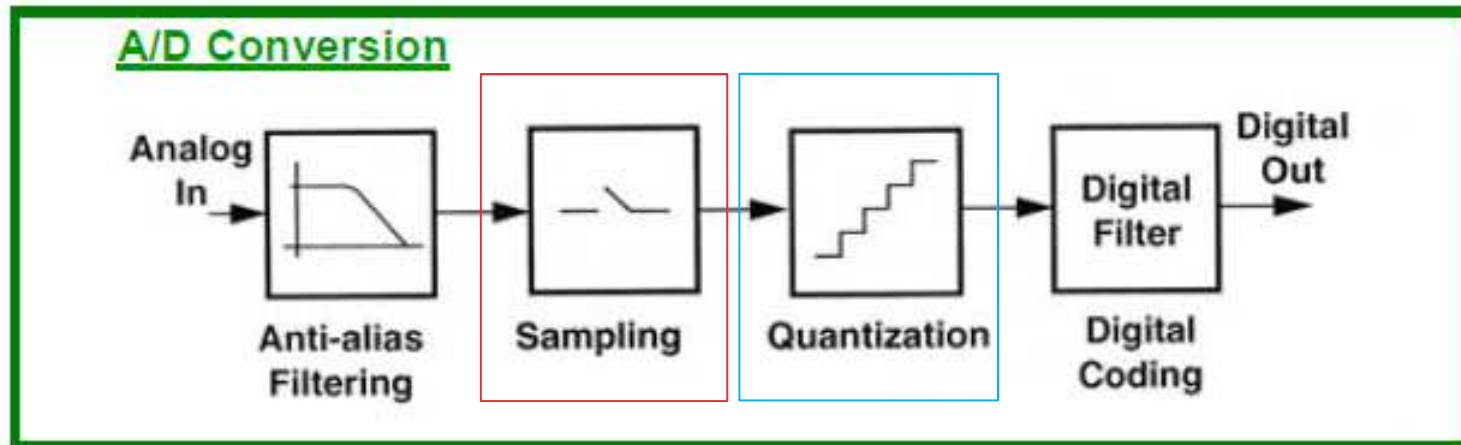
- Static testing
 - DAC testing
 - ADC testing
- Dynamic (spectral) testing
 - DAC testing
 - ADC testing

Data converter theory

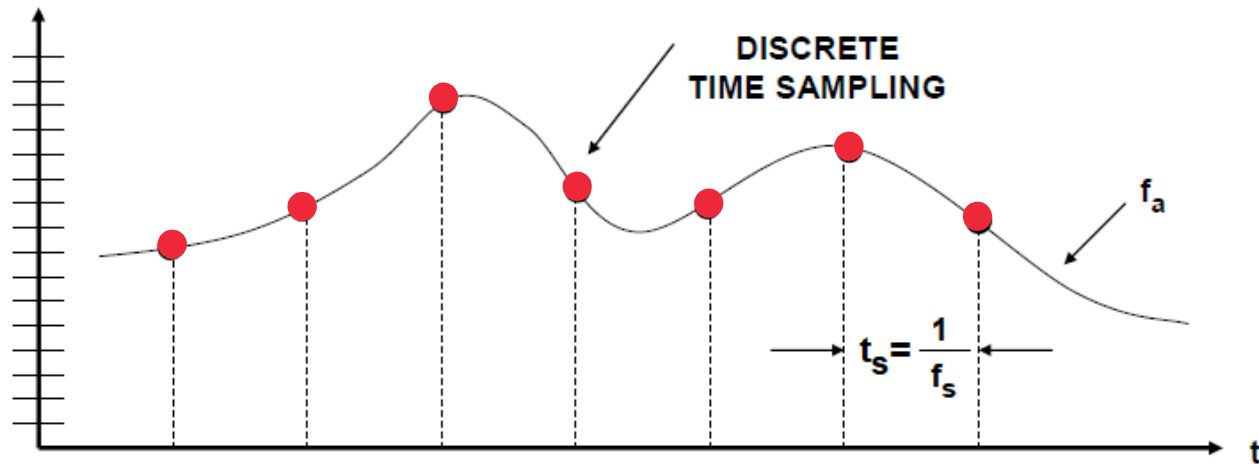
Data converter theory

- A/D and D/A conversion
- Sampling process
- Quantization process
- Sampling ADCs
- Zero-order holding process

A/D and D/A conversion



Sampling process (I)

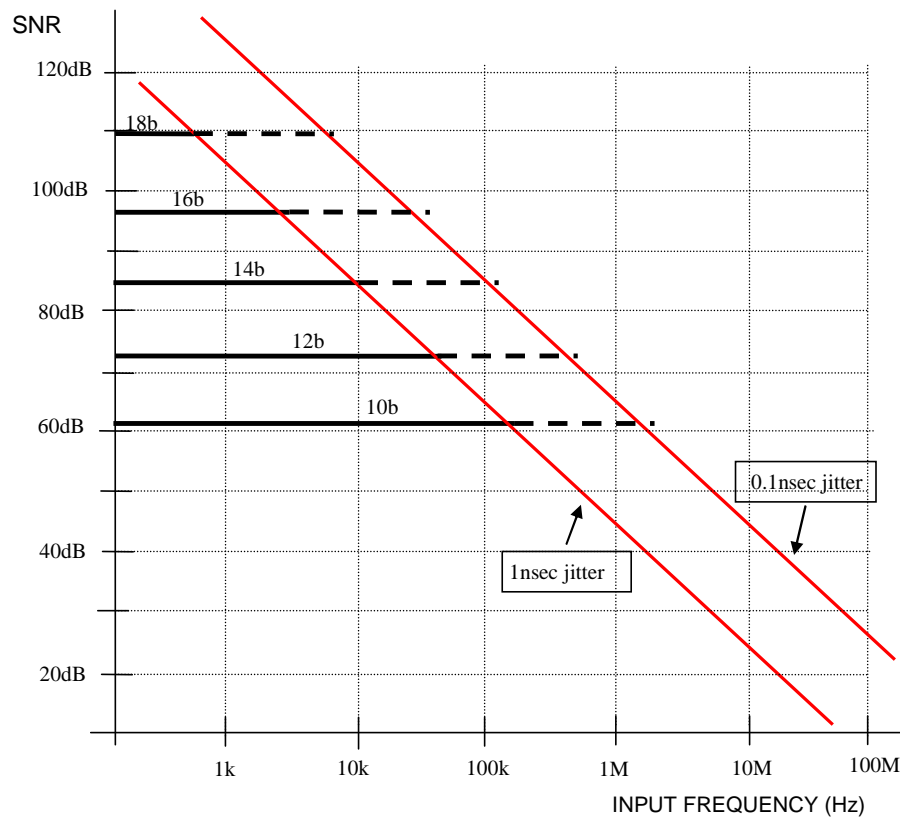


Sampling theorem [1] \rightarrow Nyquist criterion $\rightarrow f_s > 2f_{in_max}$

Sampling process (II)

Sampling jitter and switch aperture jitter introduce errors (noise) proportional to the first derivative of the ADC input signal [2].

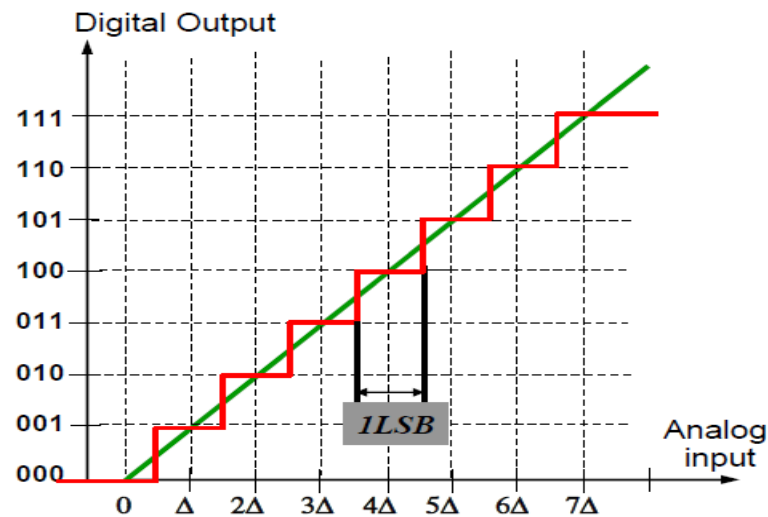
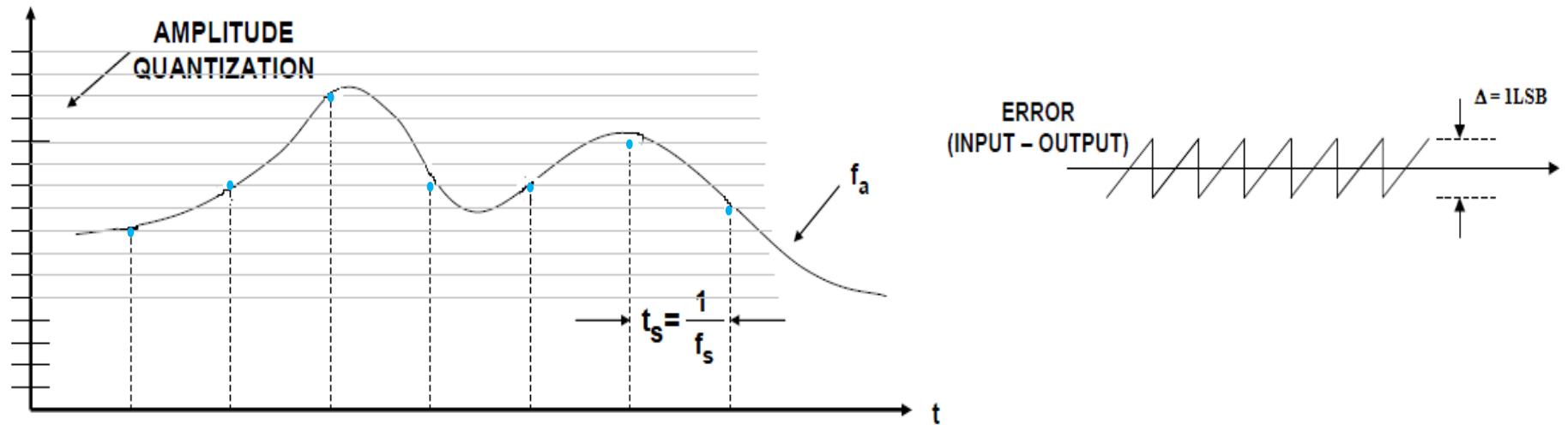
Jitter theory, impact, and conclusions are also valid for DACs.



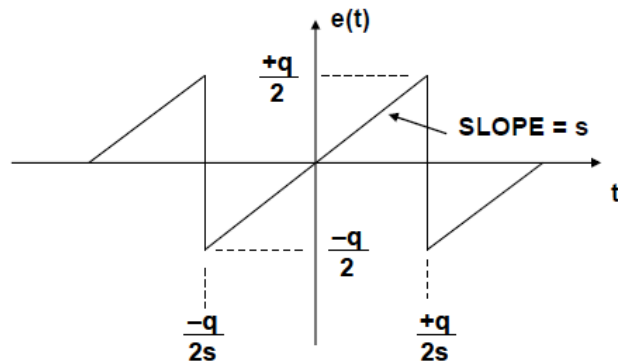
$$SNR = 20 \log_{10} \frac{1}{2\pi f_a \sigma(j)}$$

$\sigma(j)$ is the rms value of the jitter

Quantization (digitalization) process (I)



Quantization process (II)

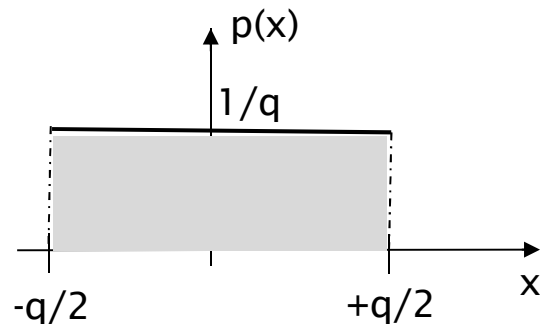


$$e(t) = st, \quad -\frac{q}{2s} < t < +\frac{q}{2s}$$

$$\overline{e^2(t)} = \frac{s}{q} \int_{-q/2s}^{+q/2s} (st)^2 dt = \frac{q^2}{12}$$



$$\text{rms quantization noise} = \frac{q}{\sqrt{12}}$$



$$\begin{cases} p(x) = \frac{1}{q}, & -\frac{q}{2} < x < +\frac{q}{2} \\ p(x) = 0, & \text{otherwise} \end{cases}$$

$$\overline{e^2(t)} = \int_{-\infty}^{+\infty} x^2 p(x) dx = \int_{-q/2}^{+q/2} \frac{x^2}{q} dx = \frac{q^2}{12}$$



$$\text{rms quantization noise} = \frac{q}{\sqrt{12}}$$

Quantization process (III)

a) FS sinewave signal = $\frac{q 2^N}{2} \sin(2\pi f_a t)$ \longrightarrow rms sinewave value = $\frac{q 2^N}{2\sqrt{2}}$

$$SNR = 20 \log_{10} \frac{q 2^N / 2\sqrt{2}}{\frac{q}{\sqrt{12}}} = 20 \log_{10} 2^N + 20 \log_{10} \sqrt{\frac{3}{2}}$$

$$SNR = 6.02N + 1.76 \text{ [dB]} \quad \text{over the dc to } f_s/2 \text{ bandwidth}$$

b) FS Triangular signal: rms value = $\frac{q 2^N}{\sqrt{12}}$

$$SNR = 20 \log_{10} \frac{q 2^N / \sqrt{12}}{\frac{q}{\sqrt{12}}} = 20 \log_{10} 2^N$$

$$SNR = 6.02N \text{ [dB]} \quad \text{over the dc to } f_s/2 \text{ bandwidth}$$

c) Random Gaussian signal: rms value = $\frac{q 2^N}{8}$
(4σ equal to FS range):

$$SNR = 20 \log_{10} \frac{q 2^N / 8}{\frac{q}{\sqrt{12}}} = 20 \log_{10} 2^N + 20 \log_{10} \frac{\sqrt{3}}{4}$$

$$SNR = 6.02N - 7.27 \text{ [dB]} \quad \text{over the dc to } f_s/2 \text{ bandwidth}$$

Quantization process (IV)

Assumptions:

- All quantization levels are exercised with equal probability
- The quantization noise is uncorrelated to the input signal
- A large number of quantization levels is used.

These assumptions are:

- true only for signals with uniform amplitude distribution, e.g. FS triangular waves
- approximately true for FS wide band sinewaves
- absolutely false for small signals and/or narrow band signals, e.g. dc inputs.

When these assumptions are valid, quantization noise is more or less uniform over the Nyquist bandwidth from dc to $f_s/2$ [3].

Otherwise, part of the quantization noise/error energy is located to (sub) harmonics of the signal.

In any case, the rms value of the quantization noise/error remains approximately $\frac{q}{\sqrt{12}}$

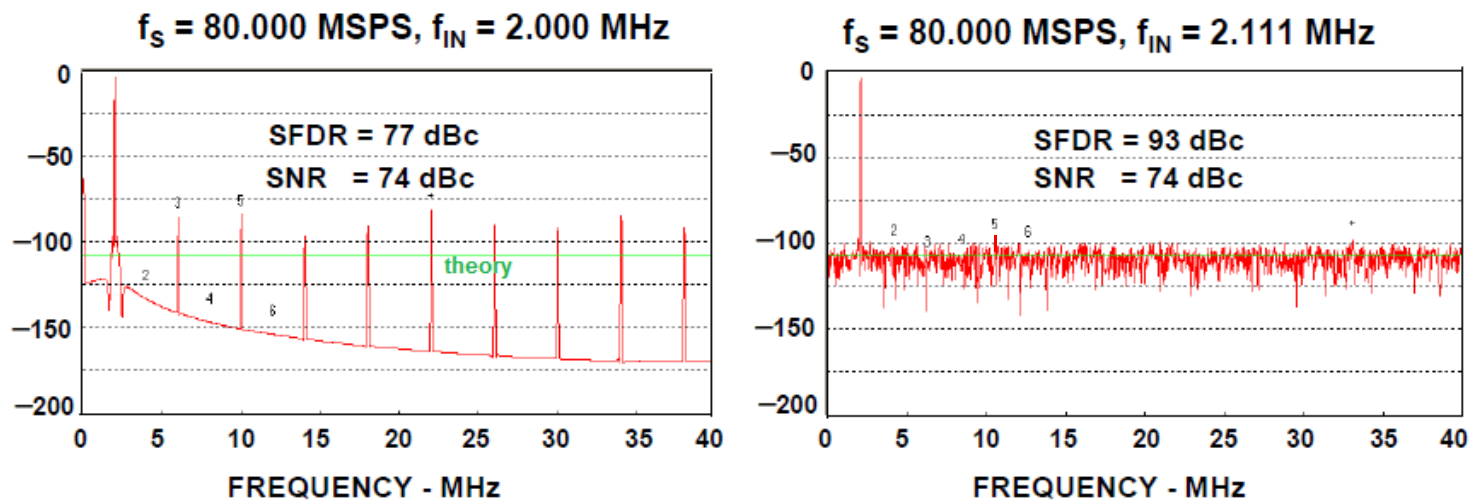
Quantization/sampling effects on large sinewave signals

For a large sinewave signal, quantization noise becomes correlated with input signal:

-- a bit of the energy is concentrated in harmonics of the signal if f_s/f_{in} is an irrational number [4]

$$SFDR \cong 8.07N + 3.29 \text{ [dB]}$$

-- most of the energy is concentrated in the (sub)harmonics of the signal if f_s/f_{in} is a rational number

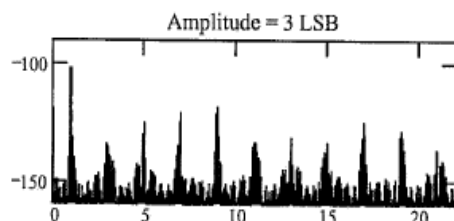
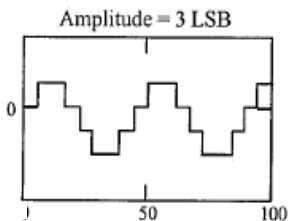
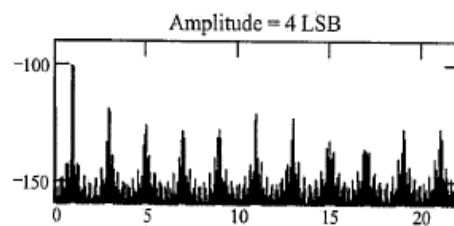
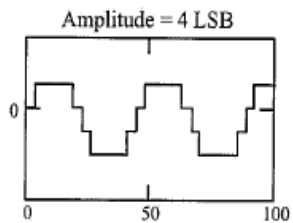
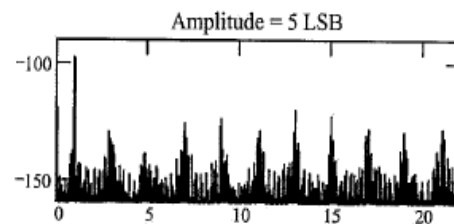
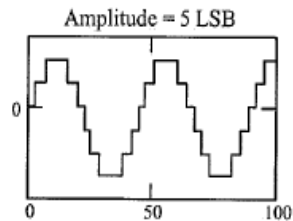


Pay attention to this artefact of the sampling process when performing data converter testing with a single-tone sinewave. Steps must be taken to measure ADC and DAC distortions, not artefacts!

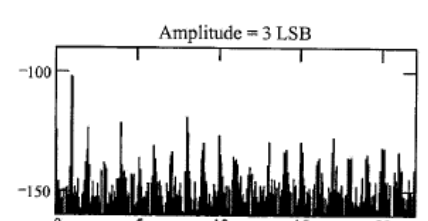
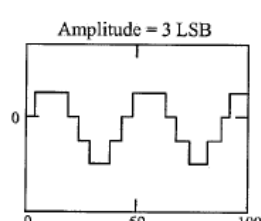
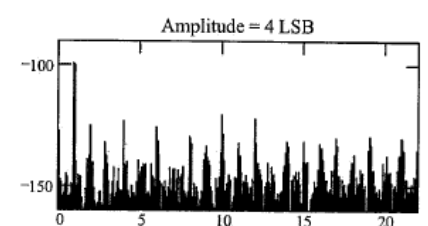
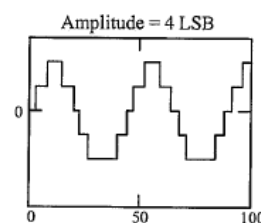
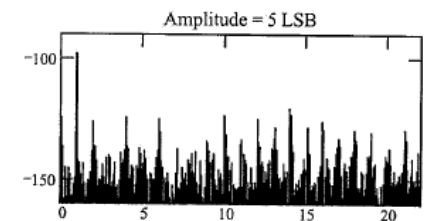
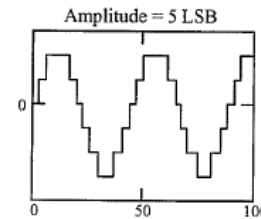
Quantization effects on small signals and dc signals (I)

Example: 16b ideal ADC

No offset



offset = 0.25 LSB



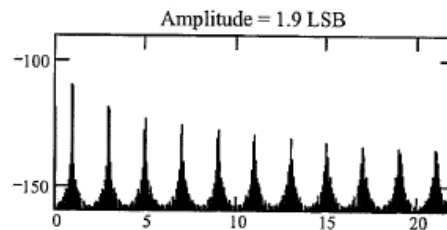
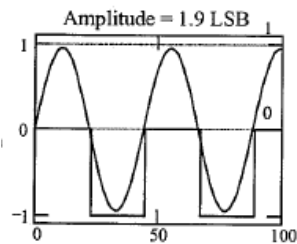
– Zero offset causes odd harmonics

Non-zero offset adds even harmonics

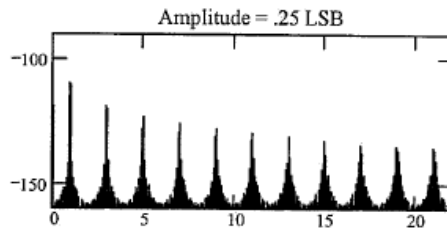
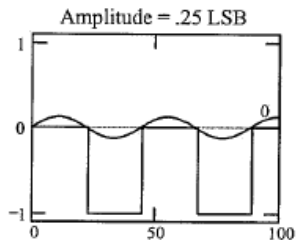
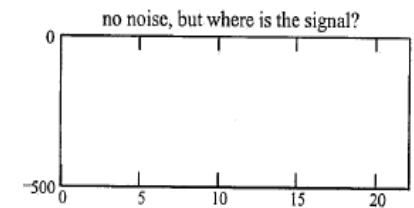
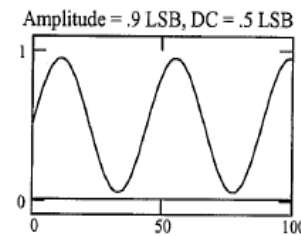
Quantization effects on small signals and dc signals (II)

Example: 16b ideal ADC

No offset



offset 0.5 LSB



No noise, but no signal

Impossible to distinguish these two signals

The behavior shown in these examples is called noise modulation.

Sampling ADC

- Sampling ADCs have a built-in input S&H circuit that allow them to process dc and ac signals.
- Encoders are not equipped with any input S&H function, thus they can process only dc or very low frequency signals.

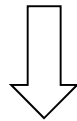
Example -- Input frequency limitations of non-sampling ADC (Encoder)

$$v(t) = \frac{q 2^N}{2} \sin(2\pi f_a t) \quad \frac{dv(t)}{dt} \Big|_{\max} = 2\pi f_a \frac{q 2^N}{2} \quad \longrightarrow \quad f_a = \frac{\frac{dv(t)}{dt} \Big|_{\max}}{q\pi 2^N}$$

If N=12b and 1LSB change ($dV = q$) is allowed during the conversion time ($dt = 8\mu s$) it results

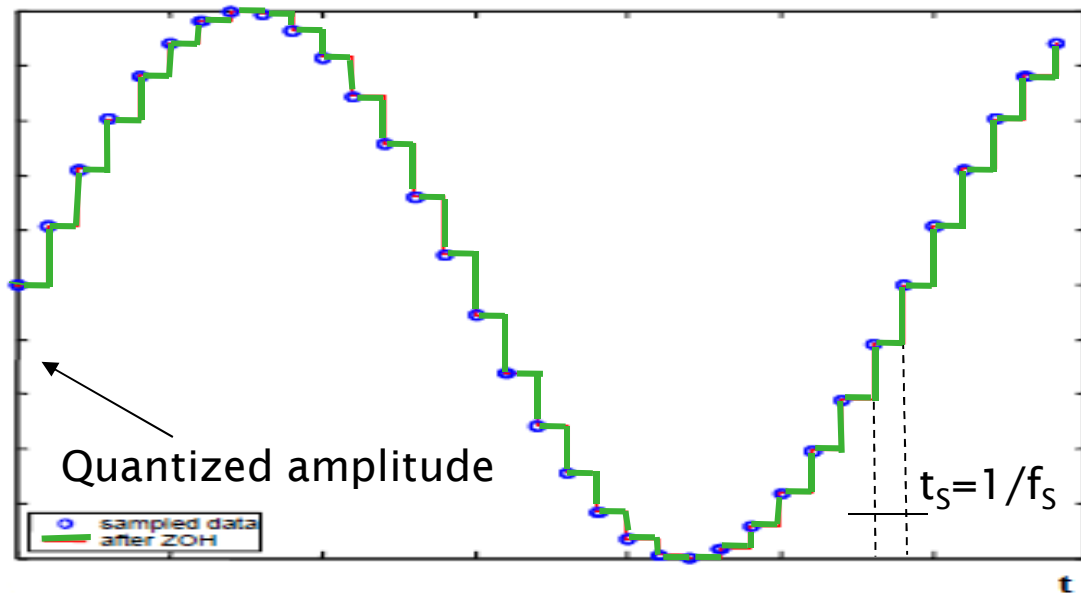
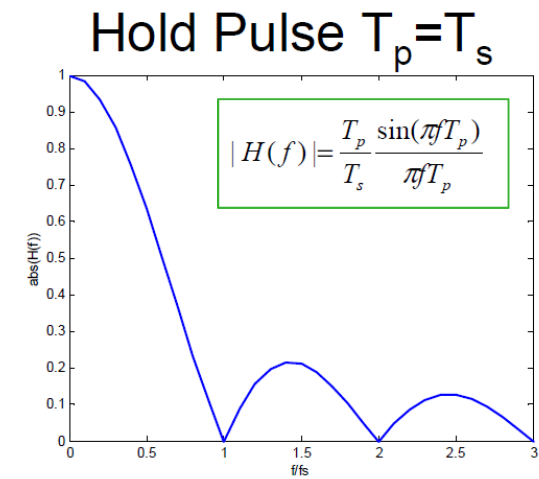
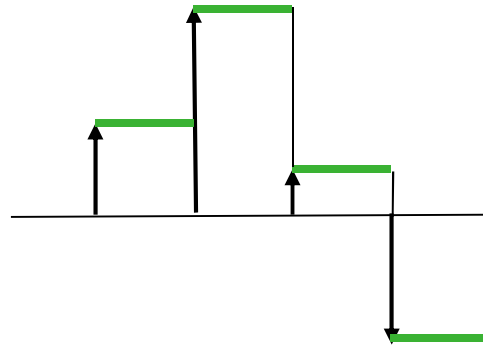
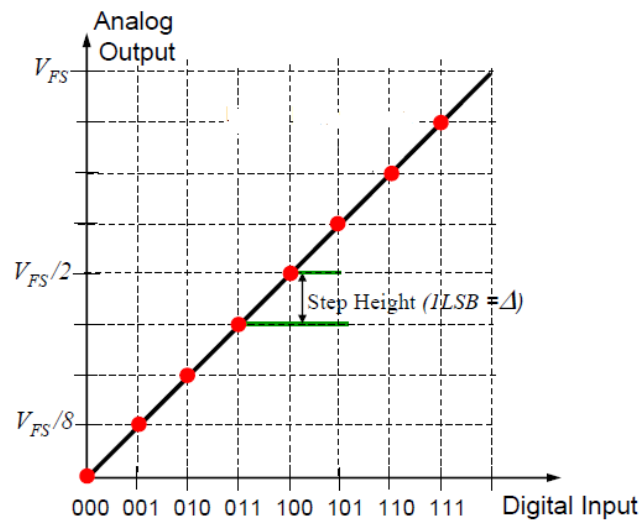
$$f_a = 9.7Hz$$

This imply that any input frequency higher than 9.7Hz is subject to conversion errors, even if a sampling frequency of 125kSpS (max input frequency 62.5kHz) is possible for the ADC.



Always use sampling ADCs

Zero-order holding process



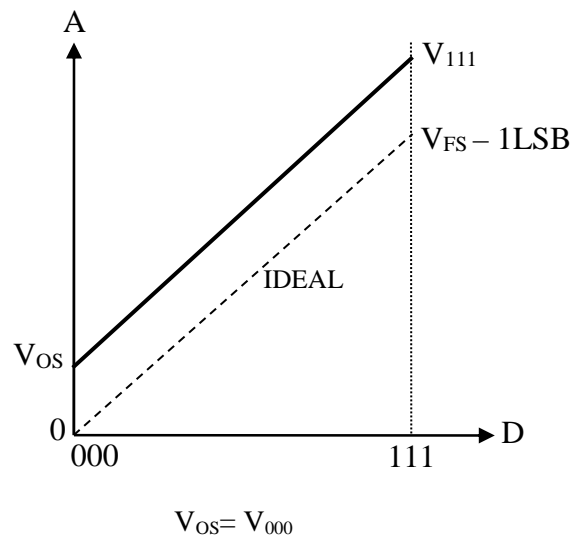
Data converter non-idealities

Data converter non-idealities

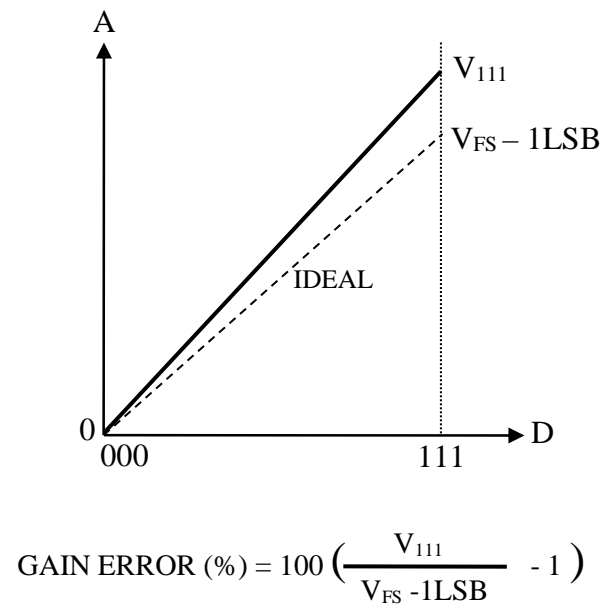
- DAC offset and gain errors
- DAC Differential Non-Linearity (DNL) and Integral Non-Linearity (INL)
- ADC offset and gain errors
- ADC Differential Non-Linearity (DNL) and Integral Non-Linearity (INL)
- Code transition noise
 - Noise dither

DAC offset and gain errors

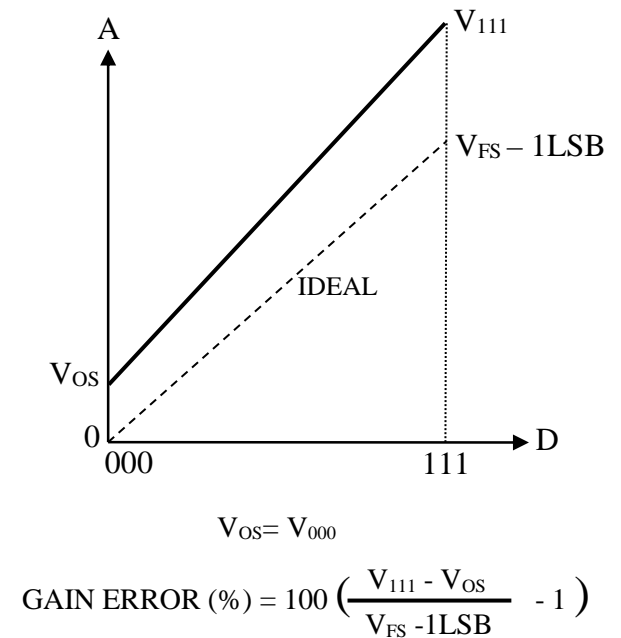
OFFSET ERROR



GAIN ERROR



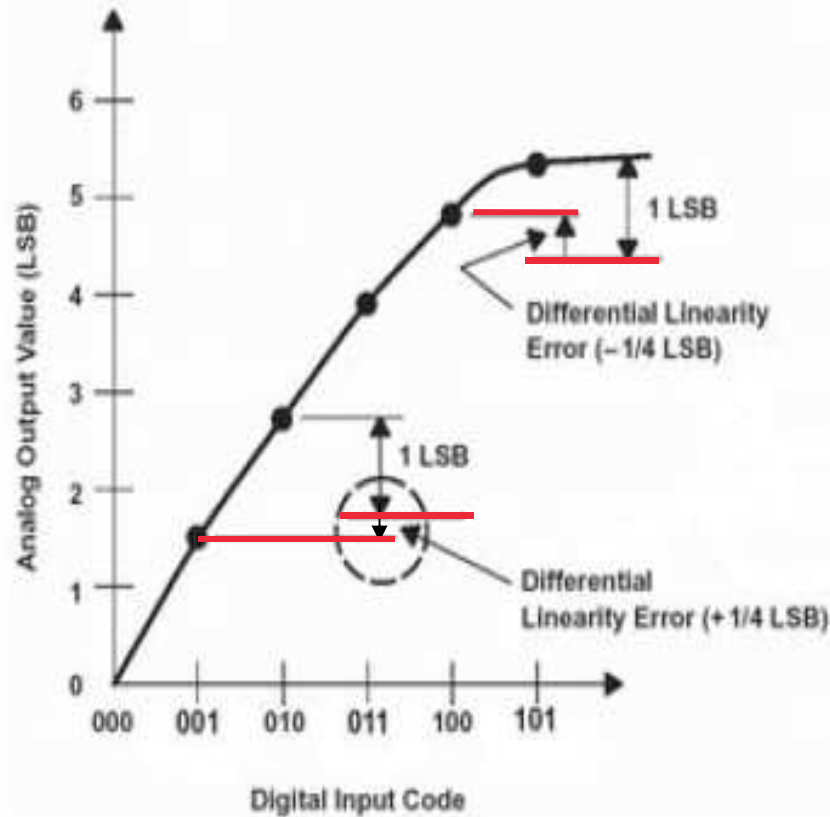
OFFSET AND GAIN ERROR



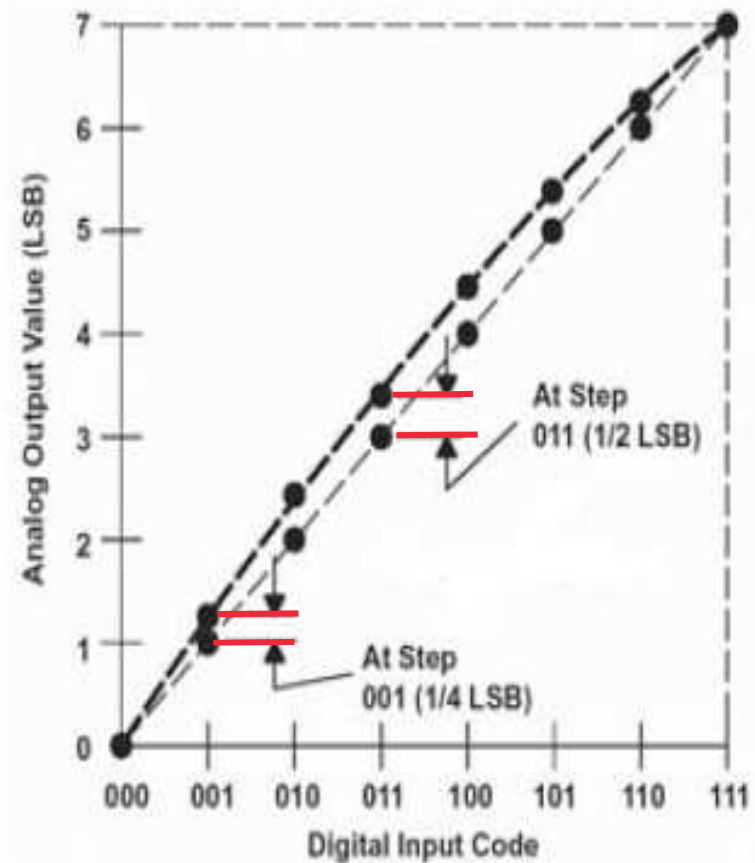
DAC differential and integral non-linearity

To determine DAC DNL and INL, the offset and gain errors must be eliminated by connecting the endpoints of the transfer characteristic with a straight line, and deriving the ideal values based on the non-ideal endpoints.

DNL = analog step deviation from 1LSB



INL = analog output deviation from its ideal location

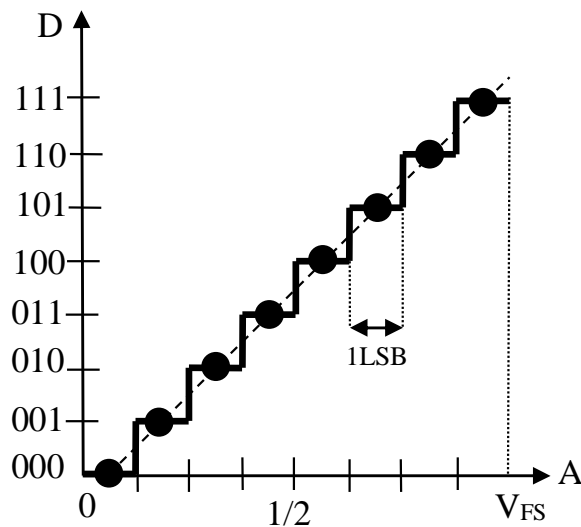


Which ADC characteristic? (I)

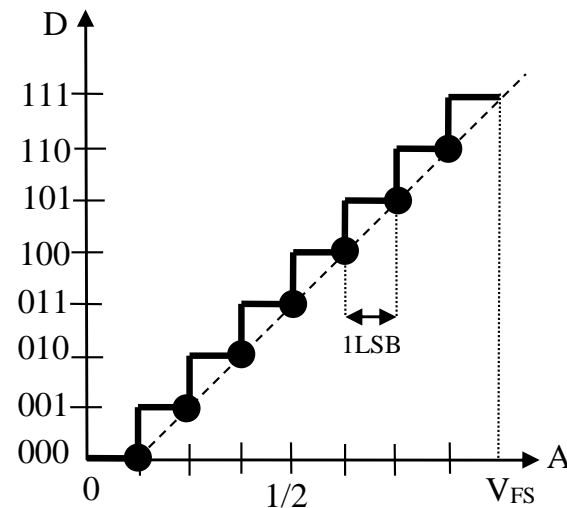
In a DAC, there is one output voltage for each digital input code, regardless of DNL or INL.

An ADC, on the other hand, does not have a unique voltage input corresponding to each output code—there is a small input voltage range equal to 1 LSB in width (for an ideal noiseless ADC) that will produce the same digital output code. This is called the *quantization uncertainty*, and it can be the source of confusion when specifying and measuring ADC static transfer characteristics.

CODE CENTERS
MEASURED INDIRECTLY

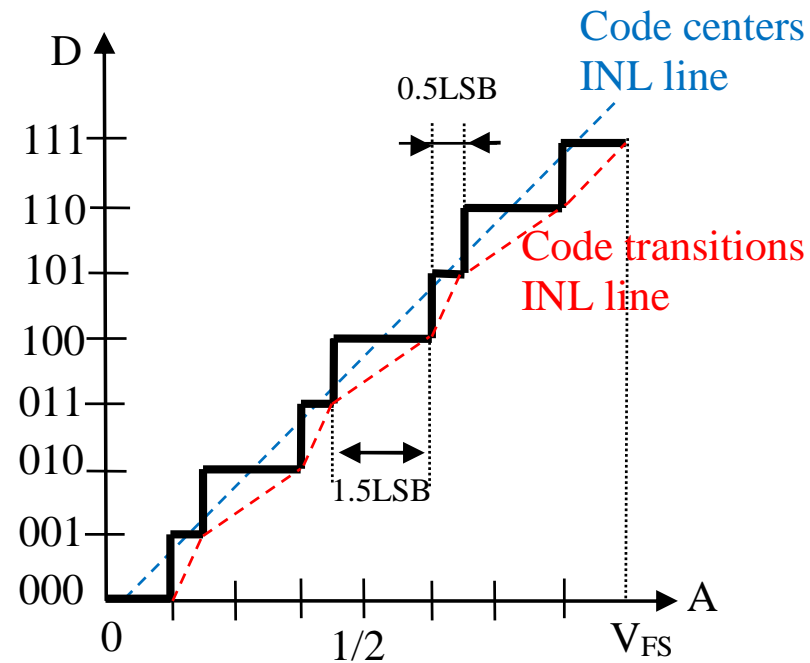


CODE TRANSITIONS
MEASURED DIRECTLY



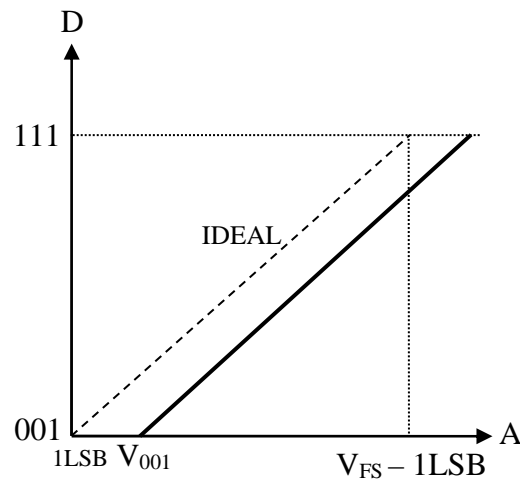
Which ADC characteristic? (II)

The code center method can lead to erroneous results → prefer the code transition method



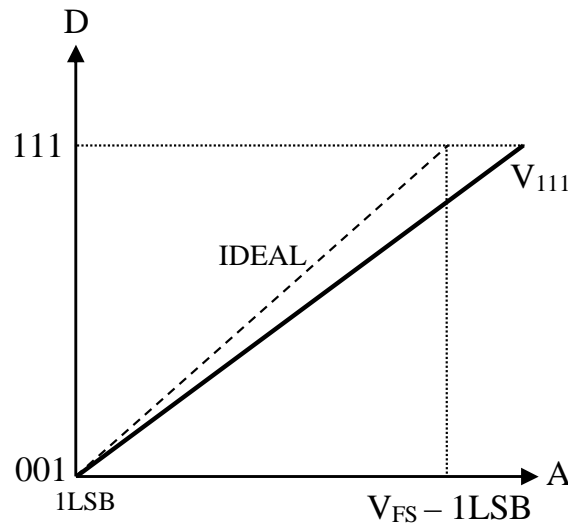
ADC offset and gain errors

OFFSET ERROR



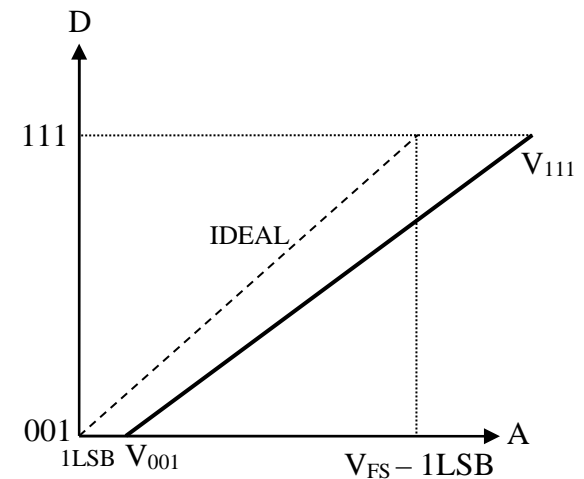
$$V_{OS} = V_{001} - 1\text{LSB}$$

GAIN ERROR



$$\text{GAIN ERROR (\%)} = 100 \left(\frac{V_{111} - 1\text{LSB}}{V_{FS} - 2\text{LSB}} - 1 \right)$$

OFFSET AND GAIN ERROR



$$V_{OS} = V_{001} - 1\text{LSB}$$

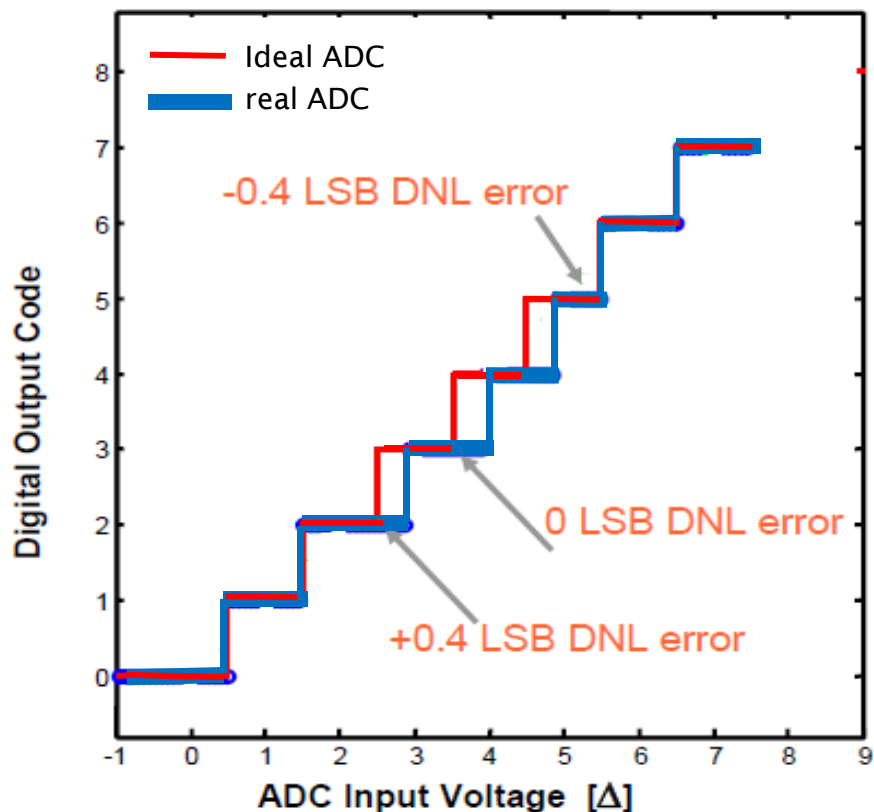
$$\text{GAIN ERROR (\%)} = 100 \left(\frac{V_{111} - V_{001}}{V_{FS} - 2\text{LSB}} - 1 \right)$$

Note: Code transitions INL used !

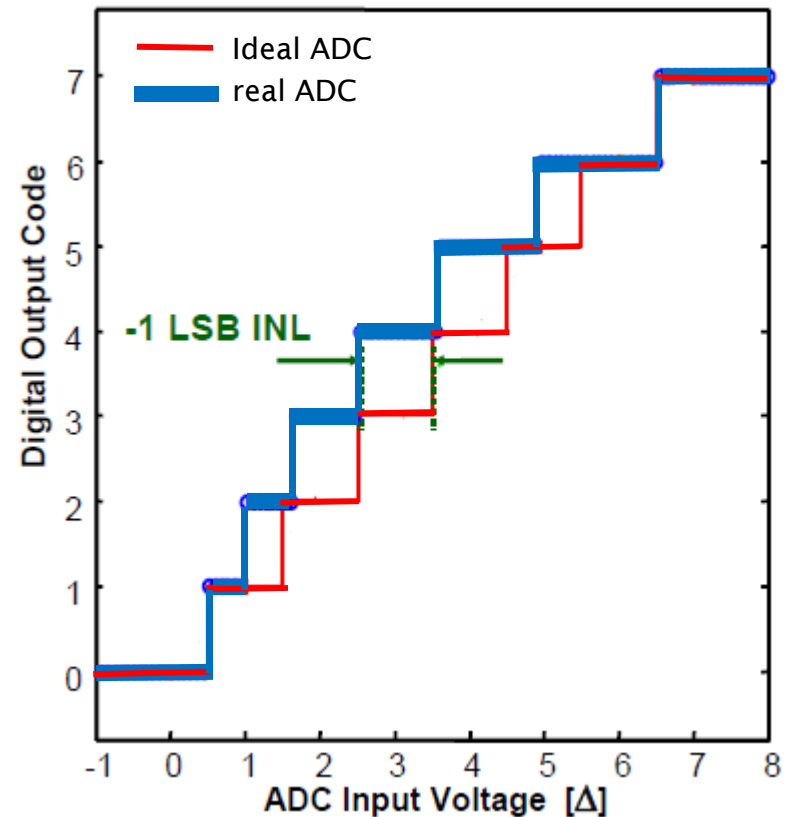
ADC differential and integral non-linearity

To determine ADC DNL and INL, the offset and gain errors must be eliminated by connecting the endpoints of the transfer characteristic with a straight line, and deriving the ideal values based on the non-ideal endpoints.

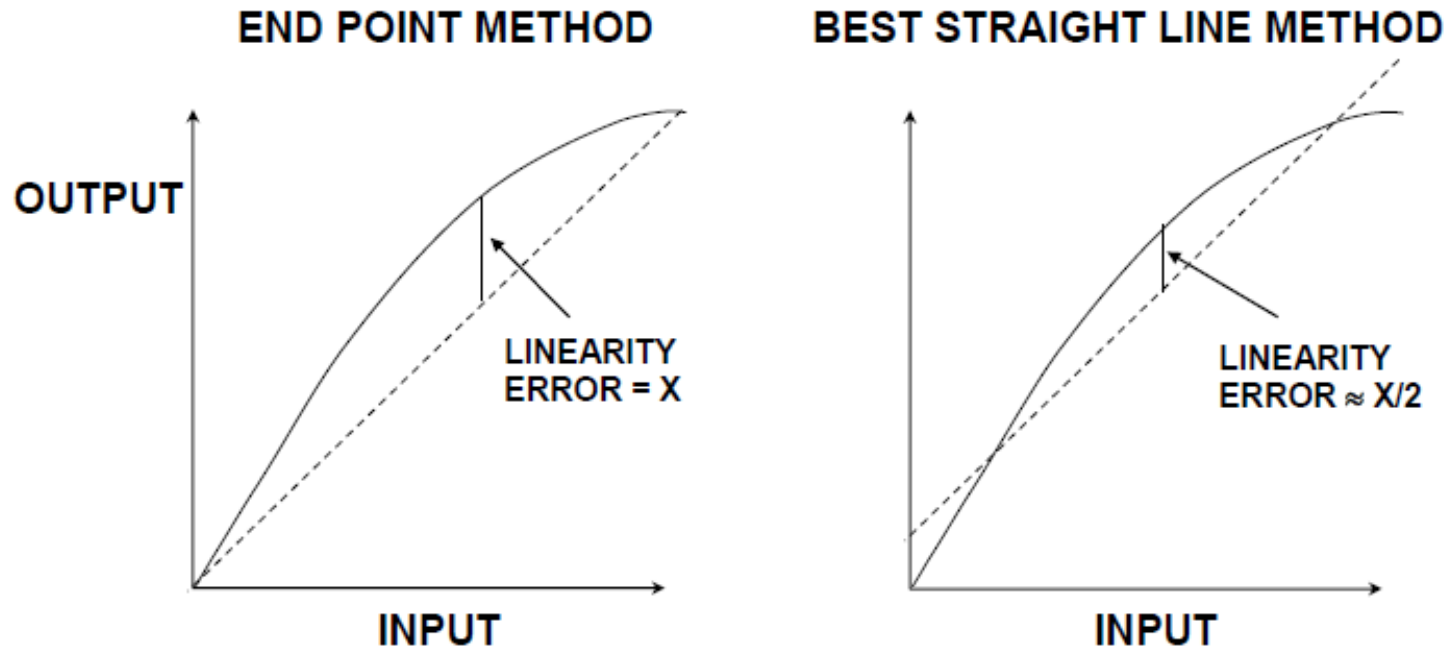
DNL = code transition width deviation from 1LSB



INL = code transition deviation from its ideal location



DAC and ADC: which INL?



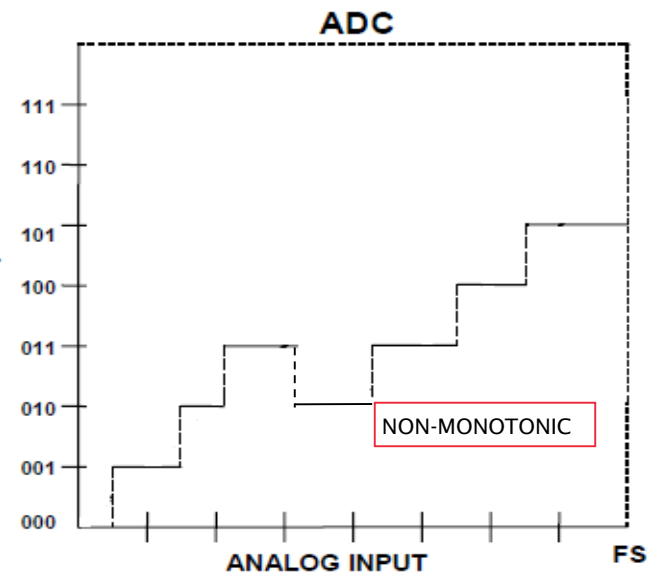
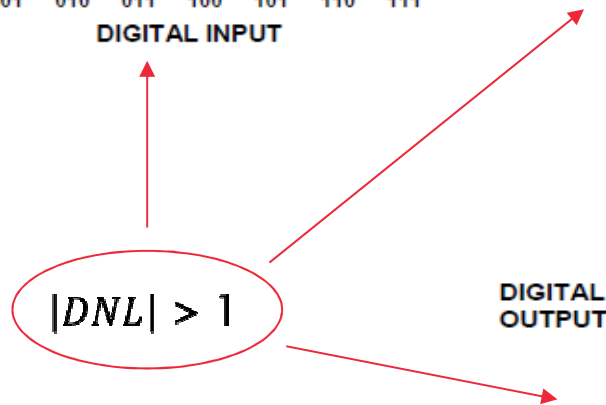
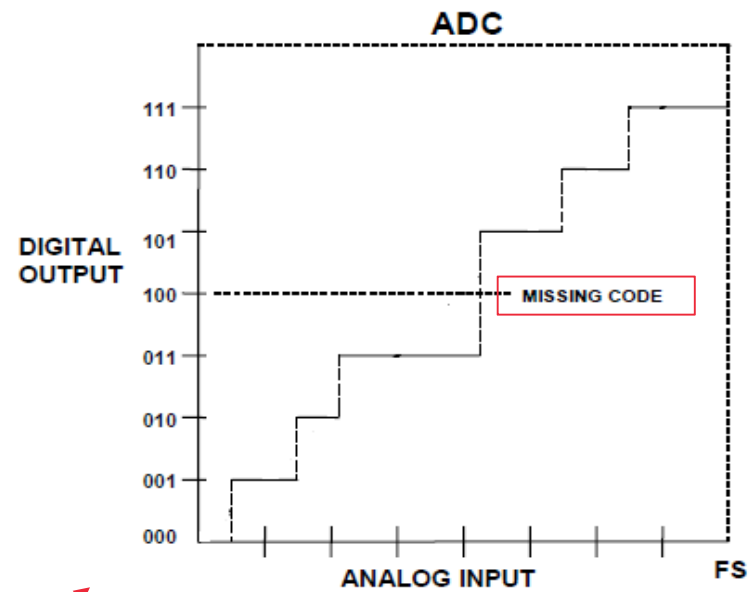
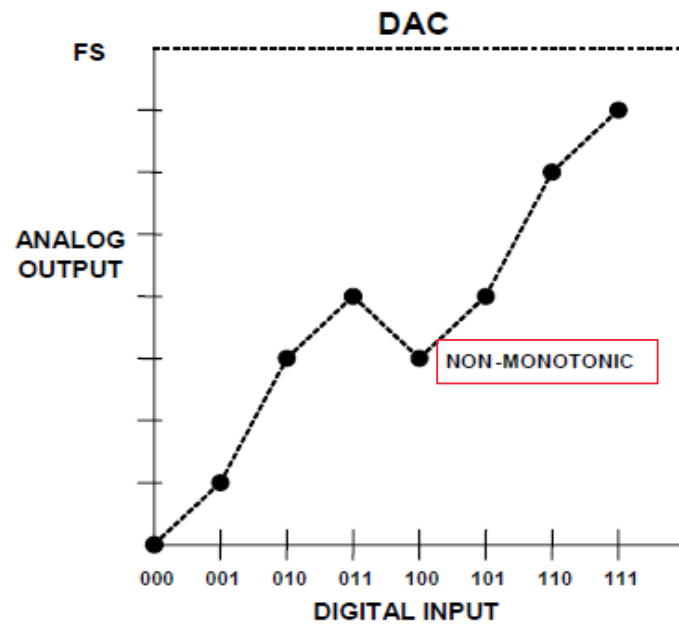
Since error budget (THD, IMD) depends on deviation from the ideal transfer characteristic, NOT from some arbitrary 'best-fit', end-to-end method is the only useful INL measurement.

Best-fit method gives a lower value of INL (even a factor of two) producing impressive data sheet, but it is not useful for error budget analysis.



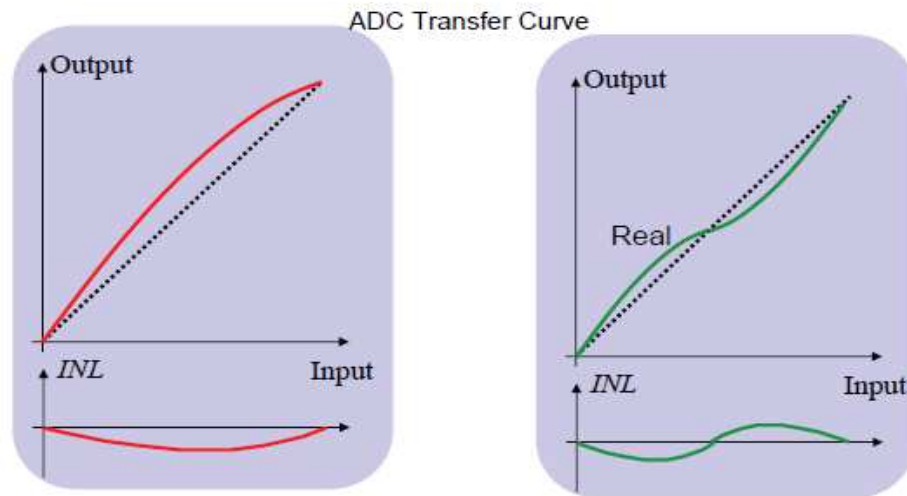
Use end-to-end method

Non-monotonicity and missing codes



Relationship between INL and SDR/SFDR (I)

Nature of harmonics depend on "shape" of INL curve.



Quadratic shaped
transfer function



even order harmonics

Cubic shaped
transfer function



odd order harmonics

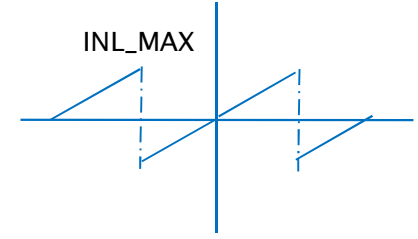
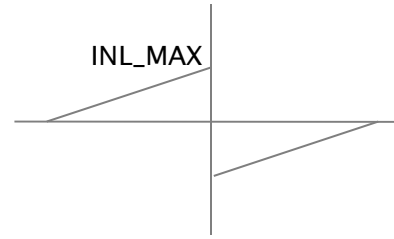
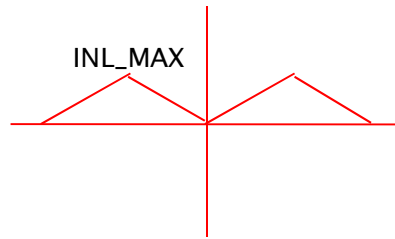
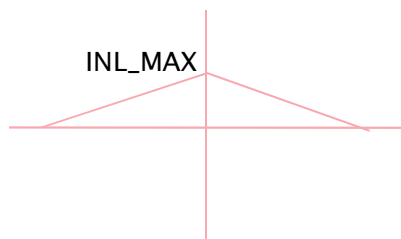
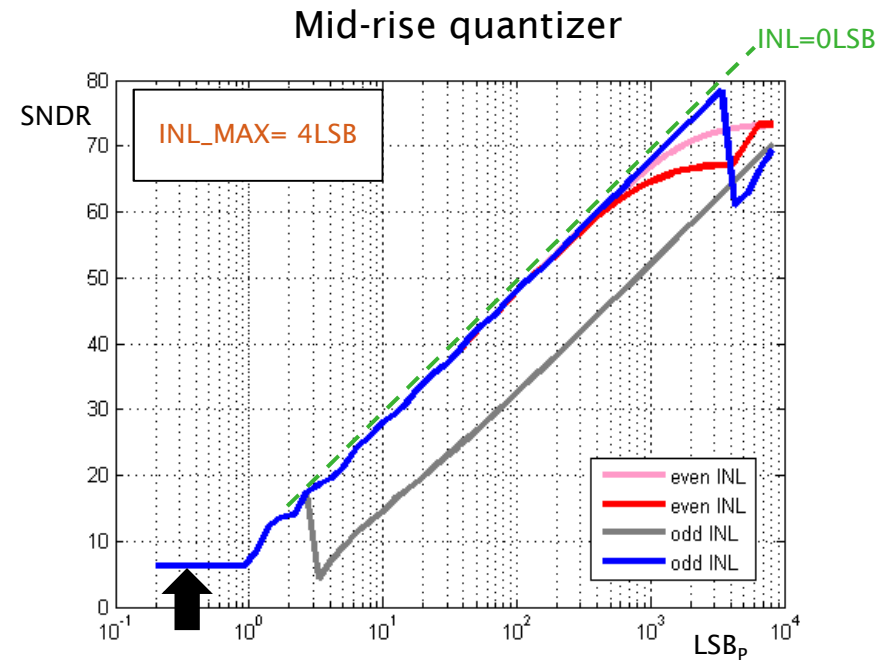
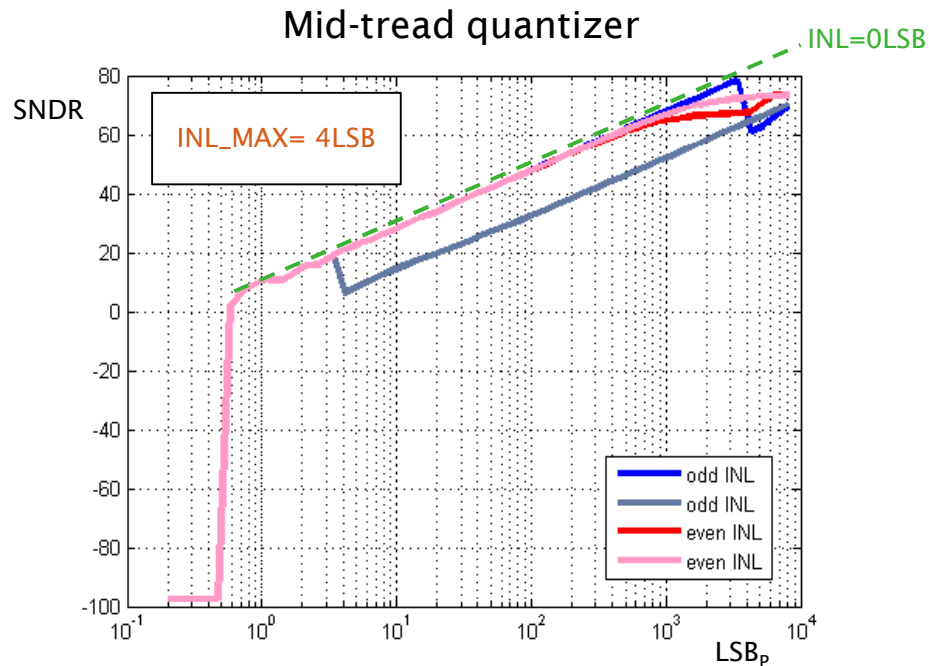
Rule of Thumb at FS signal

$$\text{SDR} \cong 20\log(2^N/\text{INL})$$

➔ Beware, this is only true under the same conditions at which the INL was taken, i.e. typically low input signal frequencies !

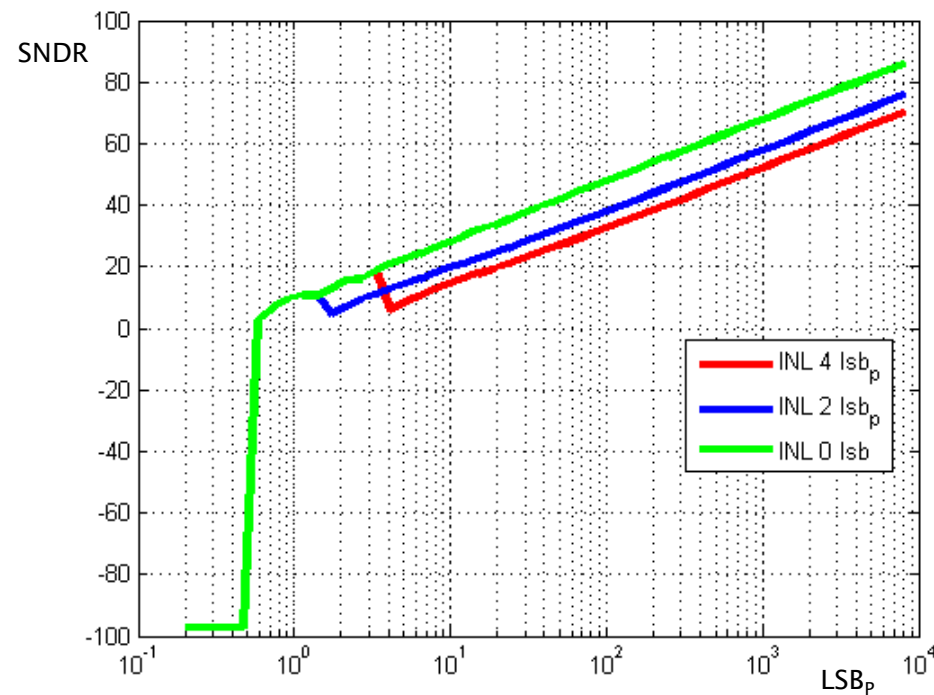
Relationship between INL and SDR/SNDR (II)

SNDR depends on quantizer type (mid-tread or mid-rise) and "shape" of INL curve.

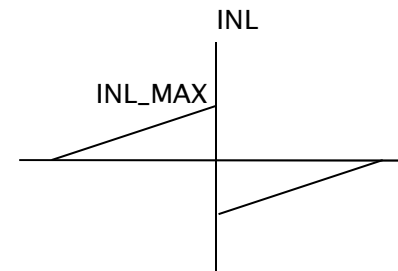


Relationship between INL and SDR/SNDR (III)

SNDR depends obviously on INL curve amplitude.



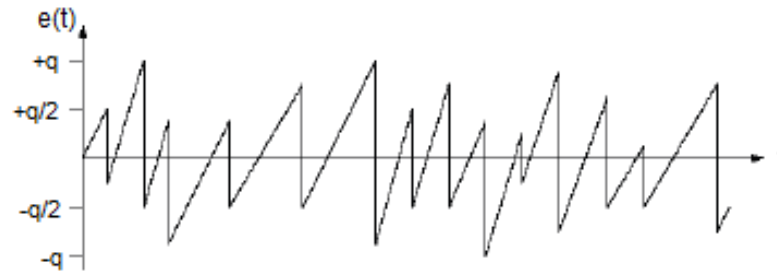
Mid-tread quantizer
with
odd INL



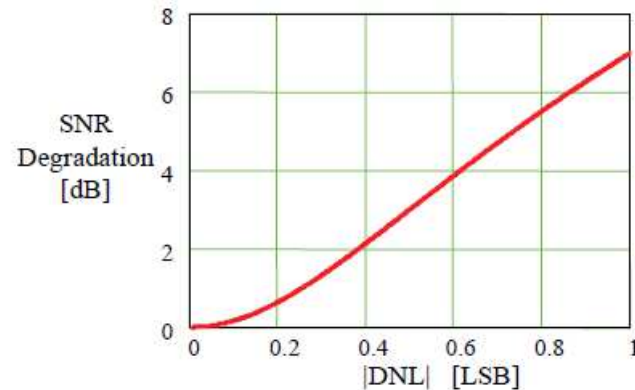
Relationship between DNL and SNR

Uniform quantization error pdf over the range $\pm q/2$ is assumed for an ideal quantizer.

DNL modifies the ideal step size q increasing the quantization error and degrading SNQR



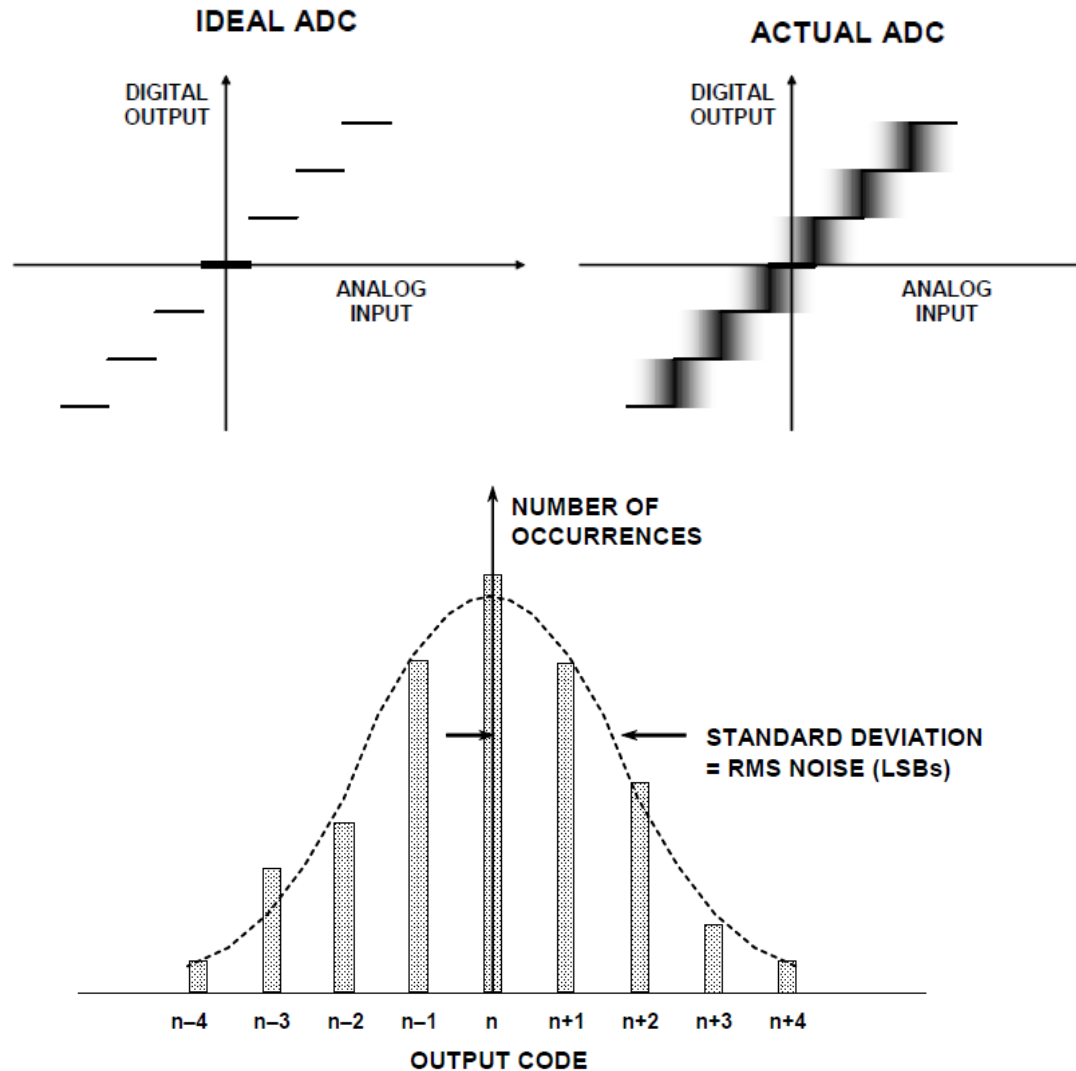
$$SQNR = \frac{\frac{1}{2} \left(\frac{2^N q}{2} \right)^2}{\frac{q^2}{12} + \frac{DNL^2}{3}}$$



➡ Beware, this is valid only for cases where DNL pdf is uniform and there are no missing codes !

Code transition noise (I)

Practical data converters have certain amount of code transition noise, usually named input-referred noise for ADCs and output-referred noise for DACs, that modifies its ideal transition region width.



Code transition noise (II)

- Data converters for time-domain applications (one-shot converters)

$$\text{Noise – Free Code Resolution} = \log_2 \left(\frac{2^N}{\max [1 ; \text{Peak – Peak Noise(LSB)}]} \right)$$

$$\text{Effective Resolution} = \log_2 \left(\frac{2^N}{\max [1 ; \text{RMS Noise(LSB)}]} \right)$$

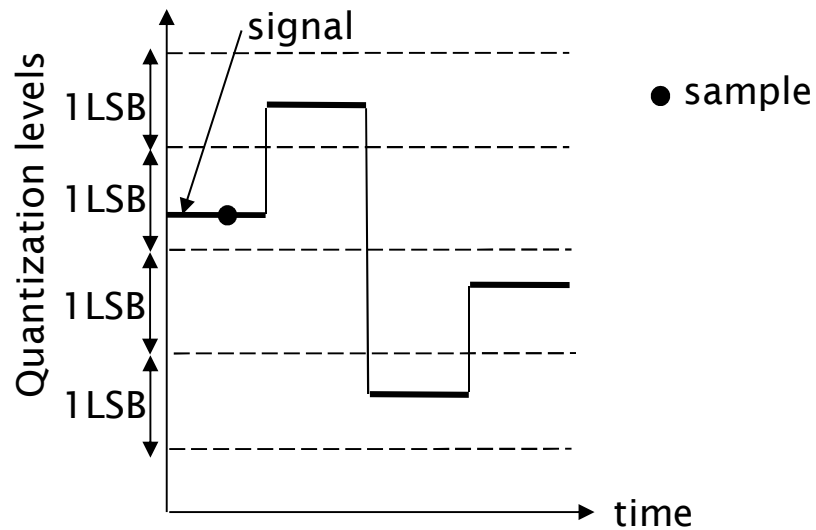
- Data converters for frequency-domain applications (signal converters)

$$\text{ENOB} = \frac{\text{SINAD} - 1.76}{6.02}$$

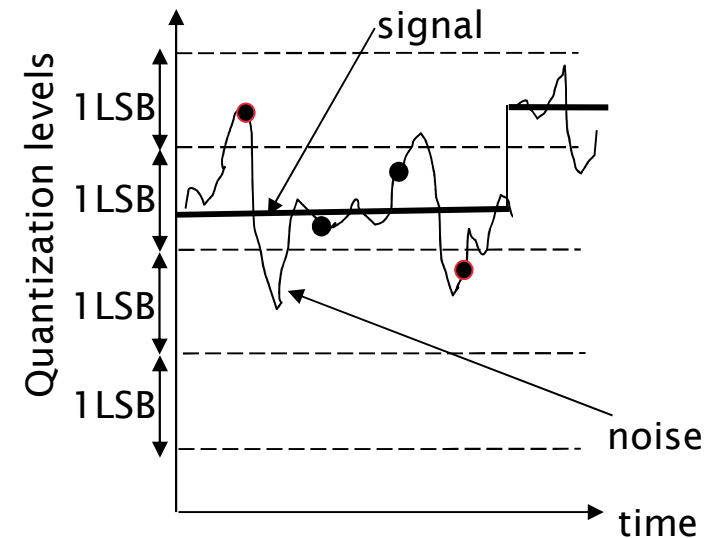
The less the noise the better the data converter performance? NOT ALWAYS TRUE!

Noise dither (I) : increases effective resolution

- Data converters for time-domain applications (one-shot converters)



Noiseless ADC



ADC with 0.5LSB noise/dither

No digital averaging: NFCR= N bits

4 samples digital averaging: NFCR= N bits

16 samples digital averaging: NFCR= N bits

NFCR= N-1 bits

NFCR= N bits

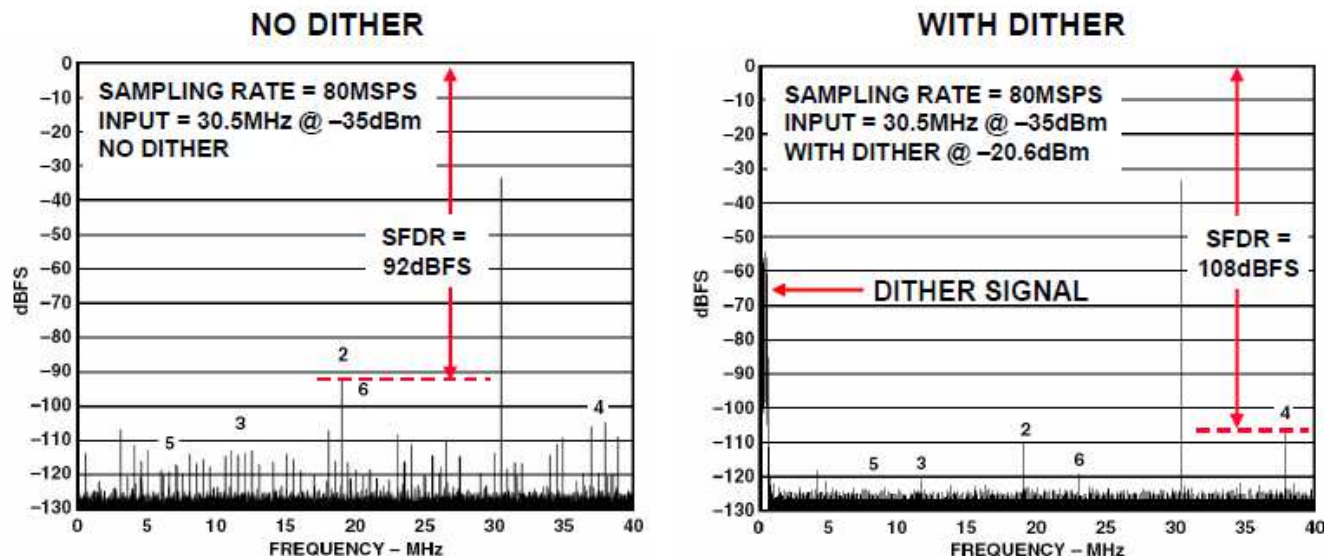
NFCR= N+1 bits

Digital averaging can increase resolution at the expense of the conversion rate.

Noise dither (II) : increases SFDR

- Data converters for frequency-domain applications (signal converters)

Example: 14b real ADC , f_s/f_{in} irrational number



Dither decorrelates the sinusoid and the quantization noise, and helps smoothing out the DNL errors (*) resulting in higher SFDR at the expense of an increase of the noise floor (reduces the dynamic range)!

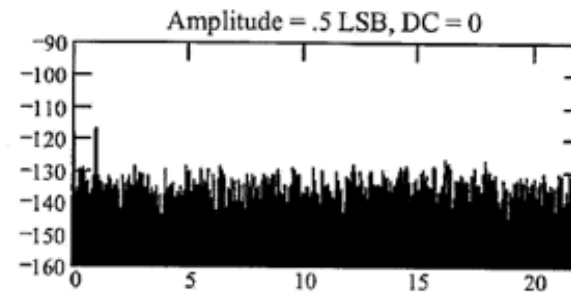
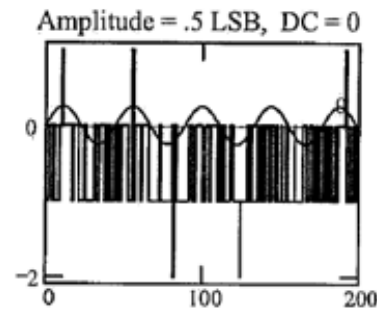
(*) for example, suppose a large DNL error leads to a missing code at quantization level 'k', the average of the two adjacent codes 'k-1' and 'k+1' is equal to 'k'.

Noise Dither (III) : randomizes transfer function

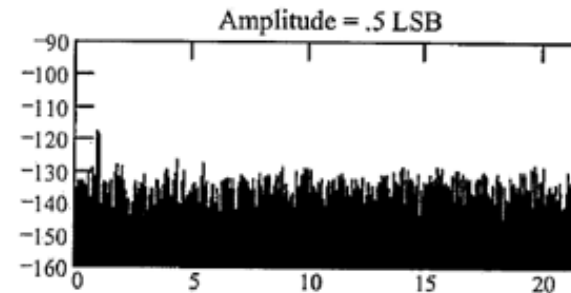
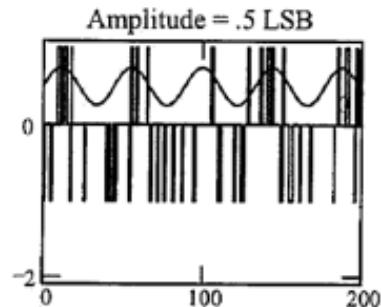
- Data converters for frequency-domain applications (signal converters)

Example: 16b ideal ADC

No offset
0.5LSB dither



offset 0.5 LSB
0.5LSB dither



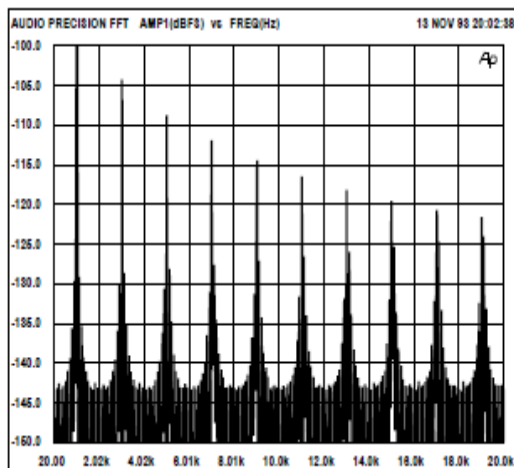
Distortion and noise modulation can be eliminated or strongly reduced by adding a small amount (<1 LSB) of dither to the signal prior to quantization.

Dither allows to distinguish signal 30dB lower than ADC resolutions at the expense of an increases of the noise floor (reduces the dynamic range)!

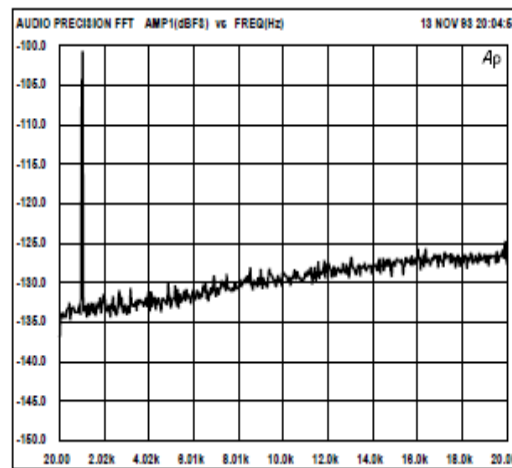
Noise dither (IV) : randomizes transfer function

- Data converters for frequency-domain applications (signal converters)

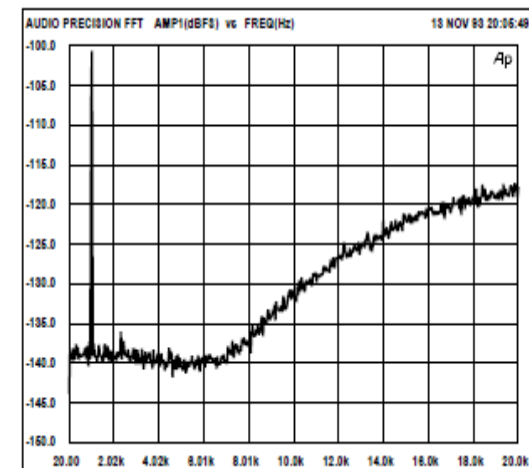
Example: 16b ideal DAC , 24b digital tone input



Truncation only



HP dither and truncation



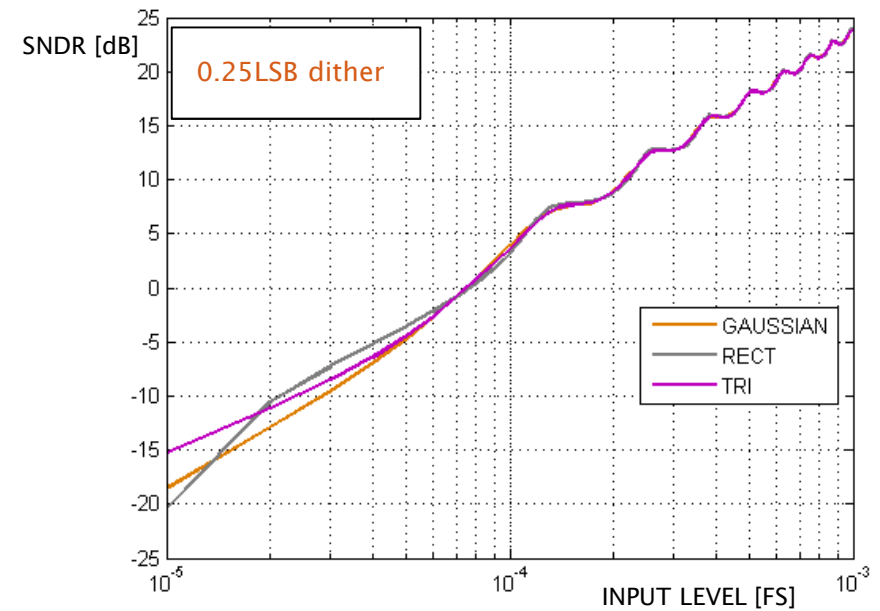
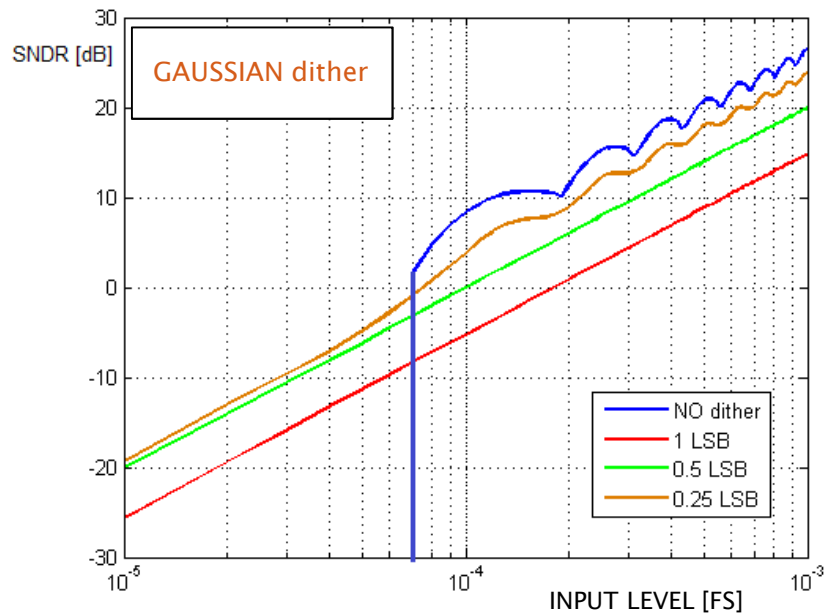
NS dither and truncation

Dither allows to distinguish signal 30dB lower than DAC resolutions at the expense of an increases of the noise floor (reduces the dynamic range)!

The general consensus is that the advantage of using dither outweighs its disadvantage.

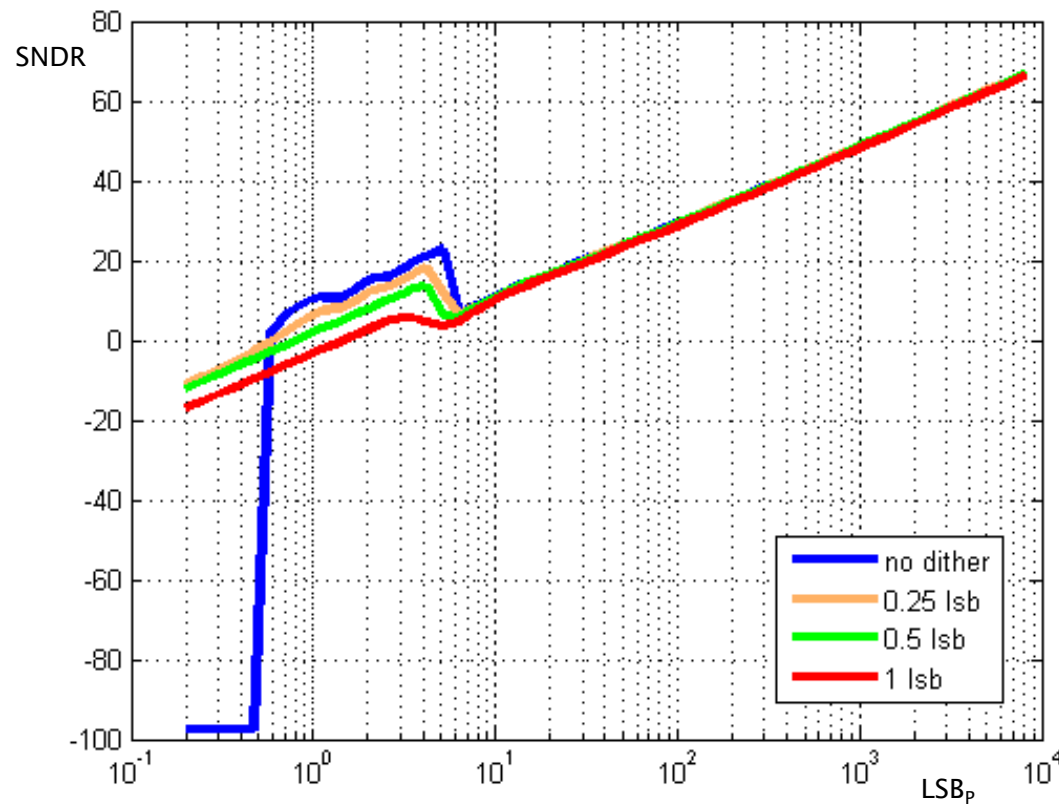
Dither level and pdf

Example: 14b ideal ADC

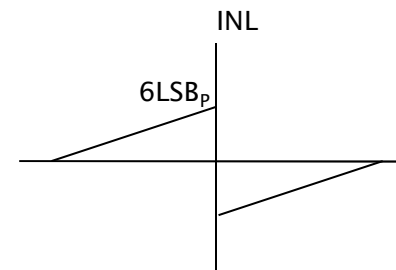


Dither vs INL

Example: 14b ADC with 6 LSB INL



Mid-tread quantizer
with
odd INL



Dithering is not effective in reducing the harmonic distortion !!!

Non-subtractive vs subtractive dither

- Previous results/considerations were based on the use of non-subtractive dither.
- If the dither is subtracted from the output, as in the case of subtractive dither, perfect decorrelation of quantization noise from the input signal is possible with lower dynamic range penalty.
- However, due to the difficulty of perfectly reproducing and subtracting a small noise dither, the effect of using subtractive dithering is reduced in practice.

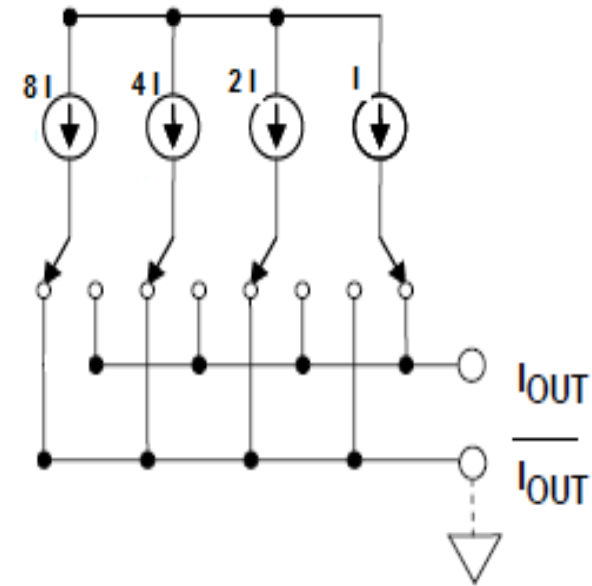
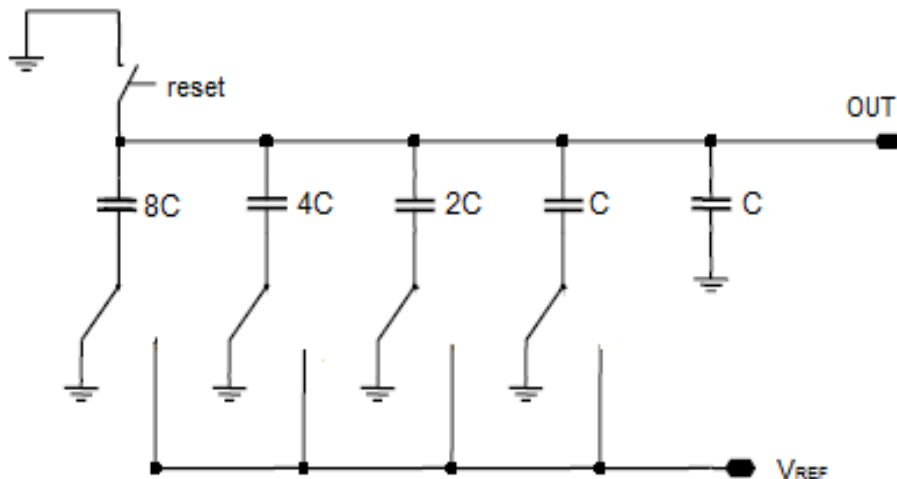
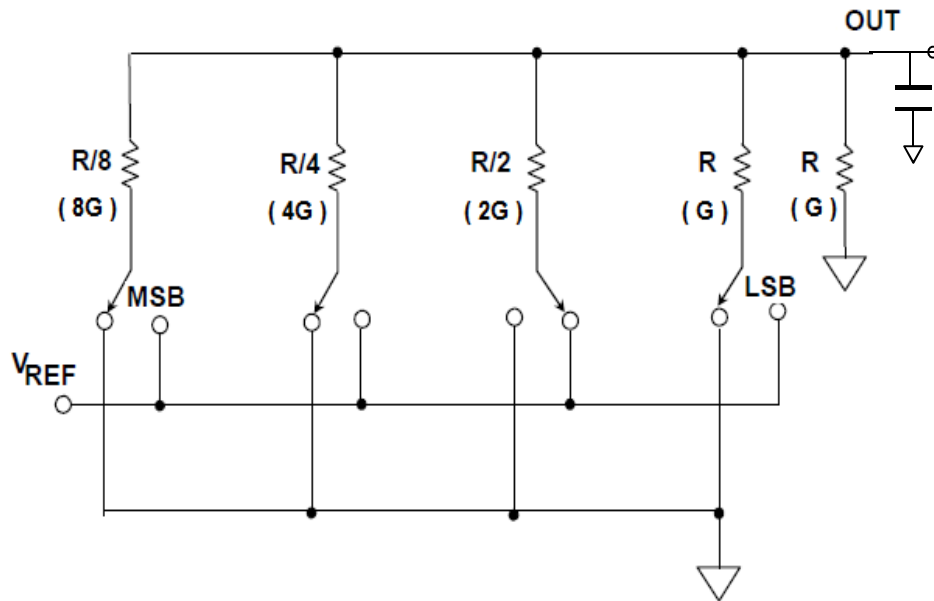
Data converter design

Basic data converter (DAC) typologies

Basic data converter (DAC) typologies

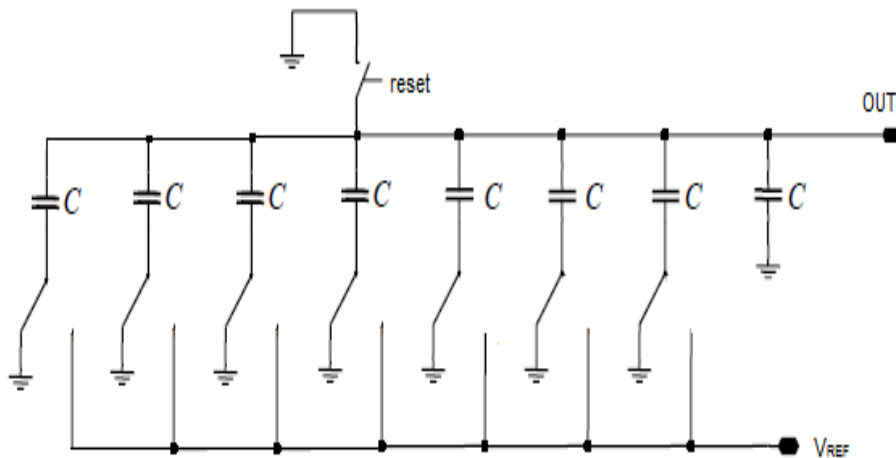
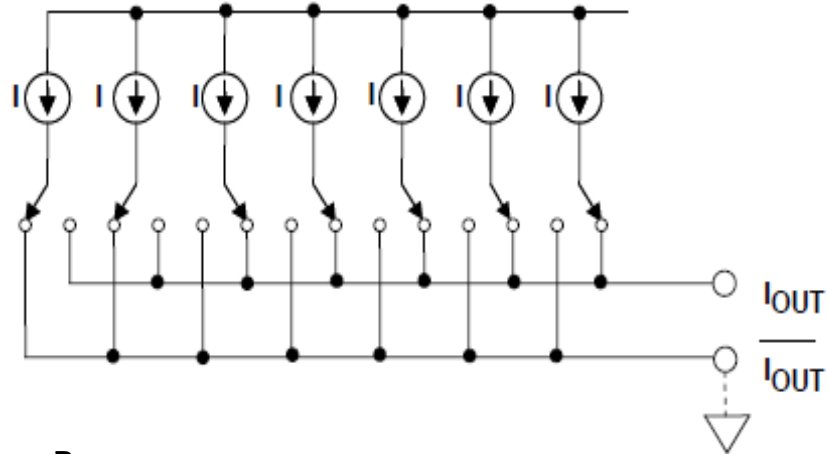
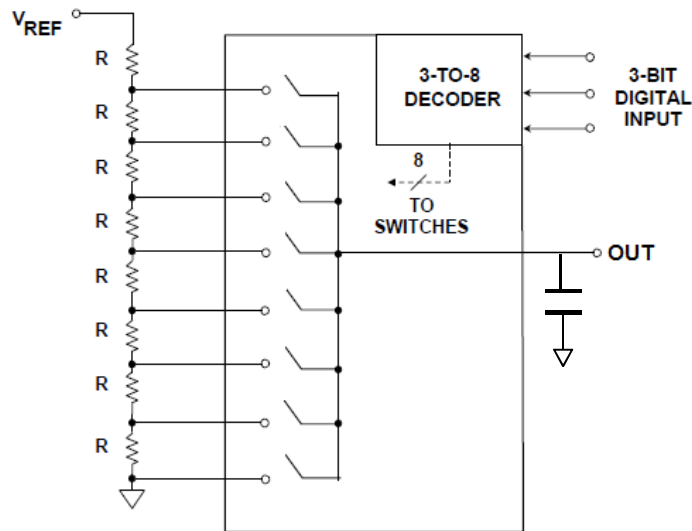
- Resistors → voltage domain
- Transistors → current domain
- Capacitors → charge domain

Binary-weighted DAC



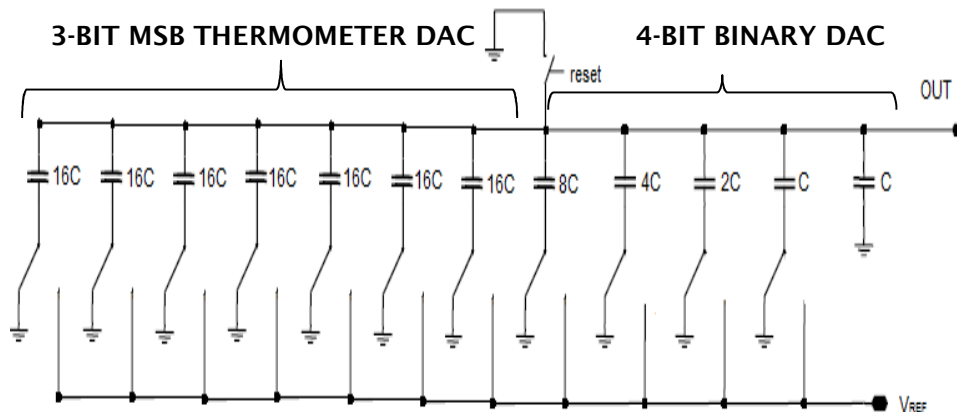
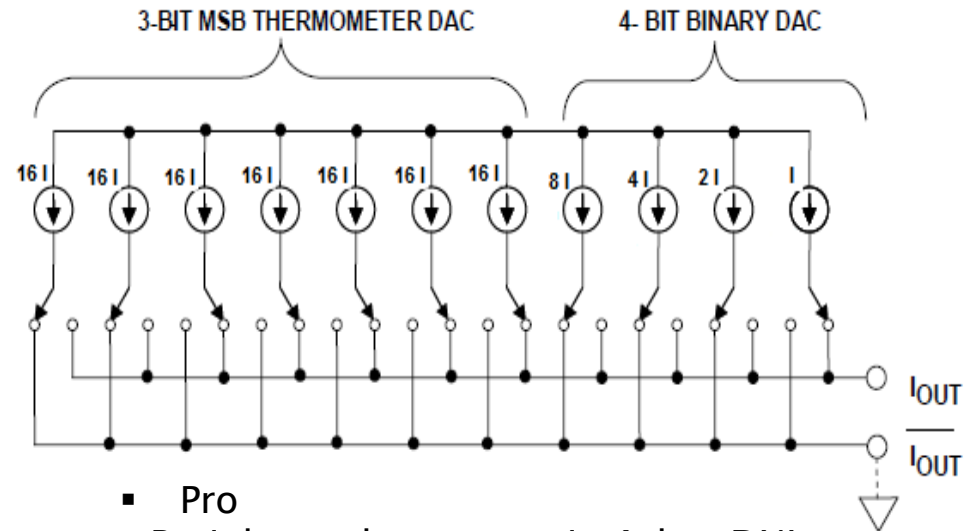
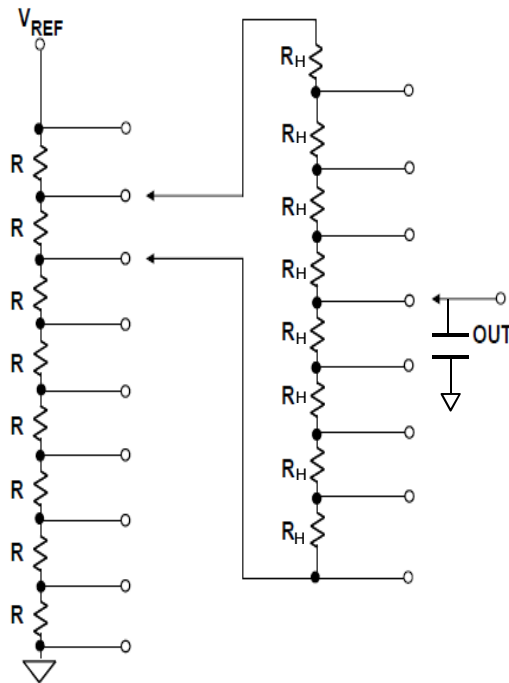
- Pro
 - R , I , C_{MOM} : compatible with std digital processes
 - I : fast speed if output node is low impedance
 - C : fast speed if reference node is low impedance
 - C : low power \rightarrow caps does not dissipate dc power
 - R : fast settling time $\rightarrow \tau = RC/2^N$
- Cons
 - about 2^N unit components, N switches for N bits \rightarrow large area for large N
 - R : max power dissipation $\sim V_{ref}^2 / (R/2^{N-2}) \rightarrow$ tradeoff between speed and power dissipation
 - C_{MIM} : not compatible with std digital processes
 - linearity : not inherently monotonic
 - linearity $\rightarrow R$, I , and C present high INL for $N > 10b$, $12b$, and $14b$, respectively.

Thermometer DACs



- Pro
 - inherently monotonic \rightarrow low DNL
 - R, I, C_{MOM} : compatible with std digital processes
 - I : fast speed if output node is low impedance
 - C : fast speed if reference node is low impedance
 - C : low power \rightarrow caps does not dissipate dc power
- Cons
 - about 2^N unit components and switches for N bits \rightarrow large area for large N
 - R : high settling time for large N $\rightarrow \tau_{max} \sim 2^N RC/4$
 - R : power dissipation $\sim V_{ref}^2 / (2^N R) \rightarrow$ tradeoff between speed and power dissipation
 - C_{MIM} : not compatible with std digital processes
 - linearity \rightarrow R, I, and C present high INL for N > 10b, 12b, and 14b, respectively.

Segmented DAC



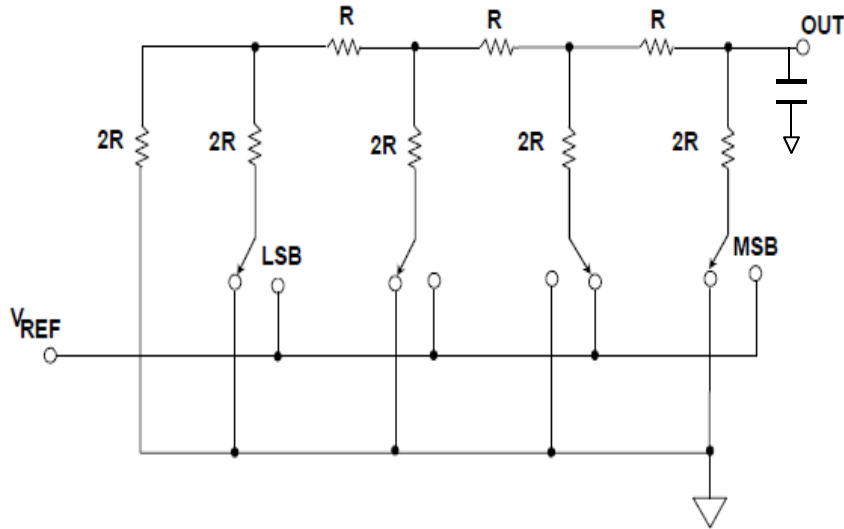
- Pro
 - R : inherently monotonic \rightarrow low DNL
 - R, I, C_{MOM} : compatible with std digital processes
 - I : fast speed if output node is low impedance
 - C : fast speed if reference node is low impedance
 - C : low power \rightarrow caps does not dissipate dc power
 - R : about $2 \times 2^{N/2}$ components for N bits

- Cons
 - I, C : about 2^N unit components and switches for N bits \rightarrow large area for large N
 - R : larger R_H values less loading, but lower speed
 - R : power dissipation $\sim V_{ref}^2 / (2^{N/2} R) \rightarrow$ tradeoff between speed and power dissipation
 - C_{MIM} : not compatible with std digital processes
 - I, C : linearity \rightarrow not inherently monotonic
 - linearity \rightarrow R, I, and C present high INL for N > 10b, 12b, and 14b, respectively.

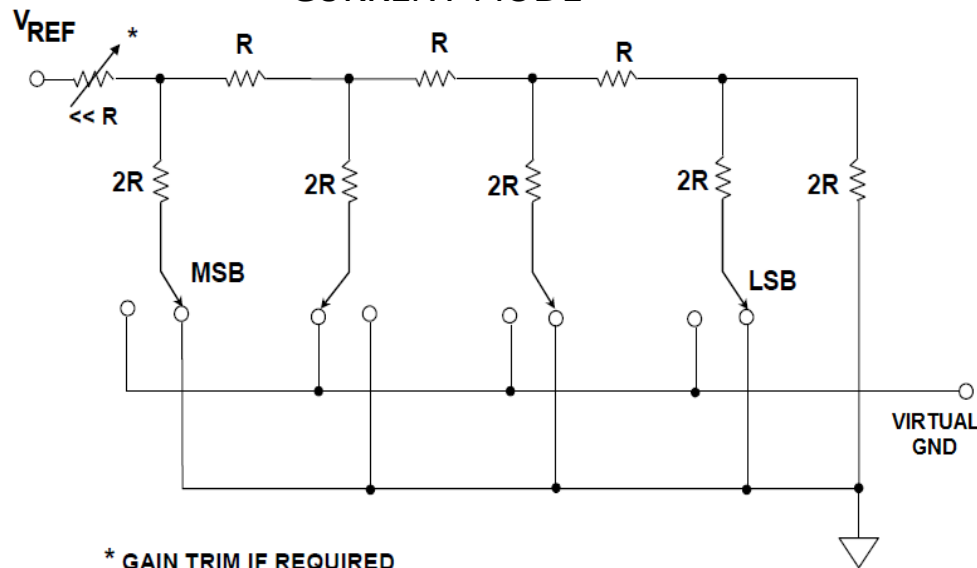
- Use buffer to prevent main string loading, but:
 - buffer offset can destroy monotonicity
 - tradeoff buffer consumption and DAC speed

R-2R DAC

VOLTAGE MODE



CURRENT MODE



* GAIN TRIM IF REQUIRED

- Pro
 - fast settling time $\rightarrow \tau = RC$ with constant output impedance R in voltage mode; virtual GND in current mode
 - compatible with pure digital technologies
 - resolution can be increased w/o heavy increase in complexity $\rightarrow 3N + 1$ unity resistors and N switches
- Cons
 - power dissipation $\sim V_{ref}^2 / 3R$ in voltage mode, V_{ref}^2 / R in current mode \rightarrow tradeoff between speed and power dissipation in voltage mode
 - linearity \rightarrow not inherently monotonic
 - linearity \rightarrow high INL for $N > 10b$

INL/DNL: Effect of short-range mismatches

INL/DNL: Effect of short-range mismatches

- INL non-linearity calculation
- DNL non-linearity calculation
- INL statistical calculation
- DNL statistical calculation
- Matching requirements
- INL/DNL improvements

INL and DNL non-linearity calculation

Consider a binary-weighted architecture as a starting point for data converter INL/DNL analysis.

If ε_m is the absolute error of the component representing the m^{th} bit of a binary-weighted ADC or DAC converter:

Binary-weighted output bit value: $b_m = 2^m + \varepsilon_m$

Full-scale value B: $B = \sum_{m=0}^{n-1} (2^m + \varepsilon_m) = 2^n - 1 + \sum_{m=0}^{n-1} \varepsilon_m$

Average step-size S: $S = \frac{B}{2^n - 1} = 1 + \frac{\sum_{m=0}^{n-1} \varepsilon_m}{2^n - 1}$

Linearity error of the k th bit: $INL_k = 2^k \cdot \left(1 + \frac{\sum_{m=0}^{n-1} \varepsilon_m}{2^n - 1} \right) - (2^k + \varepsilon_k)$

INL non-linearity calculation

Considering the 1st major bit (mid code) transition, i.e. k=n-1:

$$\begin{aligned}
 INL_{n-1} &= 2^{n-1} \cdot \left(1 + \frac{\sum_{m=0}^{n-1} \epsilon_m}{2^n - 1} \right) - (2^{n-1} + \epsilon_{n-1}) = \left(\frac{2^{n-1}}{2^n - 1} \cdot \sum_{m=0}^{n-1} \epsilon_m \right) - \epsilon_{n-1} = \frac{2^{n-1}}{2^n - 1} \cdot \left(\sum_{m=0}^{n-2} \epsilon_m + \epsilon_{n-1} \right) - \epsilon_{n-1} = \\
 &= \frac{2^{n-1}}{2^n - 1} \cdot \sum_{m=0}^{n-2} \epsilon_m + \frac{2^{n-1} - 2^n + 1}{2^n - 1} \cdot \epsilon_{n-1} = \frac{1}{2^n - 1} \left[2^{n-1} \cdot \sum_{m=0}^{n-2} \epsilon_m + (1 - 2^{n-1}) \cdot \epsilon_{n-1} \right]
 \end{aligned}$$

Considering the 2nd major bit transition, i.e. k=n-2:

$$\begin{aligned}
 INL_{n-2} &= 2^{n-2} \cdot \left(1 + \frac{\sum_{m=0}^{n-1} \epsilon_m}{2^n - 1} \right) - (2^{n-2} + \epsilon_{n-2}) = \left(\frac{2^{n-2}}{2^n - 1} \cdot \sum_{m=0}^{n-1} \epsilon_m \right) - \epsilon_{n-2} = \frac{2^{n-2}}{2^n - 1} \cdot \left(\sum_{m=0}^{n-3} \epsilon_m + \epsilon_{n-2} + \epsilon_{n-1} \right) - \epsilon_{n-2} = \\
 &= \frac{2^{n-2}}{2^n - 1} \cdot \left(\sum_{m=0}^{n-3} \epsilon_m + \epsilon_{n-1} \right) + \frac{2^{n-2} - 2^n + 1}{2^n - 1} \cdot \epsilon_{n-2} = \frac{1}{2^n - 1} \left[2^{n-2} \cdot \left(\sum_{m=0}^{n-3} \epsilon_m + \epsilon_{n-1} \right) + (1 - 2^n + 2^{n-2}) \cdot \epsilon_{n-2} \right]
 \end{aligned}$$

Considering the 3rd major bit transition, i.e. k=n-3:

$$\begin{aligned}
 INL_{n-3} &= 2^{n-3} \cdot \left(1 + \frac{\sum_{m=0}^{n-1} \epsilon_m}{2^n - 1} \right) - (2^{n-3} + \epsilon_{n-3}) = \left(\frac{2^{n-3}}{2^n - 1} \cdot \sum_{m=0}^{n-1} \epsilon_m \right) - \epsilon_{n-3} = \frac{2^{n-3}}{2^n - 1} \cdot \left(\sum_{m=0}^{n-4} \epsilon_m + \epsilon_{n-3} + \epsilon_{n-2} + \epsilon_{n-1} \right) - \epsilon_{n-3} = \\
 &= \frac{2^{n-3}}{2^n - 1} \cdot \left(\sum_{m=0}^{n-4} \epsilon_m + \epsilon_{n-2} + \epsilon_{n-1} \right) + \frac{2^{n-3} - 2^n + 1}{2^n - 1} \cdot \epsilon_{n-3} = \frac{1}{2^n - 1} \left[2^{n-3} \cdot \left(\sum_{m=0}^{n-4} \epsilon_m + \epsilon_{n-2} + \epsilon_{n-1} \right) + (1 - 2^n + 2^{n-3}) \cdot \epsilon_{n-3} \right]
 \end{aligned}$$

DNL non-linearity calculation

The worst DNL performance is surely at the 1st major bit (mid code) transition, i.e. $k=n-1$, since the MSB device (current source, capacitor, resistor) needs to be matched to the sum of all other ones:

$$\begin{aligned}
 DNL_{n-1} &= b_{n-1} - \left(\sum_{m=0}^{n-2} b_m \right) - \left(1 + \frac{\sum_{m=0}^{n-1} \epsilon_m}{2^n - 1} \right) = 2^{n-1} + \epsilon_{n-1} - \left(2^{n-1} - 1 + \sum_{m=0}^{n-2} \epsilon_m \right) - 1 - \frac{\sum_{m=0}^{n-1} \epsilon_m}{2^n - 1} = \epsilon_{n-1} - \sum_{m=0}^{n-2} \epsilon_m - \frac{\sum_{m=0}^{n-1} \epsilon_m}{2^n - 1} = \\
 &= \epsilon_{n-1} - \sum_{m=0}^{n-2} \epsilon_m - \frac{\sum_{m=0}^{n-2} \epsilon_m + \epsilon_{n-1}}{2^n - 1} = \frac{1}{2^n - 1} \left[(2^n - 1 - 1) \cdot \epsilon_{n-1} - (2^n - 1 + 1) \cdot \sum_{m=0}^{n-2} \epsilon_m \right] = \frac{1}{2^n - 1} \left[(2^n - 2) \cdot \epsilon_{n-1} - 2^n \cdot \sum_{m=0}^{n-2} \epsilon_m \right]
 \end{aligned}$$

It can be notice that in case of binary-weighted architecture $|DNL_{n-1}| = 2 \cdot |INL_{n-1}|$

Statistical considerations

ε_m ($m=0\dots n-1$) are independent Gaussian random variables with zero mean, thus it is convenient to move to statistics.

- if σ_u is the 1-sigma error respect to the ideal value of the unit component U, the absolute mismatch is $\sigma = \sigma_u / U$.
- if σ_{u1-u2} is the 1-sigma error between two 'identical' unit components U, the relative mismatch is $\sigma_{1-2} = \sigma_{u1-u2} / U$.

- It follows

$$\sigma_{1-2} = \sqrt{2}\sigma$$

σ is useful for statistical INL and DNL calculations

σ_{1-2} is the one measured and reported in the DRM

INL statistical calculation

Considering the 1st major bit transition, i.e. $k=n-1$:

$$\sigma^2(INL_{n-1}) = \frac{\sigma^2}{(2^n - 1)^2} \left[(2^{n-1})^2 (2^{n-1} - 1) + 2^{n-1} (2^{n-1} - 1)^2 \right] = \frac{\sigma^2}{(2^n - 1)^2} \left[2^{n-1} (2^{n-1} - 1) (2^{n-1} + 2^{n-1} - 1) \right] = \sigma^2 \frac{2^{n-1} (2^{n-1} - 1)}{2^n - 1}$$

$$\longrightarrow \sigma(INL_{n-1}) = \sigma \sqrt{\frac{2^{n-1} (2^{n-1} - 1)}{2^n - 1}}$$

Considering the 2nd major bit transition, i.e. $k=n-2$:

$$\sigma^2(INL_{n-2}) = \frac{\sigma^2}{(2^n - 1)^2} \left[(2^{n-2})^2 (2^{n-2} - 1 + 2^{n-1}) + 2^{n-2} (1 - 2^n + 2^{n-2})^2 \right] = \frac{\sigma^2}{(2^n - 1)^2} \left[(2^{n-2})^2 (3 \cdot 2^{n-2} - 1) + 2^{n-2} (3 \cdot 2^{n-2} - 1)^2 \right] =$$

$$= \frac{\sigma^2}{(2^n - 1)^2} \left[2^{n-2} (3 \cdot 2^{n-2} - 1) (4 \cdot 2^{n-2} - 1) \right] = \sigma^2 \frac{2^{n-2} (3 \cdot 2^{n-2} - 1)}{2^n - 1}$$

$$\longrightarrow \sigma(INL_{n-2}) = \sigma \sqrt{\frac{2^{n-2} (3 \cdot 2^{n-2} - 1)}{2^n - 1}}$$

Considering the 3rd major bit transition, i.e. $k=n-3$:

$$\sigma^2(INL_{n-3}) = \frac{\sigma^2}{(2^n - 1)^2} \left[(2^{n-3})^2 (2^{n-3} - 1 + 2^{n-2} + 2^{n-1}) + 2^{n-3} (1 - 2^n + 2^{n-3})^2 \right] = \frac{\sigma^2}{(2^n - 1)^2} \left[(2^{n-3})^2 (7 \cdot 2^{n-3} - 1) + 2^{n-3} (7 \cdot 2^{n-3} - 1)^2 \right] =$$

$$= \frac{\sigma^2}{(2^n - 1)^2} \left[2^{n-3} (7 \cdot 2^{n-3} - 1) (8 \cdot 2^{n-3} - 1) \right] = \sigma^2 \frac{2^{n-3} (7 \cdot 2^{n-3} - 1)}{2^n - 1}$$

$$\longrightarrow \sigma(INL_{n-3}) = \sigma \sqrt{\frac{2^{n-3} (7 \cdot 2^{n-3} - 1)}{2^n - 1}}$$

INL statistical calculation (cont'd)

The previous INL formulae are valid whichever the value of n.

However, Nyquist data converters usually present $n \gg 1$ (usually from 6 to 16), resulting to:

$$\sigma(INL_{n-1}) \cong \sigma \frac{1}{2} \sqrt{2^n} = \sigma_{1-2} \frac{1}{2} \sqrt{2^{n-1}} \longrightarrow \sigma(INL_{MAX}) \cong \sigma \frac{1}{2} \sqrt{2^n} = \sigma_{1-2} \frac{1}{2} \sqrt{2^{n-1}}$$

$$\sigma(INL_{n-2}) \cong \sigma \frac{\sqrt{3}}{4} \sqrt{2^n} = \sigma_{1-2} \frac{\sqrt{3}}{4} \sqrt{2^{n-1}}$$

$$\sigma(INL_{n-3}) \cong \sigma \frac{\sqrt{7}}{8} \sqrt{2^n} = \sigma_{1-2} \frac{\sqrt{7}}{8} \sqrt{2^{n-1}}$$

•
•
•

$$\sigma(INL_1) \cong \sigma \frac{\sqrt{2^{n-1}} - 1}{2^{n-1}} \sqrt{2^n} \cong \sigma \frac{1}{\sqrt{2^{n-1}}} \sqrt{2^n} = \sigma \sqrt{2} = \sigma_{1-2}$$

DNL statistical calculation

Considering the 1st major bit transition, i.e. $k=n-1$:

$$\begin{aligned}\sigma^2(DNL_{n-1}) &= \frac{\sigma^2}{(2^n - 1)^2} \left[(2^n - 2)^2 2^{n-1} + (2^n)^2 (2^{n-1} - 1) \right] = \frac{\sigma^2}{(2^n - 1)^2} [2^{3n-1} + 2^{n+1} - 2^{2n+1} + 2^{3n-1} - 2^{2n}] = \frac{\sigma^2}{(2^n - 1)^2} [2^{3n} + 2^{n+1} - 2^{2n+1} - 2^{2n}] = \\ &= \frac{\sigma^2}{(2^n - 1)^2} [2^{2n} \cdot 2^n - 2^{2n} - 2^{n+1} \cdot 2^n + 2^{n+1}] = \frac{\sigma^2}{(2^n - 1)^2} [2^{2n}(2^n - 1) - 2^{n+1}(2^n - 1)] = \frac{\sigma^2}{(2^n - 1)^2} [2^{2n} - 2^{n+1}] = \sigma^2 \frac{2^{n+1}(2^{n-1} - 1)}{(2^n - 1)}\end{aligned}$$

$$\longrightarrow \sigma(DNL_{n-1}) = \sigma \sqrt{\frac{2^{n+1}(2^{n-1} - 1)}{2^n - 1}} = \sigma(DNL_{MAX})$$

Again notice that in case of binary-weighted architecture $\sigma(DNL_{n-1}) = 2 \cdot \sigma(INL_{n-1})$

In the case $n \gg 1$ (usually from 6 to 16), it results:

$$\sigma(DNL_{MAX}) \cong \sigma \sqrt{2^n} = \sigma_{1-2} \sqrt{2^{n-1}}$$

Matching requirements for INL and DNL

If INL_{spec} and DNL_{spec} are, respectively, the INL and DNL to be guaranteed at k-sigma in a data sheet, the max mismatch for the unit component is:

$$k\sigma(INL_{MAX}) \leq INL_{spec} \Rightarrow k \cdot \sigma_{1-2} \frac{1}{2} \sqrt{2^{n-1}} \leq INL_{spec} \Rightarrow \sigma_{1-2} \leq \frac{2 \cdot INL_{spec}}{k \sqrt{2^{n-1}}}$$

$$k\sigma(DNL_{MAX}) \leq DNL_{spec} \Rightarrow k \cdot \sigma_{1-2} \sqrt{2^{n-1}} \leq DNL_{spec} \Rightarrow \sigma_{1-2} \leq \frac{DNL_{spec}}{k \sqrt{2^{n-1}}}$$

Warning: These formulae have been used by many authors to design converters with high yield in mass production [5] even if they are not fully correct since they take into account only the probability P related to the (n-1)-th code $P(INL_{n-1}) < INL_{spec}$ or $P(DNL_{n-1}) < DNL_{spec}$, this probability being the most critical one, whereas they do not consider the probabilities of all the other i-th codes $P(INL_i) < INL_{spec}$ or $P(DNL_i) < DNL_{spec}$, with $i=0,1,2,..n-2$.

Matching requirements for INL and DNL (cont'd)

The exact formulae have been analytically presented in [6], but they can be more simply derived from the previous approximated ones by introducing two parameters $f_{INL}(n,k)$ and $f_{DNL}(n,k)$, where n is number of bits of the converter and k is the number of sigma :

$$\sigma_{1-2} \leq \frac{2 \cdot INL_{spec}}{k \sqrt{2^{n-1}}} \cdot \frac{1}{f_{INL}(n,k)}$$

k σ	8bits	10bits	12bits	14bits
1 σ	1.78	1.83	1.87	1.87
2 σ	1.31	1.35	1.36	1.36
3 σ	1.17	1.19	1.20	1.20
4 σ	1.10	1.12	1.13	1.13
5 σ	1.07	1.09	1.09	1.09

$$f_{INL}(n,k)$$

$$\sigma_{1-2} \leq \frac{DNL_{spec}}{k \sqrt{2^{n-1}}} \cdot \frac{1}{f_{DNL}(n,k)}$$

k σ	8bits	10bits	12bits	14bits
1 σ	1.25	1.25	1.25	1.25
2 σ	1.1	1.1	1.1	1.1
3 σ	1.0	1.0	1.0	1.0
4 σ	1.0	1.0	1.0	1.0
5 σ	1.0	1.0	1.0	1.0

$$f_{DNL}(n,k)$$

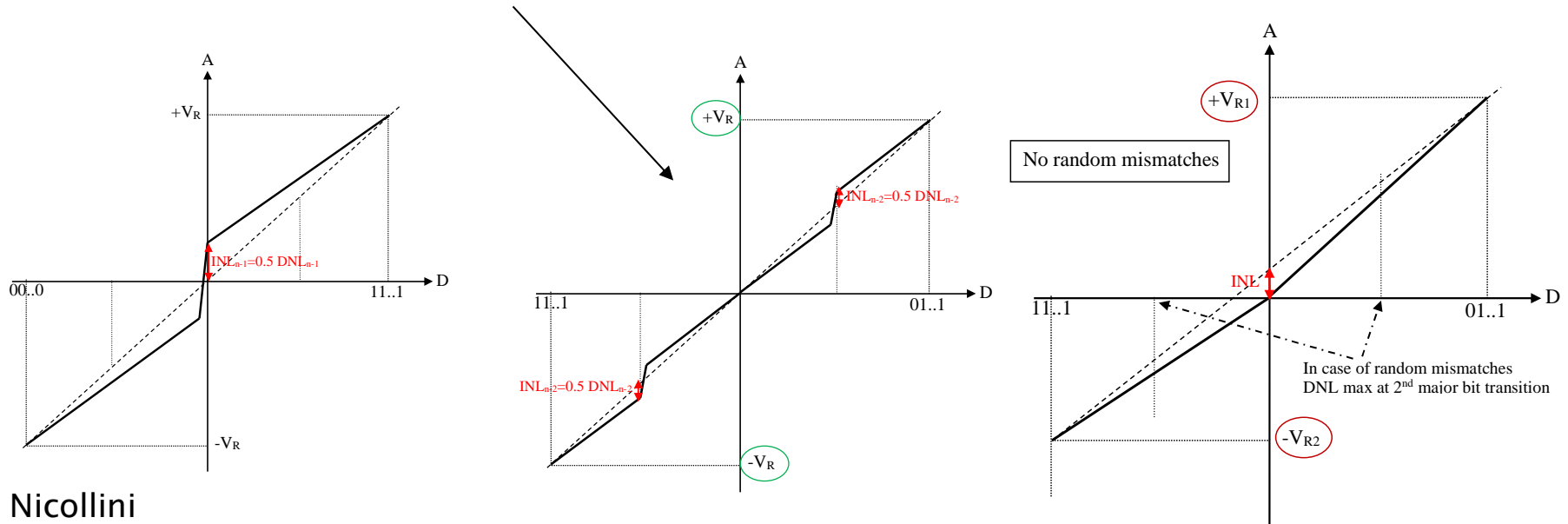
Conclusions: since at least a 4 σ design is the target for mass production yield, both $f_{INL}(n,k)$ and $f_{DNL}(n,k)$ are close to 1, thus the approximated formulae can be used.

INL and DNL improvements

- short-range (random) mismatches -

- increase data converter area \Rightarrow **INL** and **DNL** can be halved by increasing array area by a factor of 4 !
- use a differential structure \Rightarrow **INL** and **DNL** are reduced by $\sqrt{2}$ since array area is doubled !
- use sign-and-magnitude architecture (*) \Rightarrow **INL** and **DNL** can be halved by using same array area !
or **INL** and **DNL** are reduced by $\sqrt{2}$ with half array area !

(*) with equal analog full-scale values



DNL improvements

- short-range (random) mismatches -

- use segmented or thermometer-coded architectures \Rightarrow **DNL** is reduced by $\sqrt{2}$ for each bit, other than the MSB, that is inserted in the segmentation. This way DNL can be reduced at similar/lower values than INL ones w/o impacting the array area.

Example

To have a $DNL = \pm 0.25LSB$ with the matching (area) request set by an $INL = \pm 1.0LSB$, the array must be segmented in (2^7-1) smaller arrays of 2^{n-7} elementary unit devices, i.e. the MSB and the following six bits are used to implement the segmentation.

Implementations

Resistor string arrays \rightarrow segmented architecture, two-stage row/column decoding logic with 2nd stage decoder and switches in the resistor cell [7].

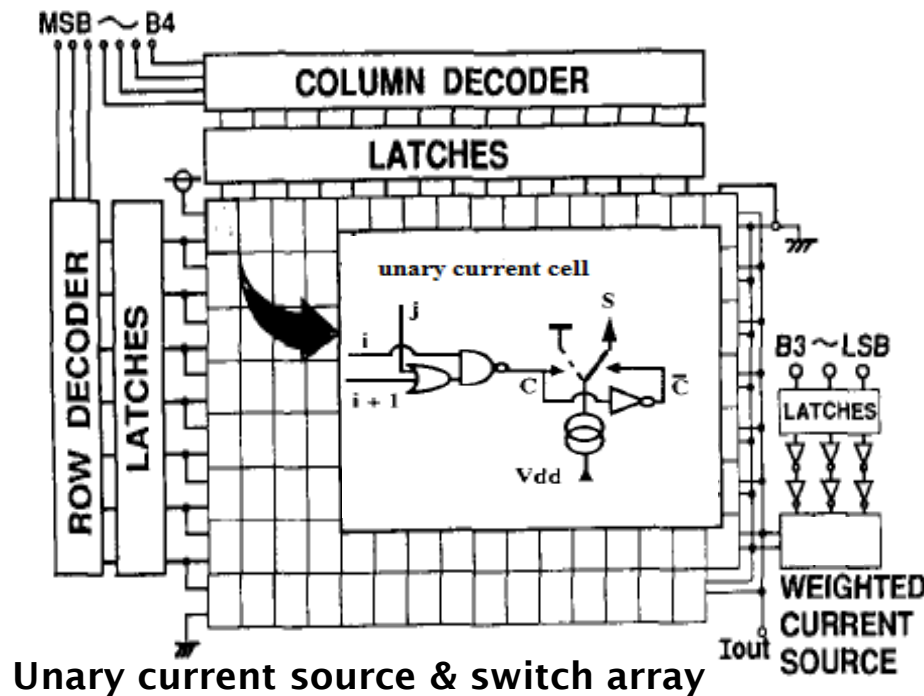
Capacitive arrays \rightarrow segmented architecture, two-stage row/column decoding logic with 2nd stage decoder and switches outside the capacitance cell.

Current steering arrays \rightarrow segmented architecture, two-stage row/column decoding logic with 2nd stage decoder and switches inside [8] or outside [9] the current cell, full decoding logic [10].

DNL improvements

- segmentation implementations (I) -

Two-stage row/column decoding logic with 2nd stage decoder and switches in the current cell [8]



- Low complexity (decoding and cell selection)
- Update rate slightly limited
- Impacted by global mismatches since area of the segmented array is very large.

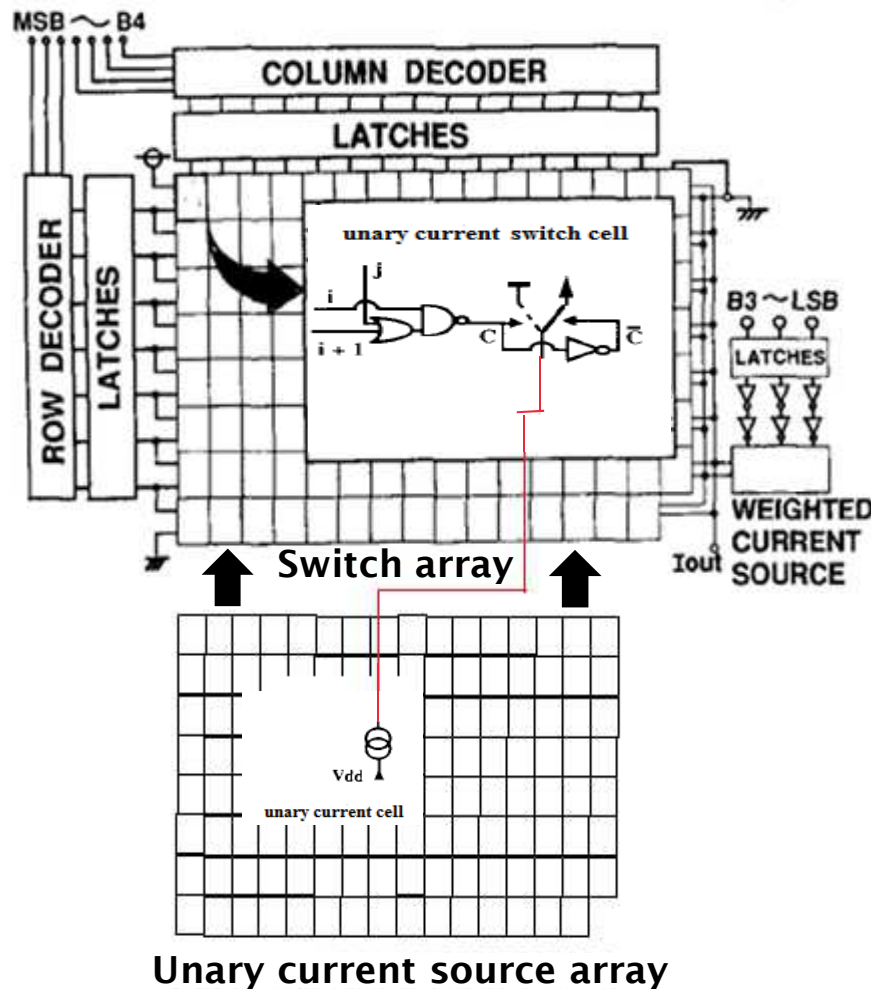
Latches options:

- in the row/column decoders only
- in the cell local decoder only
- both in the row/column and cell decoders

DNL improvements

- segmentation implementations (II) -

Two-stage row/column decoding logic with 2nd stage decoder and switches outside the current cell [9]



- Low complexity (decoding and cell selection)
- Update rate slightly limited
- Reduced impact from global mismatches since area of the segmented array is decreased.

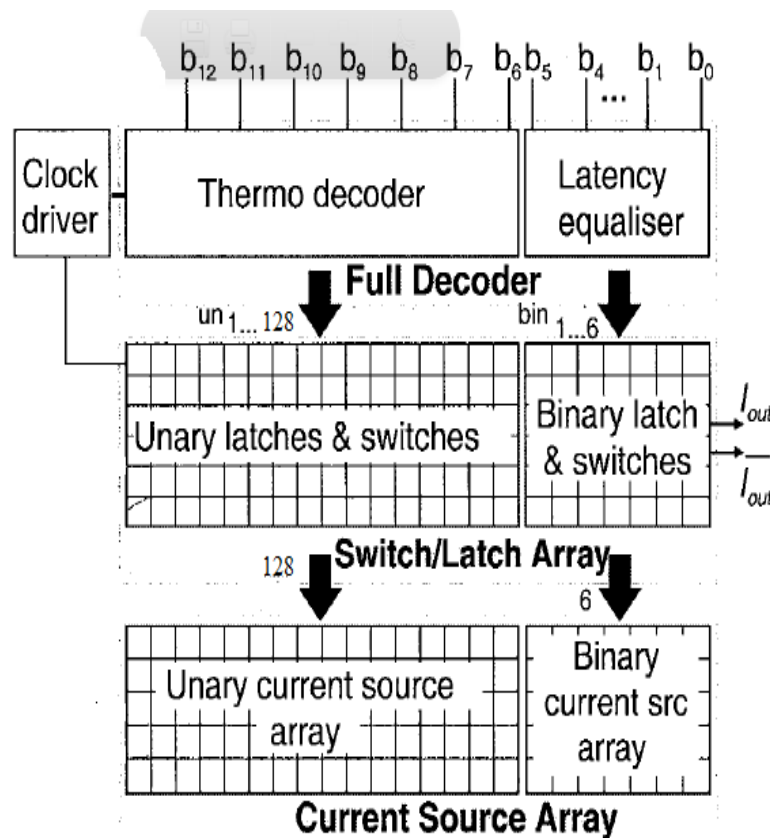
Latches options:

- in the row/column decoders only
- in the cell local decoder only
- both in the row/column and cell decoders

DNL improvements

- segmentation implementation (III) -

Full decoding logic [10]



- High complexity (decoding and cell selection)
- Update rate severely limited
- Minimum impact by global mismatches since area of the segmented array is decreased and a more efficient switching algorithm can be used (see later on)

Latches options:

- in the full decoder only
- in the local cell decoder only
- both in the full and cell decoders

INL/DNL: Effect of long-range mismatches

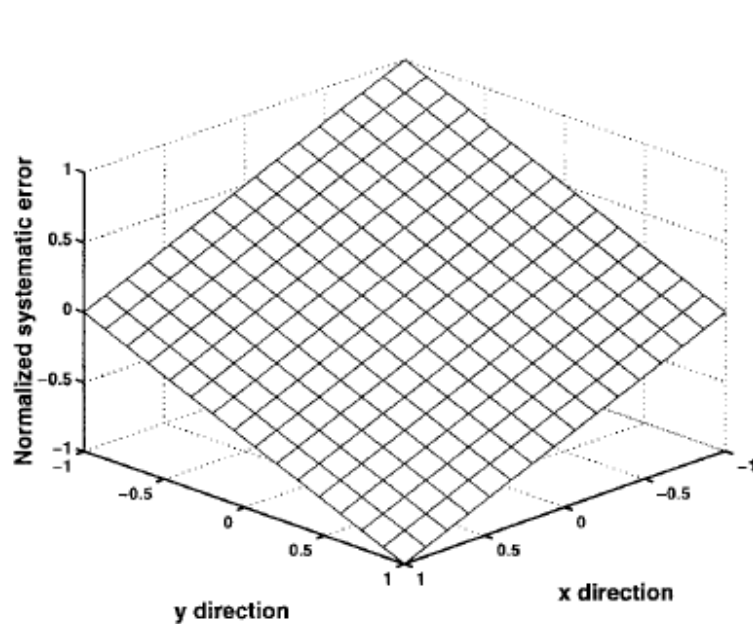
INL/DNL: Effect of long-range mismatches

- Basic long-range mismatches
- INL/DNL degradation limitations
 - Binary-weighted architectures
 - Segmented and thermometer-coded architectures

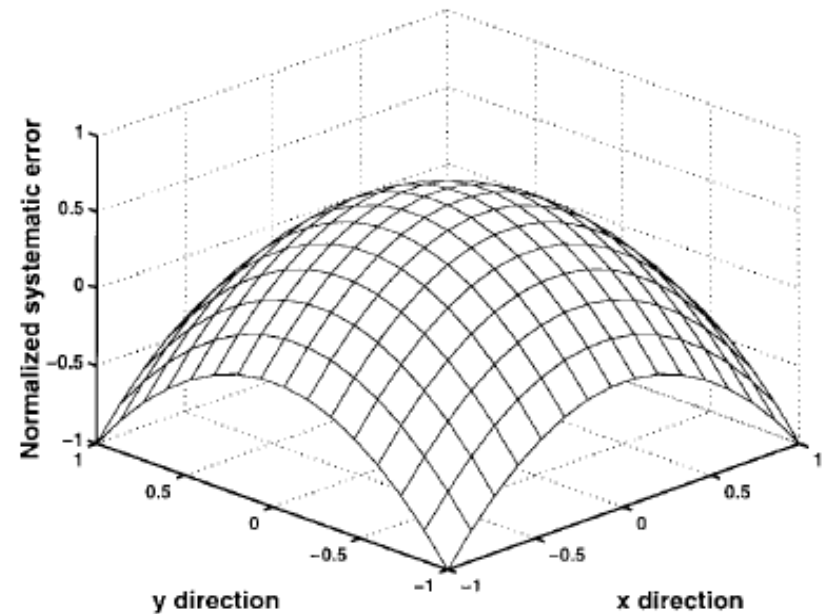
Basic long-range mismatches

Wafers generally exhibit a radial pattern (oxide thickness, implantation dose, etc). This results to a shift in the nominal value of the devices (current sources, capacitors, resistors) inside the array approximately linear with the device separation distance [9].

On the other hand, if the array area is (very) large or in presence of stress gradients, errors approximately parabolic across the array matrix are usually present [9] .



2-D linear (graded) error



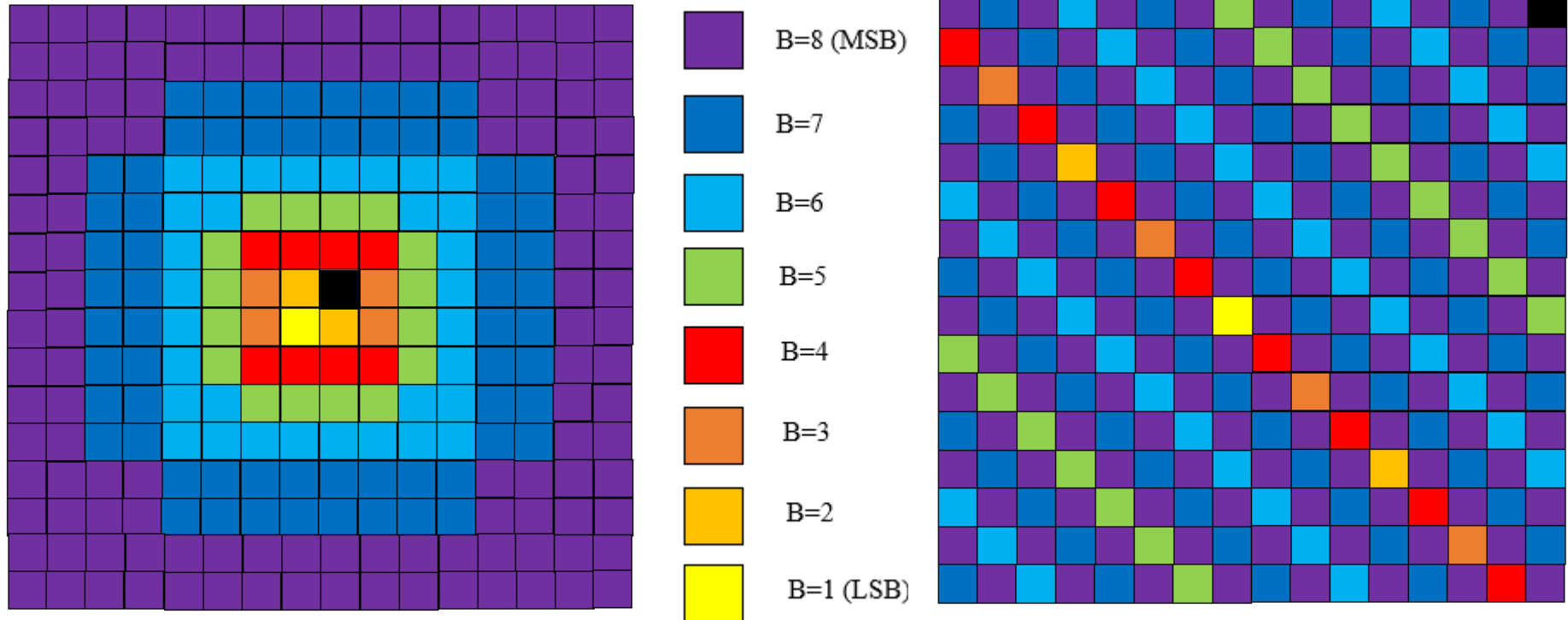
2-D parabolic error

INL and DNL degradation limitations (I)

- long-range mismatches -

Binary-weighted architectures

Example: 8b binary-weighted



Common-centroid layout

best for graded errors

Pseudo common-centroid layout

good for graded and parabolic errors


IMPORTANT: In case of binary-weighted architectures these layout techniques strongly reduce long-range INL and DNL errors.

INL and DNL degradation limitations (II)


- long-range mismatches -

Segmented and thermometer-coded architectures - two-stage row/column decoding (I)

Example: 6b segmented + 2b binary-weighted




1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64




Sequential switching

Very high long-range INL errors
(full error accumulations)




6	2	1	5	7	3	4	8
14	10	9	13	15	11	12	16
22	18	17	21	23	19	20	24
30	26	25	29	31	27	28	32
38	34	33	37	39	35	36	40
46	42	41	45	47	43	44	48
54	50	49	53	55	51	52	56
62	58	57	61	63	59	60	64




1-D Hierarchical symmetrical switching

High long-range INL errors
(1-D error accumulations)



46	42	41	45	47	43	44	48
14	10	9	13	15	11	12	16
6	2	1	5	7	3	4	8
38	34	33	37	39	35	36	40
54	50	49	53	55	51	52	56
22	18	17	21	23	19	20	24
30	26	25	29	31	27	28	32
62	58	57	61	63	59	60	64



2-D Hierarchical symmetrical switching

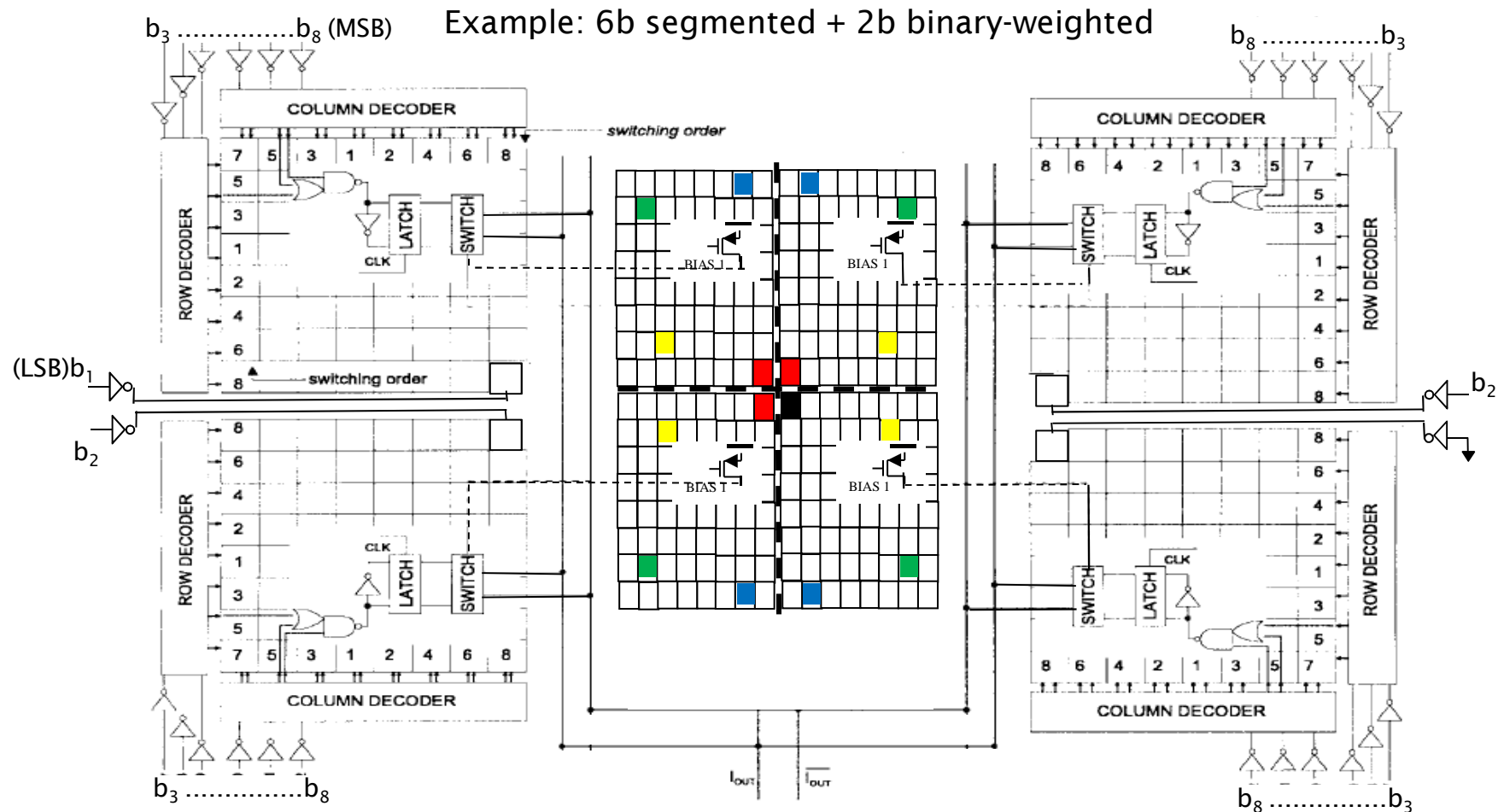
Moderate long-range INL errors
(no error accumulations)

IMPORTANT: whichever type of switching is chosen, the long-range DNL is always 'linked' to the distance between the binary decoded array and an unary cell → The larger is the unary array, the higher is the DNL.

INL and DNL degradation limitations (III)

- long-range mismatches -

Segmented and thermometer-coded architectures - two-stage row/column decoding (II)



2-D Centroid switching → each unary cell is split in four X and Y mirrored subcells with 2D sym. sw. scheme

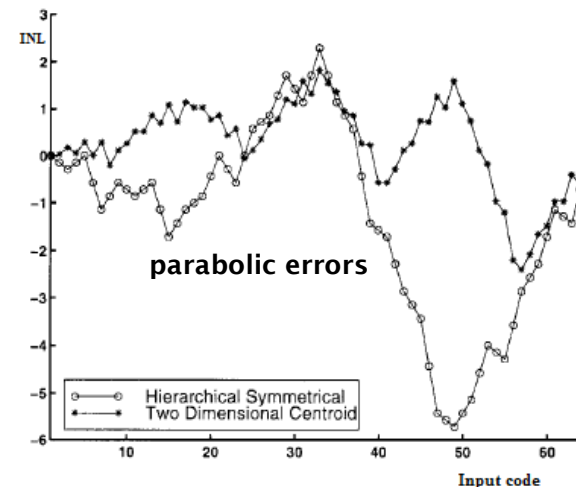
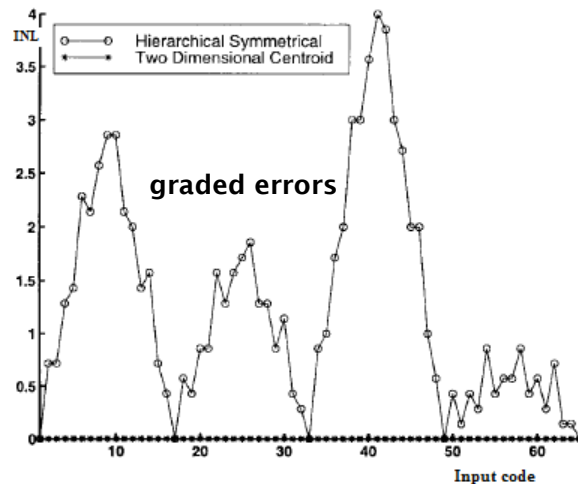
Low long-range INL errors

INL and DNL degradation limitations (IV)

- long-range mismatches -

Segmented and thermometer-coded architectures - two-stage row/column decoding (III)

- In case of graded errors, 2-D Centroid switching presents long-range INL and DNL errors proportional to the distance between the binary decoded array and the center of the unary (segmented) array.
- In case of parabolic errors, 2-D Centroid switching presents long-range INL errors comparable to 2-D hierarchical symmetrical switching, whereas long-range DNL errors are 'linked' to the distance between the binary decoded array and an unary cell.

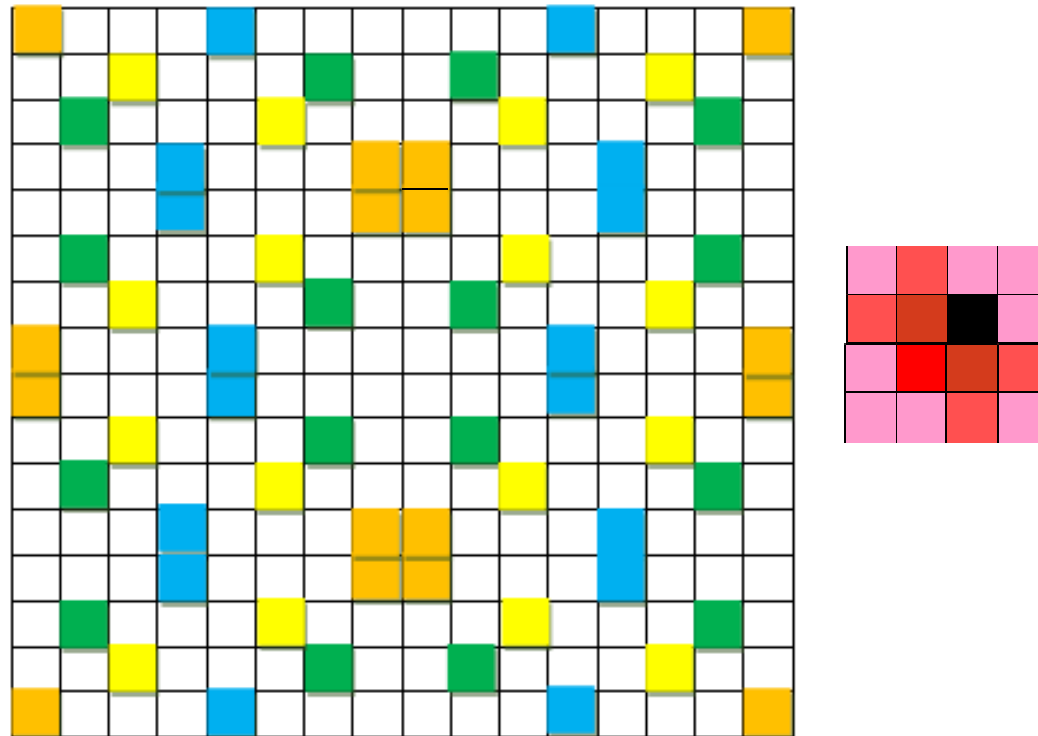


INL and DNL degradation limitations (V)

- long-range mismatches -

Segmented and thermometer-coded architectures – full decoding (I)

Example: 4b segmented + 4b binary-weighted



Q^2 Random Walk switching → each unary cell is split in four (Quad) subcells in each Quadrant

Very low long-range INL errors

INL and DNL degradation limitations (VI)

- long-range mismatches -

Segmented and thermometer-coded architectures – full decoding (II)

- In case of graded errors, Q^2 *Random Walk* switching presents long-range INL and DNL errors proportional to the distance between the binary decoded array and the center of the unary (segmented) array.
- In case of parabolic errors, Q^2 *Random Walk* switching reduces long-range INL errors by an order of magnitude compared to other less complex switching schemes, whereas long-range DNL errors is more or less proportional to the distance between the binary decoded array and the center of the closer quarter of the unary (segmented) array.

INL/DNL: Effect of output impedance in current steering data converters

INL/DNL: Effect of output impedance in current steering data converters

In the case of current steering data converters, INL and DNL are also impacted by their output impedances. It can be demonstrated that INL and DNL degradations are given by the following expressions:

$$\text{Single-ended case} \left\{ \begin{array}{l} INL_{MAX} = INL\left(\frac{N}{2}\right) = \frac{R_L N^2}{4r_{out}} \\ DNL_{MAX} = DNL(0) = DNL(N) = \frac{R_L N}{r_{out}} \end{array} \right.$$

$$\text{Differential case} \left\{ \begin{array}{l} INL_{MAX} = INL\left(\frac{N}{4}\right) = \frac{3R_L^2 N^3}{64r_{out}^2} \\ DNL_{MAX} = DNL(0) = DNL(N) = \frac{R_L^2 N^2}{2r_{out}^2} \end{array} \right.$$

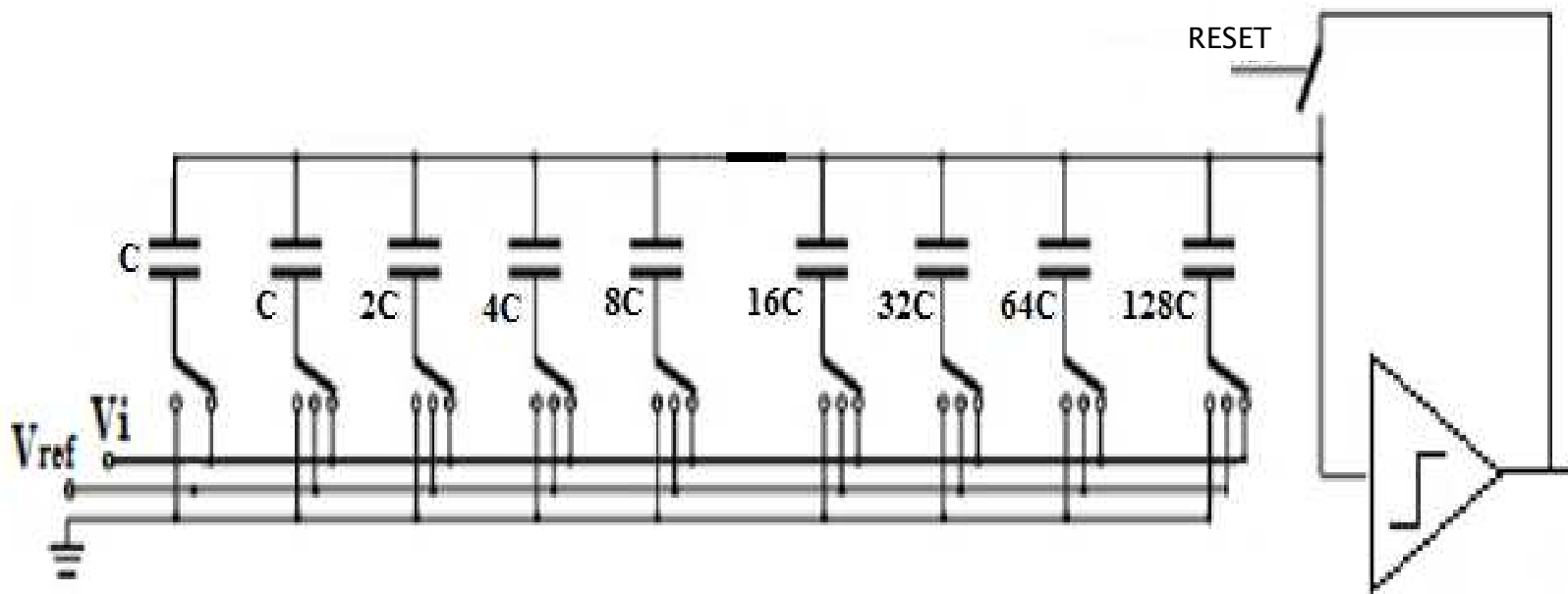
DACs with main and sub arrays

DACs with main and sub arrays

- Motivations
- Implementations

Motivations (I)

When the minimum unit component value, and this is particularly true for MIM and MOM capacitors, supplied by general-purpose design-kits is much larger than necessary to meet the linearity requirements, it results to a considerable very large DAC array.



Binary-weighted capacitive DAC architecture

Motivations (II)

Three solutions to solve this problem:

a) adopt full custom unit components → no standard DK - this approach requires extra efforts to design, test, characterize, and model them.

b) Switch half of the termination component of the array → standard DK.

- suitable for a differential structure only.
- the number of components, as well as the DAC area, is reduced by a factor of 2.

c) split the DAC into main and sub arrays (also called upper and lower arrays) → standard DK.

A N-bit DAC array is split into a M-bit main array and a P-bit sub array, with $M+P=N$.

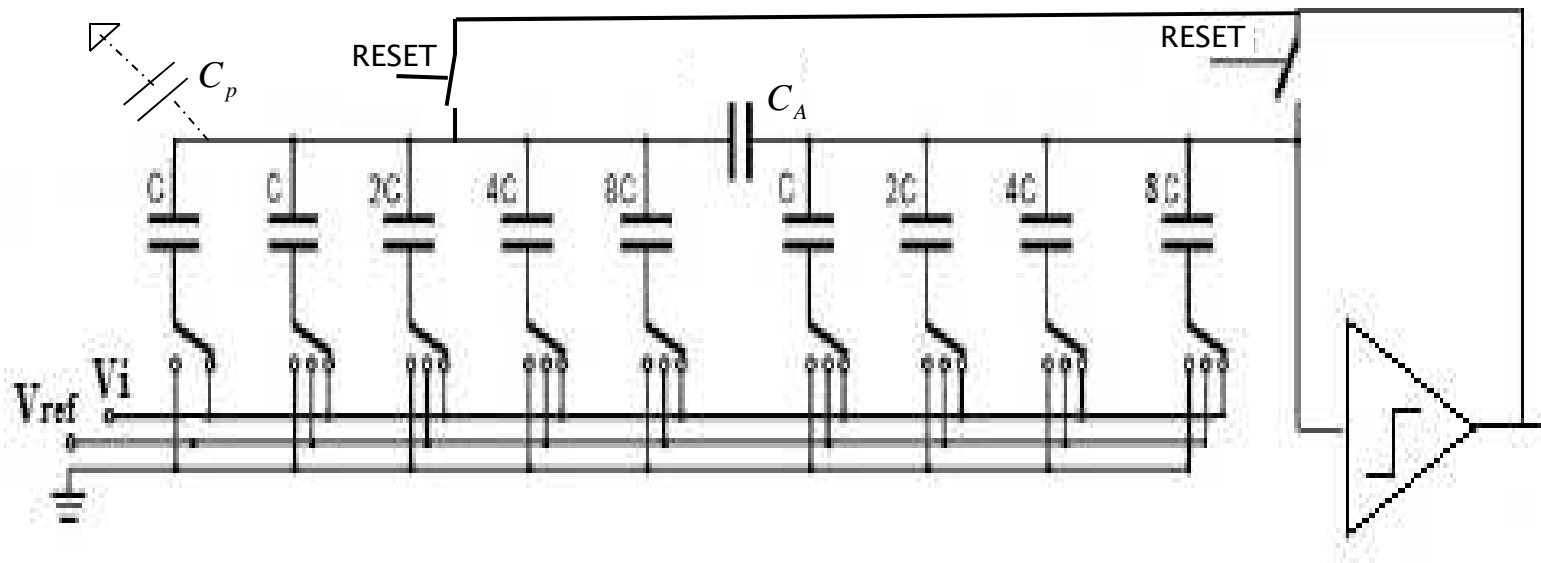
- suitable for both single-ended and differential structures.
- the number of components is reduced from 2^N to about $2^M + 2^P \ll 2^N$ if N is large.
- considerable simplification in layout placement and connections.
- no wasted DAC area to meet the N-bit linearity requirement.

Pay attention to the parasitic capacitor linked to the sub-array common node → non-linearity !!!!

Rule of thumb → $C_p < C$ to assure monotonic behavior of the converter.

Split DAC: Classical implementation

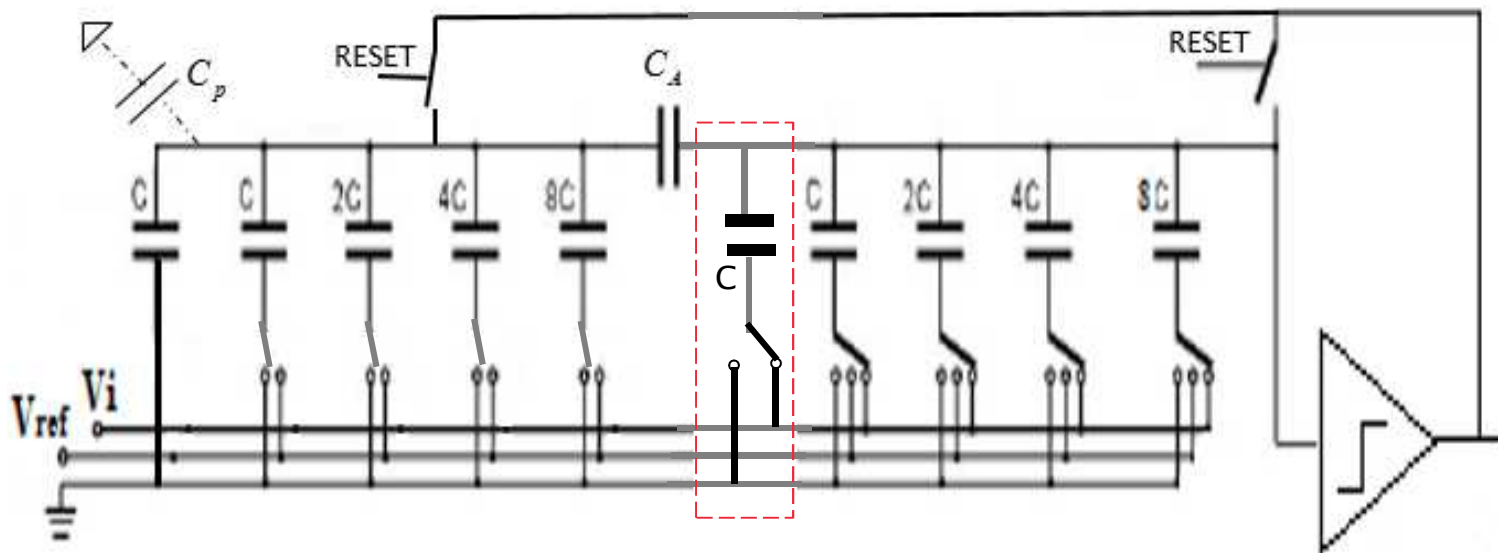
- Input signal is sampled by both upper and lower array capacitors.
- Attenuation capacitor $C_A = \frac{2^p}{2^p - 1} C$, where p is the number of bits in the lower array



→ Unavoidable mismatch between nominal values of C_A and C results to non-linearity !!!

Split DAC: Other implementation (I)

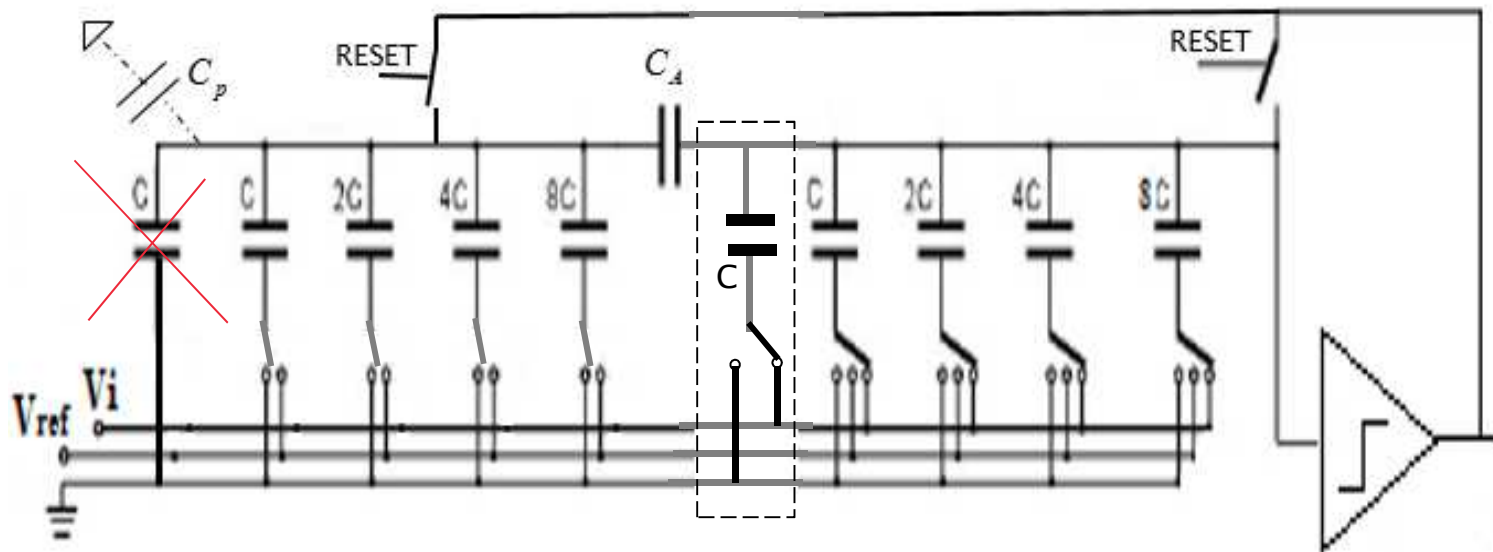
- Input signal is sampled by upper array capacitors only.
- Upper array is equipped with an extra unitary capacitor that plays the role of the lower array during input sampling phase and is never used during conversion phase.
- Attenuation capacitor $C_A = \frac{2^p}{2^p - 1} C$, where p is the number of bits in the lower array



→ Unavoidable mismatch between nominal values of C_A and C results to non-linearity !!!

Split DAC: Other implementation (II)

- Lower array termination capacitor is removed.
- Input signal is sampled by upper array capacitors only.
- Upper array is equipped with an extra unitary capacitor that plays the role of the lower array during input sampling phase and is never used during conversion phase.
- Attenuation capacitor $C_A = C$, independently from the number of bits in the lower array



→ No mismatch between nominal values of C_A and C !!!

Self-calibration

Self-calibration

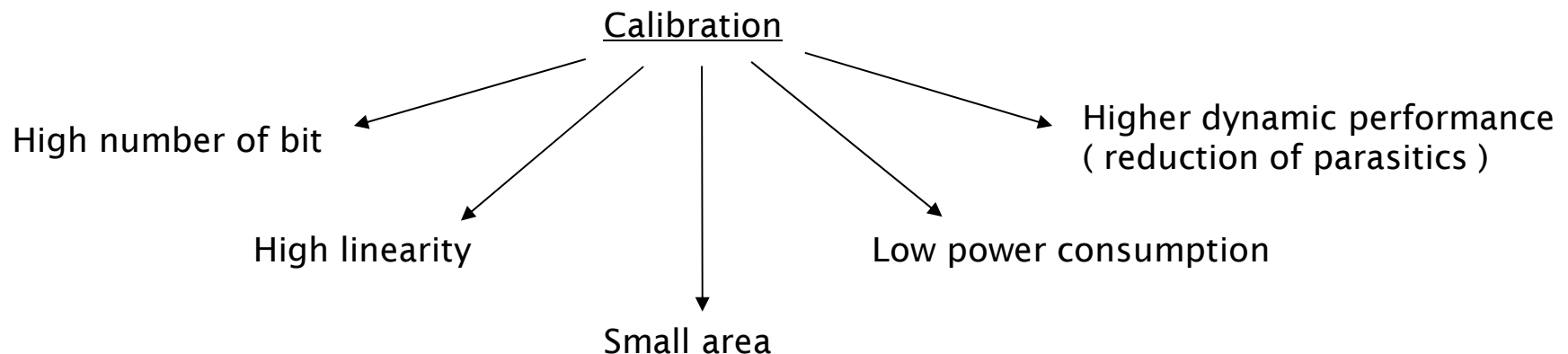
- Motivations
- Background calibration
- Foreground (at start-up) calibration
- Off-chip software processing calibration

Motivations

The higher the number of bits of a data converter, the larger the array area needed to statistically maintain a required level of matching accuracy between components, and in turn the more complex the layout and architecture, i.e. the decoding and the switching sequence, to cope with long-range errors → Intrinsic accuracy data converters

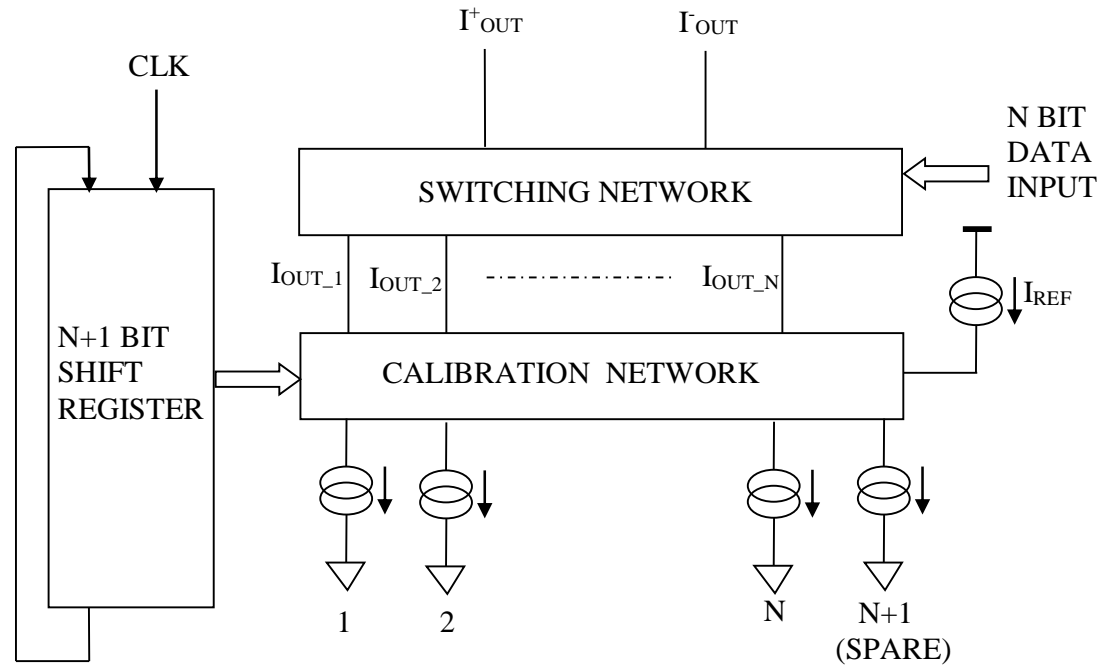
This not only results in large die size but also in significant degradation of high-frequency performance of the converter since large area introduces substantial parasitic effects which severely impact the settling behavior of the data converter causing the Spurious-Free Dynamic Range (SFDR) to rapidly drop at high frequencies.

To overcome these drawbacks, and reduce data converter sensitivities to process, temperature, packaging, and aging, calibration has been introduced on the thermometer-coded components.



Background calibration

Basic principle

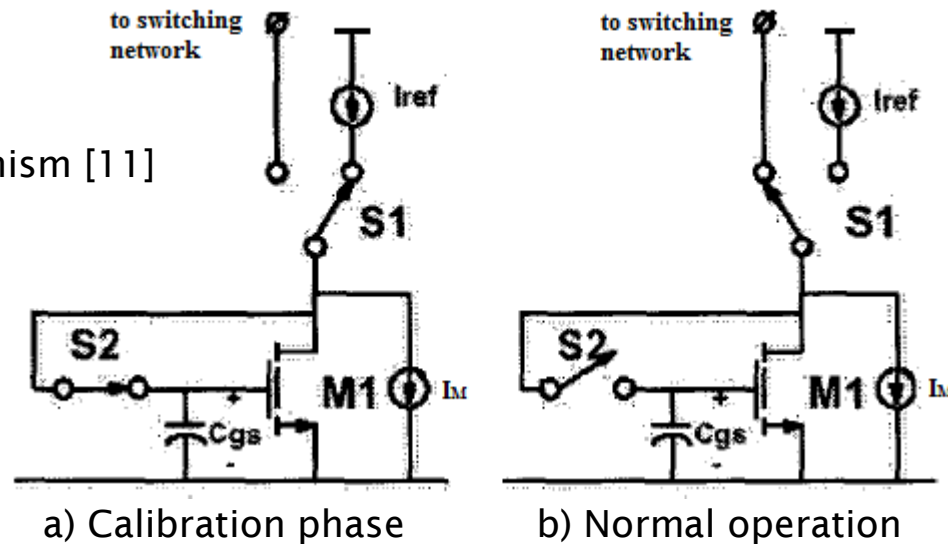


The calibration network connects $N-1$ cells to their corresponding outputs, whereas the cell under calibration is instead connected to the reference one. Since this last cell is not connected to its output terminal, the spare cell is switched to this terminal. In this way it is guaranteed that there are always N equal cells available at the output terminals.

- This allows the DAC to be in use while being continuously calibrated.
- However, the operation of switching in and out cells introduces spurs at the calibration frequency in the output spectrum. In theory, time domain randomization of the calibration slot length can spectrally spread out these spurs into noise.

Background calibration example

Basic correction storing mechanism [11]



Pros:

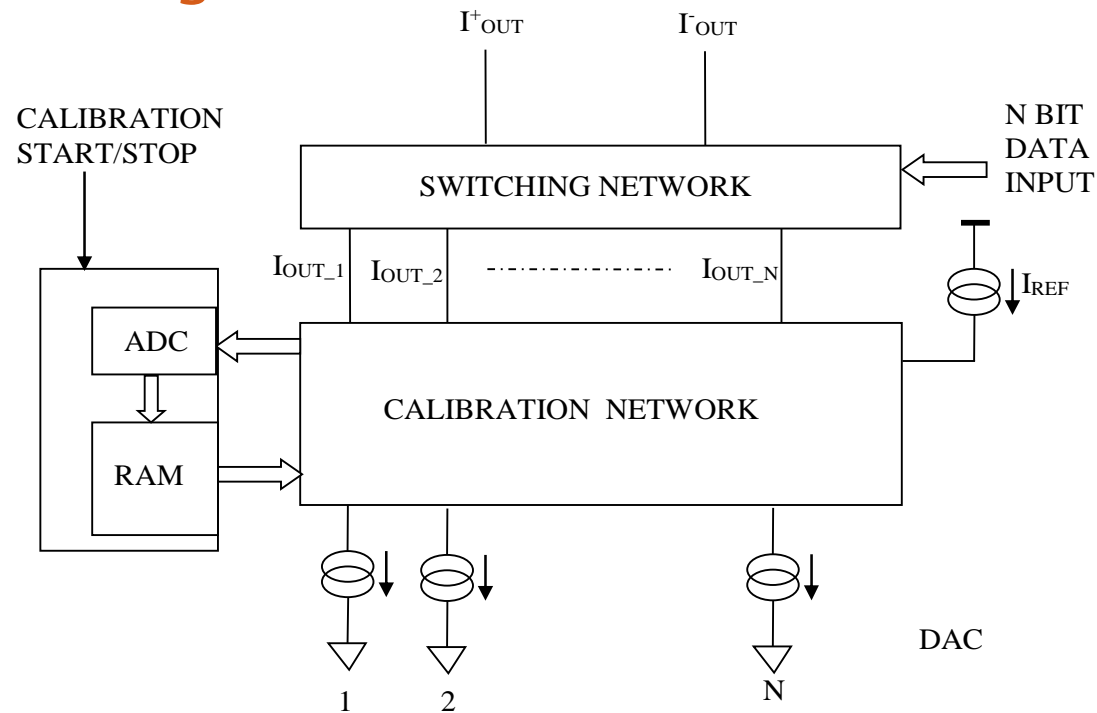
- Simplicity: the correction voltage is analog and it is stored on the gate capacitance of M1.
- Area saving: very small extra area needed to implement switches S1 and S2.

Cons:

- Dynamic storage: correction voltage needs to be periodically refreshed to compensate for junction leakage of switch S2 when off, thus calibration is lost if the clock is turned off during power saving modes.
- Accuracy limitation: channel charge dumping when S2 is switched off and leakage on capacitor.
- Accuracy limitation: V_D of M1 and I_M must be the same during calibration and normal operation.

Foreground calibration

Basic principle



The calibration network connects one cell at a time to the reference one and the difference is adjusted as close to zero as possible through a common (SAR) ADC and calibration DAC(s).

- The converter must be taken off line and not used while being calibrated.
- The calibration circuit is not clocked during normal operation and does not use power or inject noise during normal operation.

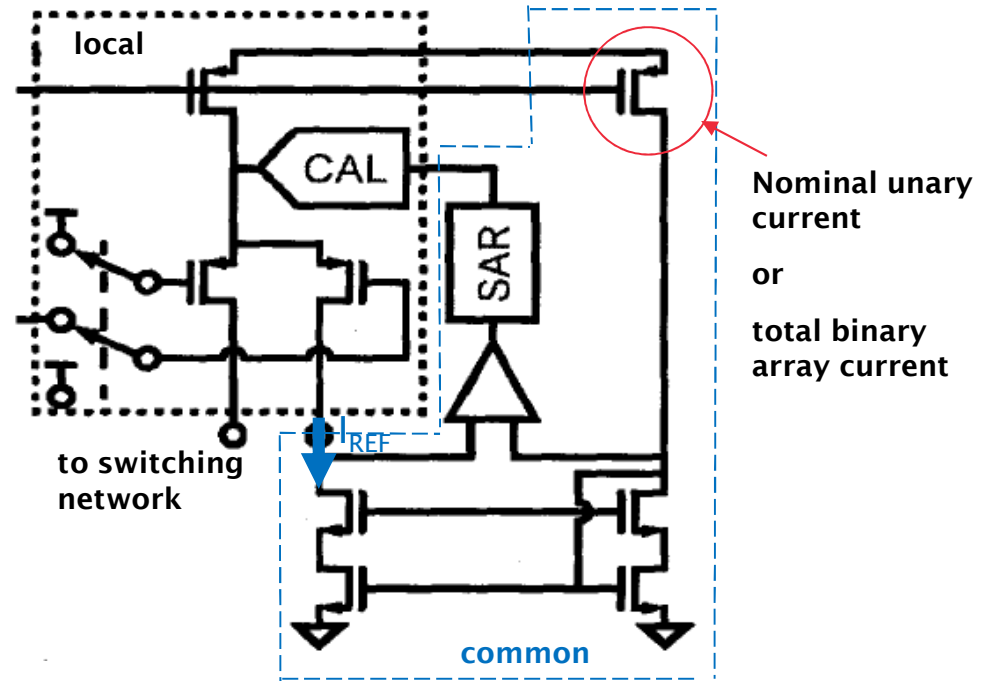
Two major sub-classes:

a) correction of the individual current sources
(multiple correction DACs)

b) correction of the output current
(single correction DAC)

Foreground calibration of individual current sources

Basic correction storing mechanism [13]



Pros:

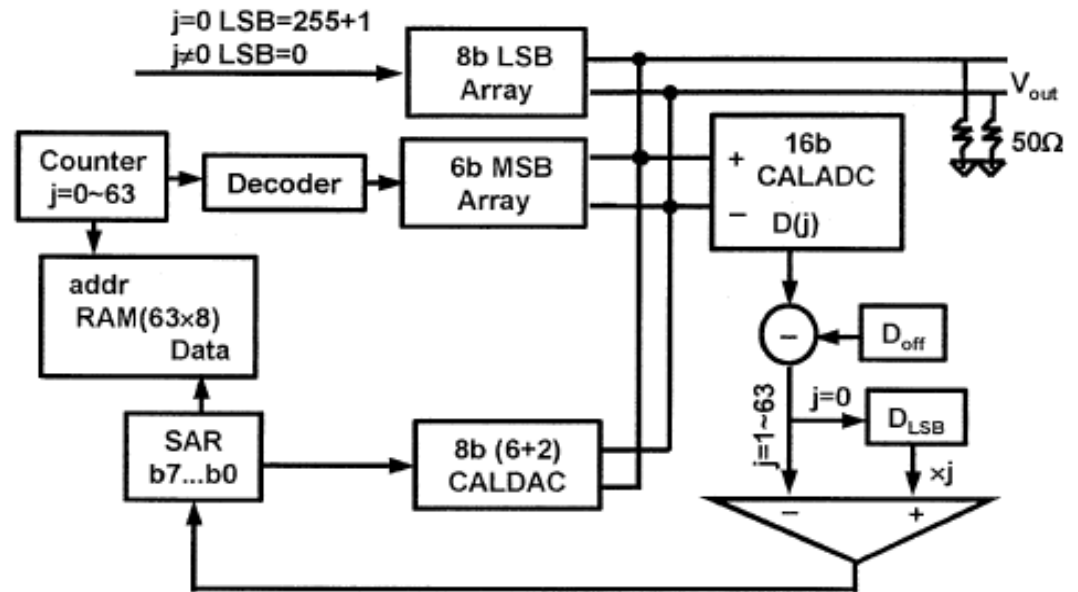
- Static storage: correction voltage does not need to be periodically refreshed, thus calibration is active also if the clock is turned off during power saving modes.
- Accuracy: accurate calibration, depending on calibration DAC resolution.
- No dynamic performance limitations: the correction is embedded in each current source and it is static, i.e. not carried out during normal DAC operation.

Cons:

- Complexity : the correction voltage is digital and stored in a memory, thus calibration needs to implement a SAR algorithm on several local low resolution DACs.
- Area wasting: large area used for local DACs.

Foreground calibration of the output current

Basic correction storing mechanism [12]



Pros:

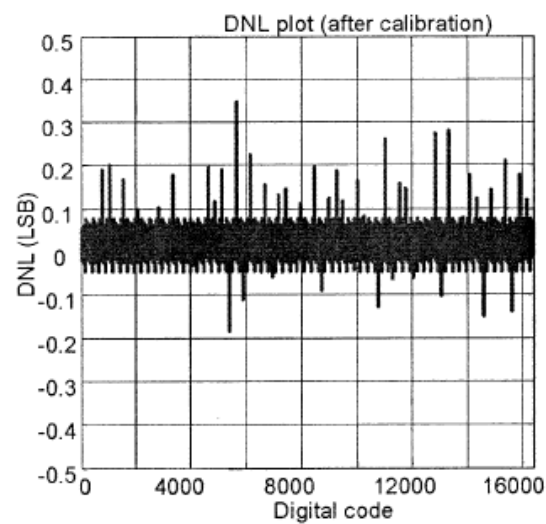
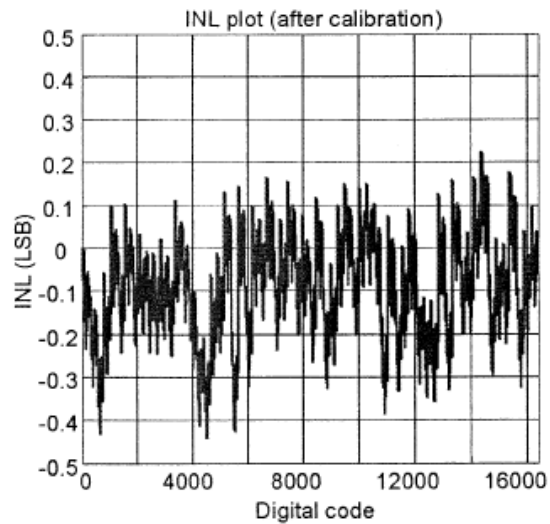
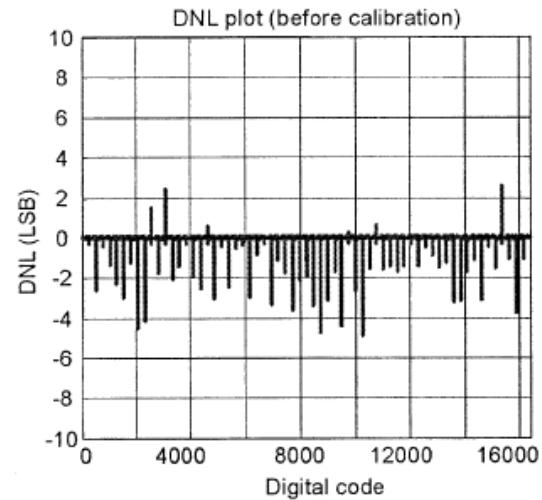
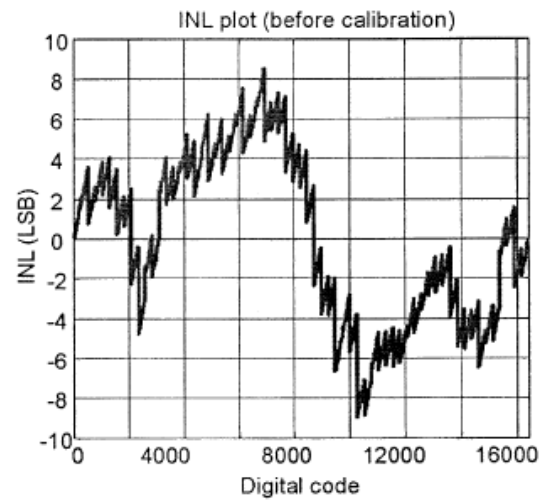
- Static storage: correction voltage does not need to be periodically refreshed, thus calibration is active also if the clock is turned off during power saving modes.
- Accuracy: accurate calibration, depending on ADC and calibration DAC resolutions.

Cons:

- Complexity : the correction voltage is digital and stored in a memory, thus calibration needs a high resolution ADC and a medium resolution DAC.
- Area wasting: large area used for ADC and calibration DAC.
- Dynamic performance limitations: the correction is not static, i.e. different calibration current sources are carried in and out according to the selected code during normal DAC operation.

Foreground calibration of the output current (cont'd)

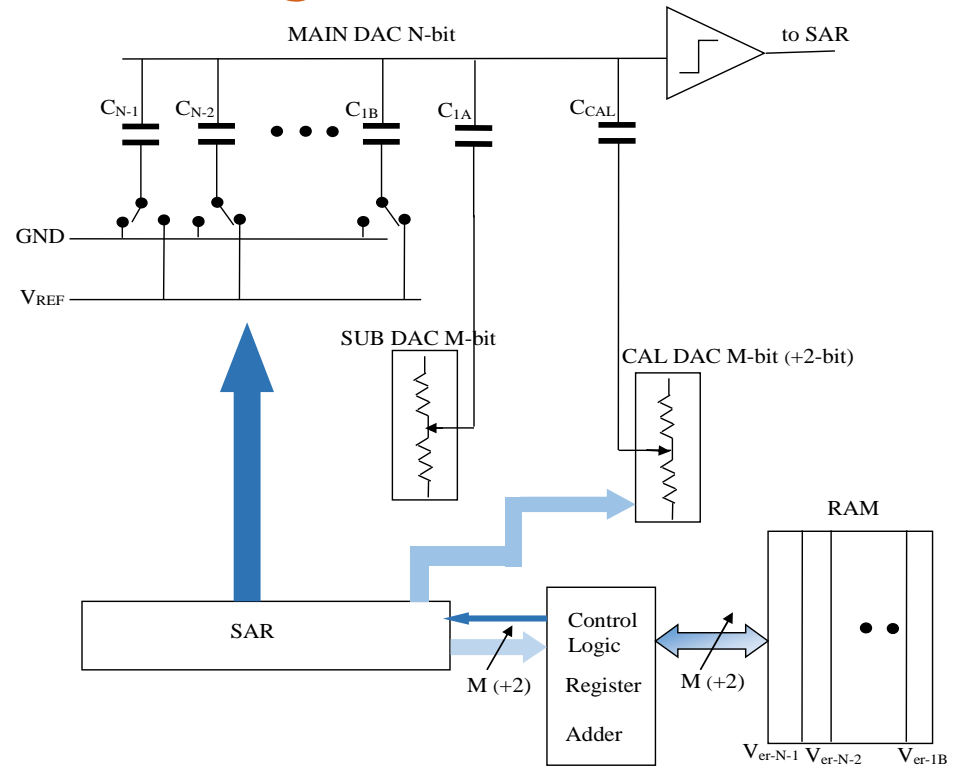
INL/DNL measured results : 1LSB @ 14bits



Foreground calibration with single-correction DAC

Basic correction storing mechanism [14]

The difference between each capacitor in the array and the sum of all capacitors at the lower bit position is digitally measured and later used for mismatch correction.



The calibration cycle begins by measuring the nonlinearity $V_{err-N-1}$ due to MSB cap C_{N-1} . This is done by sampling V_{REF} on all caps except C_{N-1} that is tied to GND, and then reversing all the switches.

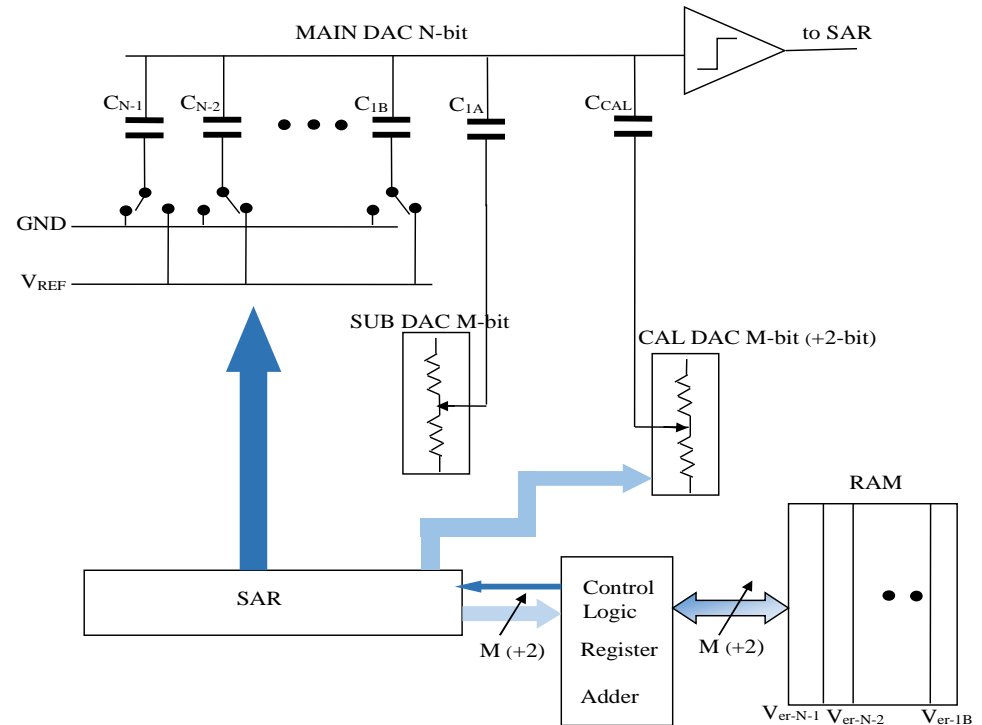
The residual voltage $V_{res-N-1}$ at the top plate, which is two times the error $V_{err-N-1}$, is digitized by CAL DAC with 2 bit more precision than the SUB DAC. Then, $V_{err-N-1} = 0.5 V_{res-N-1}$ is stored in the RAM.

Similarly, errors due to smaller caps are measured. It can be shown that the general relation between residual voltages V_{res-k} and error voltages V_{err-k} is given by [14] (see ex. in Appendix 1):

$$V_{err-k} = 0.5 \cdot \left(V_{res-k} - \sum_{i=k+1}^{N-1} V_{err-i} \right) \quad k = 1B, 2, \dots, N-2$$

V_{err-k} are stored in the RAM and added, according to the input code, before being sent to CAL DAC.

Foreground calibration with single-correction DAC (cont'd)



Pros:

- Static storage: correction voltage does not need to be periodically refreshed, thus calibration is active also if the clock is turned off during power saving modes.
- Accuracy: accurate calibration, depending on calibration DAC resolution.

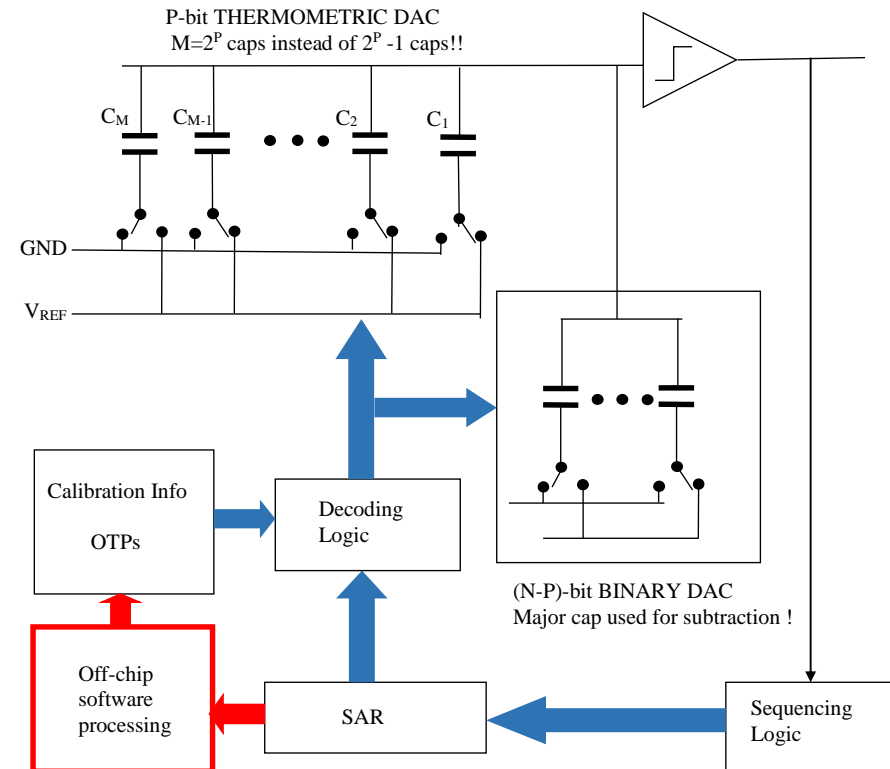
Cons:

- Complexity : the correction voltage is digital and stored in a memory, thus calibration needs to implement a SAR algorithm on a medium resolution DACs, and perform several digital add/subtract operations at each calibration step.
- Area wasting: extra area due to CAL DAC (2 bit more than SUB DAC), memory, registers, and adder.

Off-chip software processing calibration

Basic “correction” storing mechanism [15]

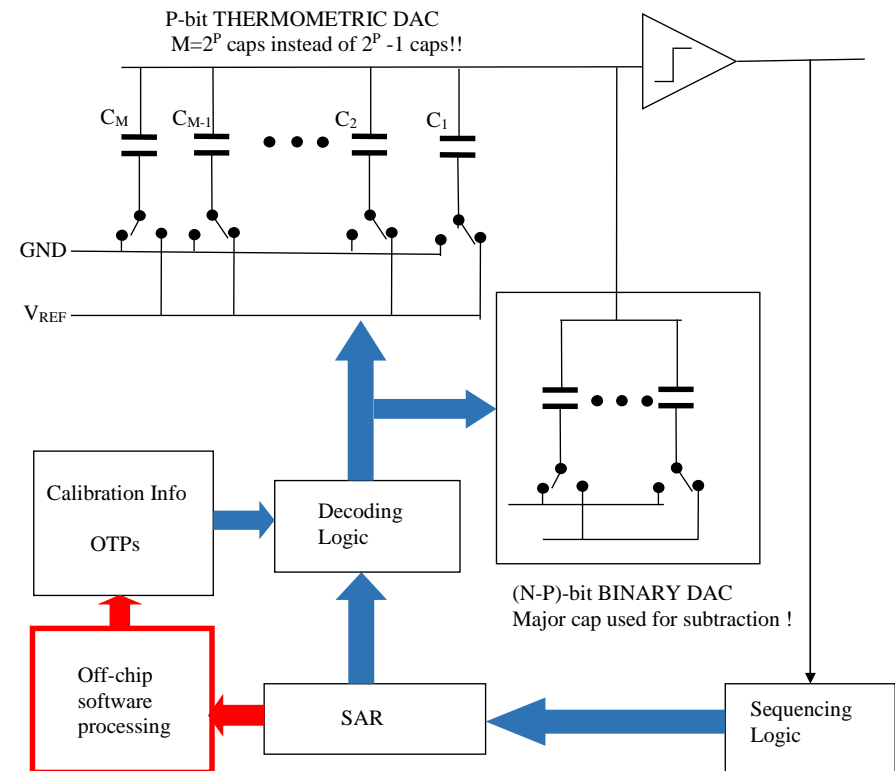
The difference between each capacitor and an arbitrary chosen reference one C_{ref} in the thermometric array is digitally measured using the binary array and later on used for mismatch “correction”.



During calibration of each thermometric capacitor C_k , V_{REFP} is sampled on C_k whereas V_{REFN} is sampled on reference capacitor C_{ref} . Then C_k is connected to V_{REFN} and C_{ref} to V_{REFP} and the residual voltage V_{res-k} at the top plate, which is two times the error V_{err-k} , is digitized by the binary DAC with extra 3-b or 4-b precision (using a scaled voltage reference).

Since the DAC can also perform subtraction (principle is shown in Appendix 2), both positive and negative errors can be measured and stored in an external memory to be processed later on. Since values to be measured can be very small, a single measure for V_{err-k} is not enough accurate due to quantization and comparator noises. Thus several V_{err-k} measurements are extracted (4096 measures in [15]) and then averaged out to get an accurate V_{err-k} value.

Off-chip software processing calibration (cont'd)



Pros:

- Static storage: calibration is executed once during production testing, and results are stored in OTPs.
- Accuracy: accurate calibration since measures are averaged out.
- Area wasting: negligible extra area and no power consumption.

Cons:

- Test time: large amount of measures and software processing increase production test time.
- What about aging?

Off-chip software processing calibration algorithms

Question: After capacitor relative errors have been measured and stored off-chip, how can be possible to “correct” for them w/o using dedicated analog hardware on chip?



The good idea is a smart usage of the intrinsic redundancy offered by the thermometric decoding when selecting the elements.

In other way, the best element selection sequence that minimizes INL is stored on-chip in OTP cells during the production testing process.

Two algorithms can be used to fix this selection sequence:

Algo 1: based on worst DNL values



$$INL_{\text{worst}} = DNL_{\text{worst}} / 2$$

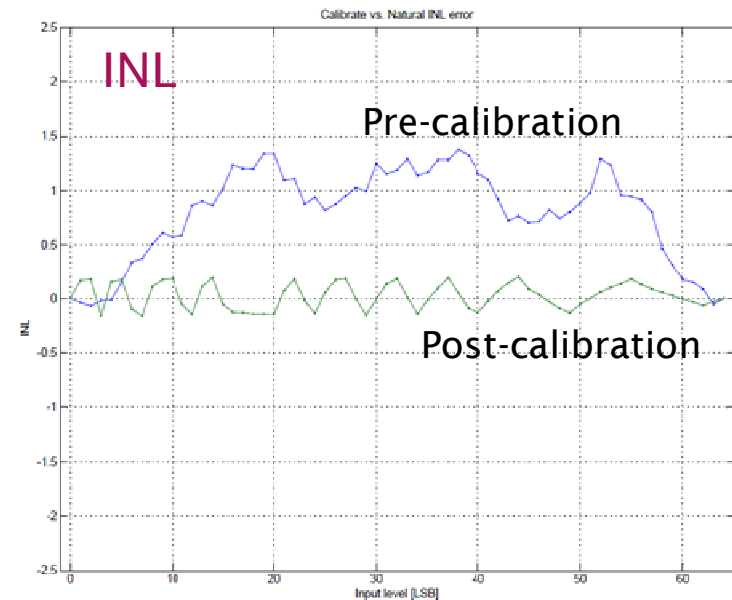
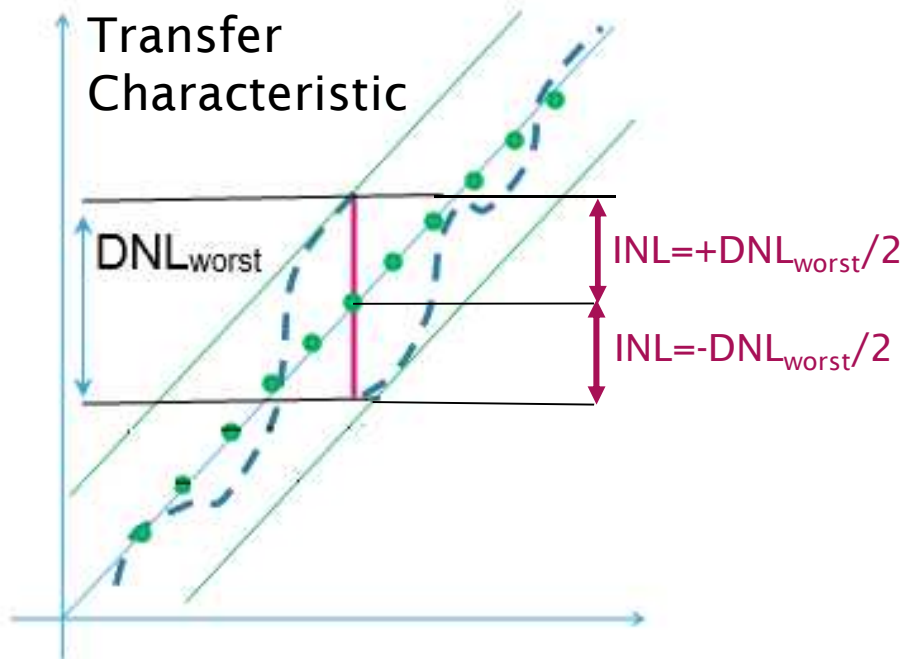
Algo 2: based on ‘good’ and ‘bad’ sets of elements



$$INL_{\text{worst}} = DNL_{\text{goodset_worst}} / 2$$

Off-chip software processing calibration algorithm 1

- a) Select the element with DNL_{worst} in the full set of elements.
- b) Select the elements with low DNLs such as their cumulate DNL is close to $DNL_{\text{worst}}/2$.
- c) Place in the OTP this low DNL element sequence followed by the DNL_{worst} element.
- If the $DNL_{\text{worst}}/2$ value has been accurately approximated, the INL will “jump” from $DNL_{\text{worst}}/2$ to $-DNL_{\text{worst}}/2$.
- d) Remove the previous elements from the full set of elements.
- e) Iterate from a) till the full set of elements is empty.



Off-chip software processing calibration algorithm 2

Since the distribution law for the thermometric element is Gaussian, a large number of them is placed near the center, showing good accuracy. This is the 'good' set of elements. The rest is the 'bad' set of elements.

a) split the set of thermometric elements into two subsets A and B with equal number of elements, where A contains only 'good' elements, whereas B contains remaining ones.

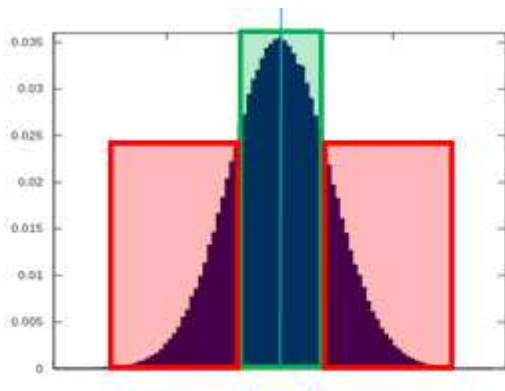
$$\text{Since } \sum_{all} DNL_k = 0 \longrightarrow \sum_A DNL + \sum_B DNL_k = 0 \longrightarrow \sum_A DNL_k = -\sum_B DNL_k$$

The subset A should minimize the absolute value of the sum of the DNL of all its elements.

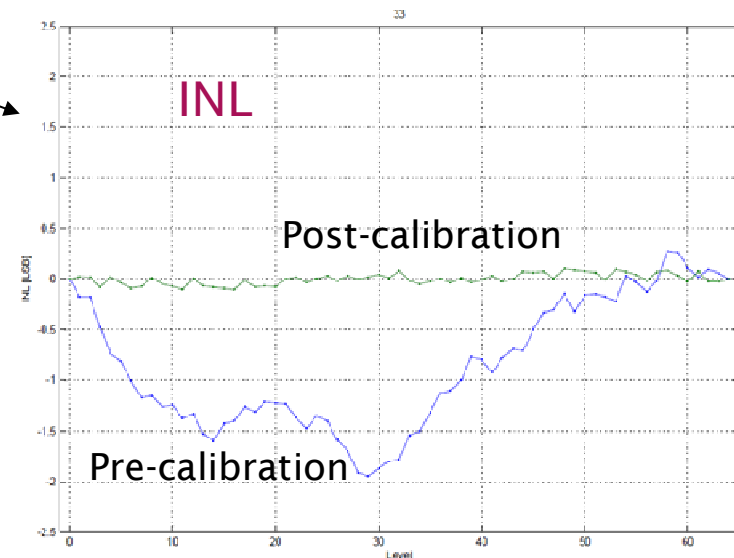
b) The ordering of the elements of the A subset can be fixed with an algorithm similar to Algo 1, resulting to $INL = DNL_{\text{goodset_worst}} / 2$.

d) For the first half of the thermometric scale use only elements from subset A

e) For the second half of the thermometric scale substitute subset B in place of subset A in the lower part of the array, and use again subset A in the higher section of the array but with the elements used in the reversed order.



'Good' set
'Bad' set



Redundancy in ADCs

Redundancy in ADCs

- Motivations
- Radix=2 design with redundant conversion steps
- Radix=2 designs with redundant decision levels
- Radix<2 design with redundancy in search-space convergence

Motivations

Self-calibration has been introduced in high resolution data converters to mitigate the linearity and performance limiting factors due to component mismatches while maintain a reasonable silicon area. Moreover, calibration reduces data converter sensitivities to process, temperature, packaging, and aging.

However

- When dealing with A/D converters, since self-calibration cannot help resolve dynamic conversion errors due to incomplete decision level settlements and unlimited pdf of the noise, each DAC tentative has to settle within $\pm 0.5\text{LSB}$ of its ideal value and a $\sigma < 0.1\text{LSB}$ must be forecast for the noise. These constraints result to an increase of ADC power consumption and area.

To cope with these non-idealities, redundancy has been introduced in A/D data converters:

- Radix=2 design with redundant conversion steps
- Radix=2 designs with redundant decision levels
- Radix<2 design with redundancy in search-space convergence

- When dealing with D/A converters, dynamic performance limitations are mainly due to glitches, as we will see later on.

Radix=2 design with redundant conversion steps

The strategy is to have one or more correction cycles per conversion with the standard convergence rate of 2/cycle for the other decisions in the conversion.

During a SAR operation at most 2 out of N comparisons are likely to be critical because of noise and incomplete settling. One of these critical decisions will certainly be the last one where errors must be avoided, whereas the other can be any of the previous (N-1)-th comparisons where errors can be possible.

- if error is not bigger than ± 1 LSB, it can be corrected by adding one correction cycle just after the N-th cycle and selecting a comparator strategy during both N-th and correction cycles to reduce as much as possible the error probabilities of these two last decisions [16],[17].
- if error is not bigger than ± 3 LSB (or one lower ± 2 LSB and the other lower than ± 1 LSB), it can be corrected by adding two conversion cycles, one after the (N-1)-th cycle and the other after the N-th cycle, and selecting a comparator strategy during (N-1)-th, N-th, and the two correction cycles to reduce as much as possible the error probabilities of these four last decisions [17].
- In the general case, (2^i+1) -LSB errors can be corrected, e.g. incomplete settling of the first decision level can be the case, by following this strategy and just moving the first correction cycle after the (N-i)-th cycle.

In all the cases errors are resolved by a very simple reconstruction logic based on adders.

Correction Algorithm

D	Dx	Correction
0	1	NO
1	0	NO
1	1	+1
0	0	-1

↑ ↑
Conversion cycle Correction cycle

Important!!!

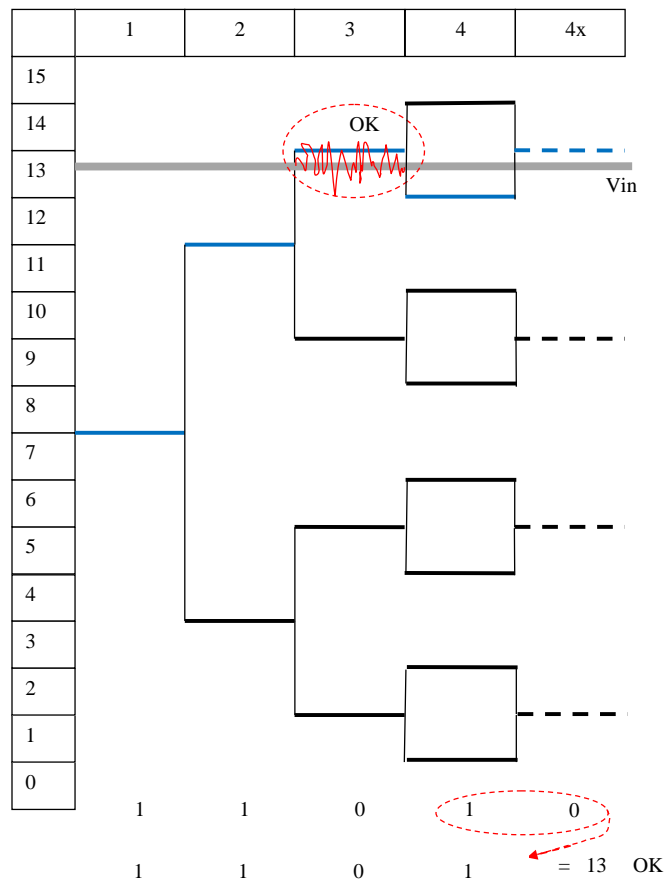
This correction algorithm works only if the outputs of conversion cycle and its related correction cycle are correct.

To reduce as much as possible the error probabilities of these two decisions, dedicated comparator strategies can be used (see examples).

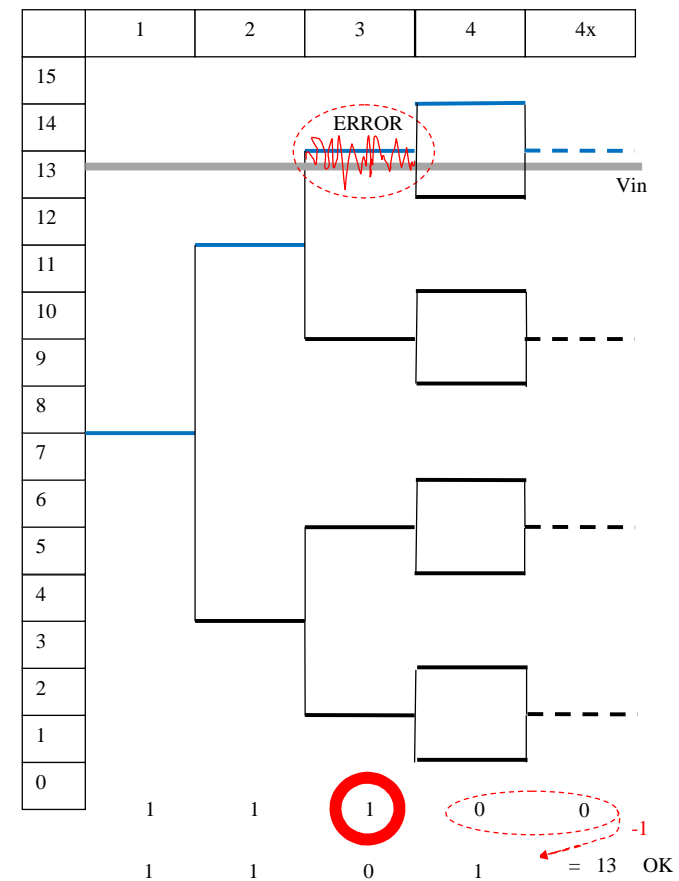
Ex.1: incomplete settling / comparator only noise $< 1 \text{ LSB}$

In the lucky case that noise is due mostly to comparator, i.e. clean reference and power supplies, no noise coupling from substrate, etc., it is enough to employ a low-noise comparator only during the N-th and correction cycles and process these last two decisions according to a simple true table [16].

No error

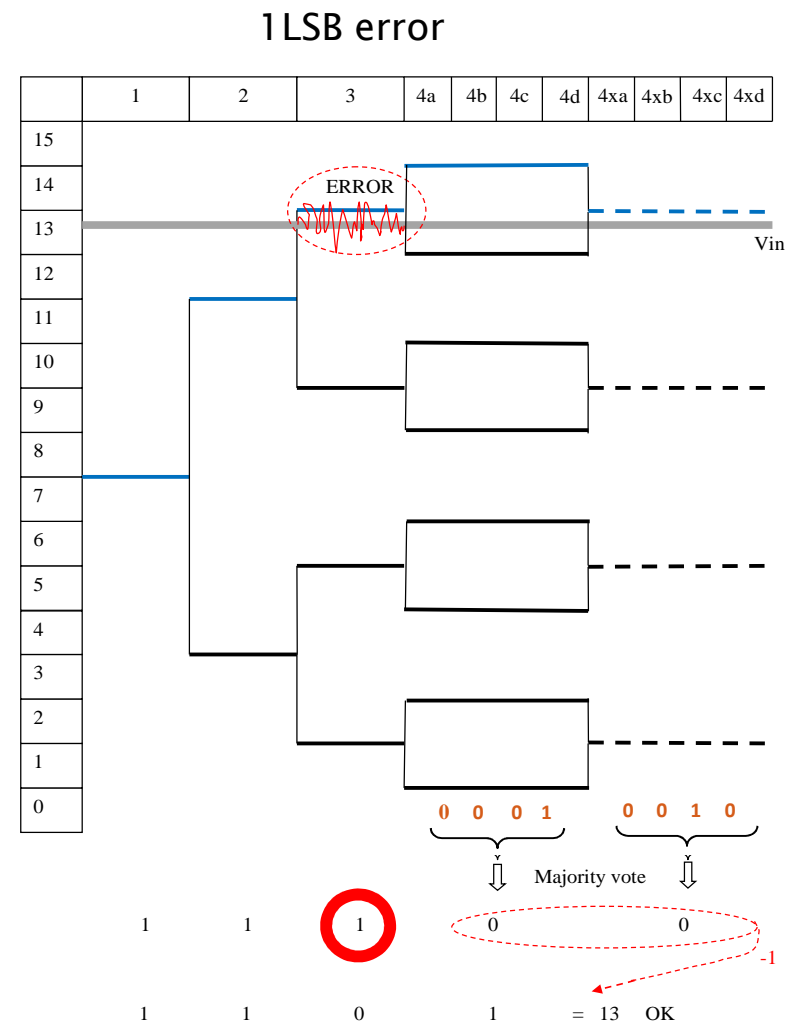
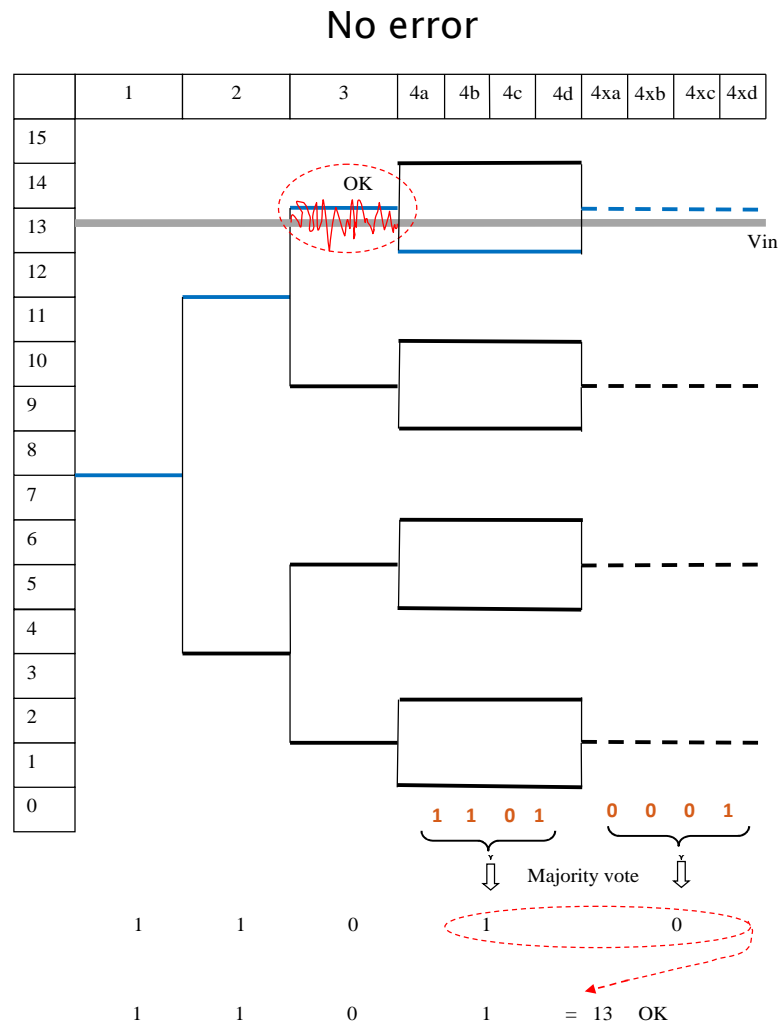


1LSB error



Ex.2: incomplete settling / ADC noise < 1 LSB

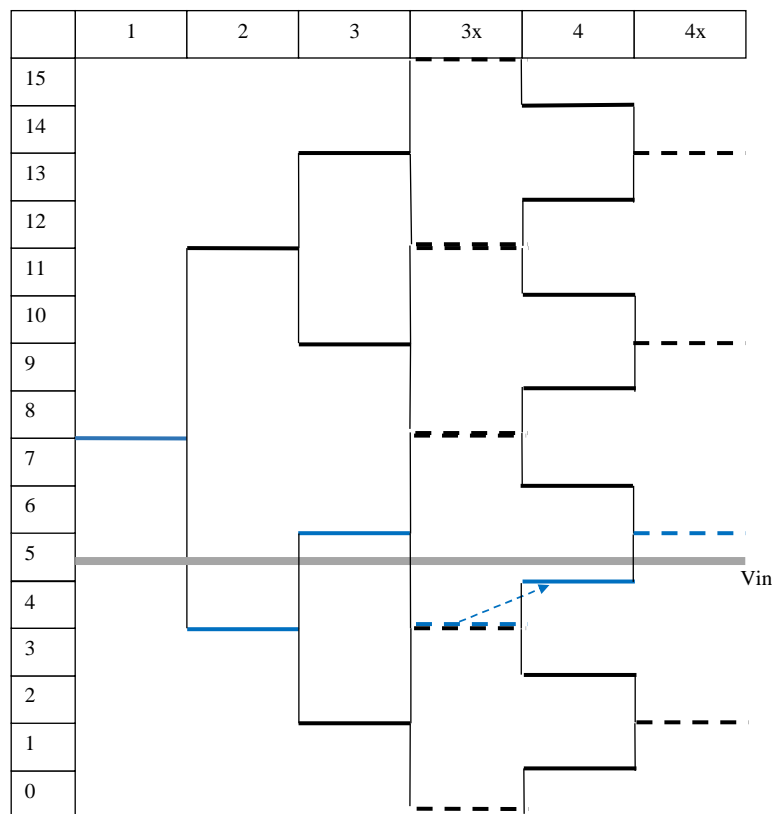
In the case comparator noise is not dominant over other noise sources, it has no sense to employ a low-noise comparator, but instead the N-th and correction cycles are repeated multiple times and then a majority vote has been taken to average out the conversion noise in order to reduce as much as possible the error probabilities of these two last decisions [18].



Ex.3: incomplete settling < 3LSB

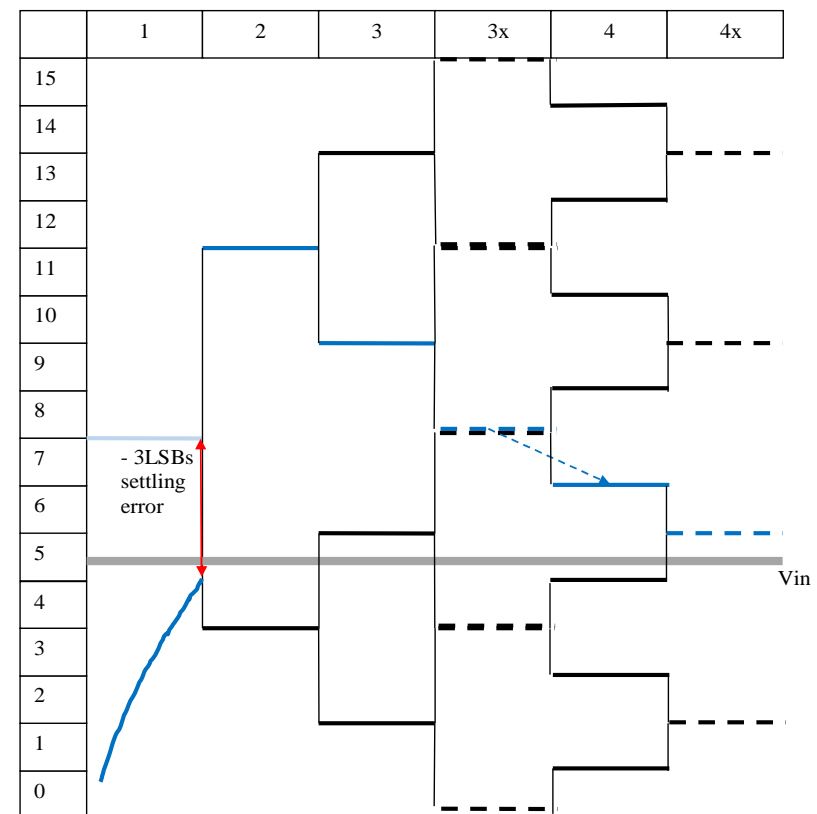
In case of incomplete settling of 3LSBs, two correction cycles can be added, one after the (N-1)-th cycle and the other after the N-th cycle, and the previous strategy (repeating cycles and majority voting) can be used for (N-1)-th, N-th, and the two correction cycles to reduce as much as possible the error probabilities of these four last decisions.

No settling errors



0 1 0 1 0 1 = 5 OK

Incomplete settling of first decision level



1 0 0 0 1 0 = 5 OK

Radix=2 design with redundant decision levels

The strategy is to determine one or more MSBs using one extra decision level to create overlapping trajectories, keeping the standard convergence rate of 2/cycle.

This can be done by using two comparators instead of one in order to resolve 1.5b instead of 1b, and adding a very simple reconstruction logic based on full-adders.

In each cycle of the conversion the comparator thresholds are shifted by a number of LSB that represents the redundancy in this cycle.

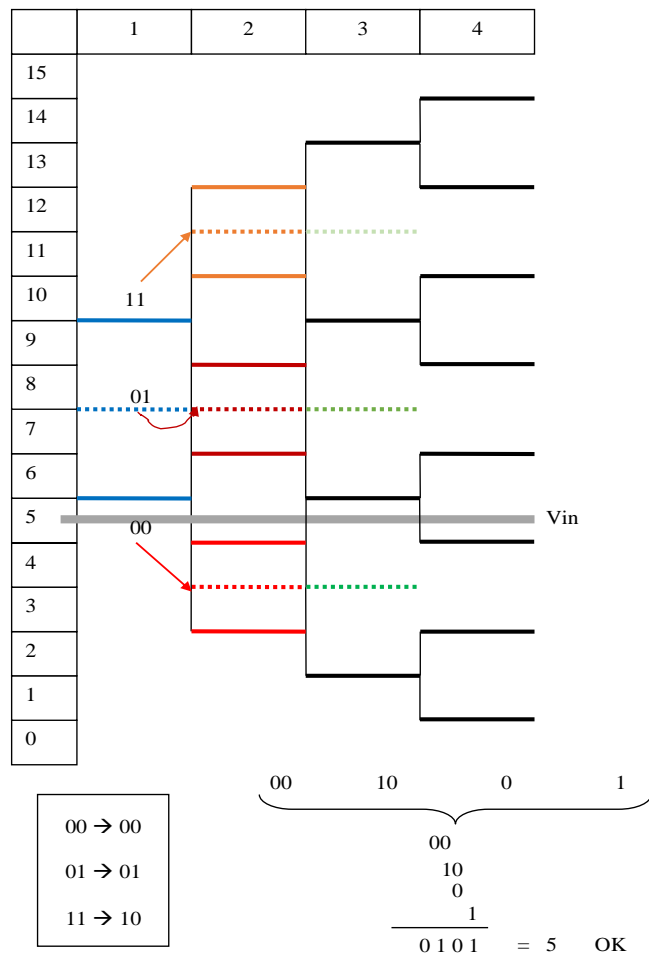
- if errors lower than $\pm k$ LSBs, occurring in the determination of the first MSB, have to be corrected, two comparators with threshold shifts of $+k$ LSBs and $-k$ LSBs are used in the first cycle of the conversion.
- if errors lower than $\pm k$ LSBs, occurring in the determination of the first MSB, and errors lower than $\pm h$ LSB, occurring in the determination of the second MSB, have to be corrected, two comparators with threshold shifts of $+k$ LSBs and $-k$ LSBs are used in the first cycle of the conversion, and the same comparators with threshold shifts of $+h$ LSBs and $-h$ LSBs are used in the second cycle of the conversion.
- If errors in LSBs linearly related with the step sizes of the decision levels have to be corrected, two comparators are used on the first, second, third, ... MSB determinations with built-in threshold shifts of k LSBs, $(K/2)$ LSBs, $(K/4)$ LSBs ..., respectively [19].

This kind of (large) redundancy is exploited to cope with incomplete decision level settlements (especially the MSB one) when conversion rate has to be increase w/o an adequate increase of the power consumption.

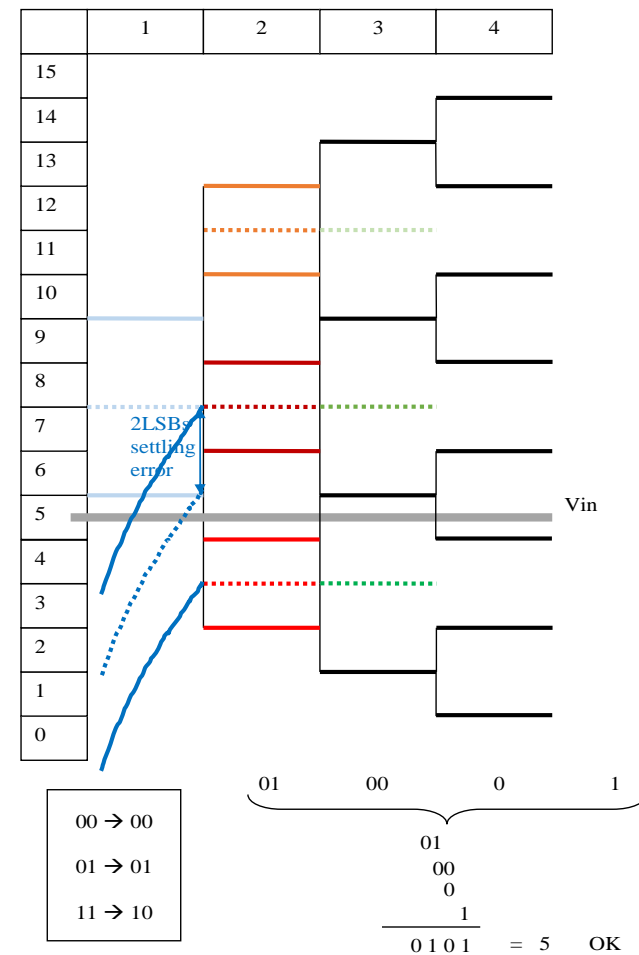
Ex.1: incomplete settling < 1/8 Full-Scale

If both comparators decide '1' decision level steps up, if both comparators decide '0' decision level steps down, if one comparator decide '0' and the other '1' the decision level does not change. In all cases the comparator threshold shift is halved until it disappears when below 1LSB.

No settling errors



Incomplete settling of first decision level



Radix<2 design with redundancy in search-space convergence

The strategy is to use M steps for N bit resolution, where $M > N$. In this way non-binary search algorithms present 2^M comparison patterns and 2^N output levels. In other words, for a given output level there can be multiple comparison patterns, which means that there is some redundancy. Thus, even if the comparator decision takes a mistake at some stage, it is possible to have correct ADC output.

Non-binary algorithms are used with radix of $2^{N/M} < 2$ [20], but recently this restriction has been avoided and a generalized non-binary algorithm with integer step sizes has been presented [21].

- If $p(k)$ is the value of k -th step size, it must be satisfied the following:

$$p(1) = 2^{N-1} \quad \sum_{i=1}^M p(i) = 2^N - 1$$

if $M=N$ and $p(k)=2^{N-k}$, it is equivalent to the binary search algorithm.

if $p(k)=\beta^{M-k}$ with $\beta=2^{N/M}$, it is the conventional non-binary search algorithm with radix β .

- Redundancy $q(k)$ can be set for each comparison step k , and can be defined as follows:

$$q(k) = -p(k+1) + 1 + \sum_{i=k+2}^M p(i)$$

This formula can be intuitively understood as follows: for the k -th output bit, once decision is made, the next decision level will move up or down by the step size of $p(k+1)$. If this decision is erroneous then the sum of the follow-on step sizes $p(k+2)$ $p(k+3)$ $p(M)$ and 1 must be larger than the value of $p(k+1)$ to counteract this mistake. The exceeded amount is the redundancy for that comparison step.

Radix<2 design with redundancy (cont'd)

Design method for generalized non-binary algorithm:

a) Design for redundancy at each step:

$$2^M - 2^N = \sum_{k=1}^{M-1} 2^k q(k) \quad \Rightarrow \quad \text{choose } q(k), k=1, 2, \dots, M-1$$

b) Calculate the required step:

$$p(k+1) = -q(k) + 2^{M-k-1} - \sum_{i=k+1}^{M-1} 2^{i-k-1} q(i)$$

c) Calculate the N-bit output code D_{out} from the M-bit conversion code B:

$$D_{out} = 2^{N-1} + \sum_{i=2}^M [2b(i-1) - 1] \cdot p(i) + [b(M) - 1] = \sum_{i=2}^M b(i-1) \cdot 2p(i) + b(M)$$

Ex.1: incomplete settling < 1/4 Full-Scale

:

a) Choose redundancy at each step:

N=5 , M=6

$$2^M - 2^N = \sum_{k=1}^{M-1} 2^k q(k) \implies 64 - 32 = \sum_{k=1}^{M-1} 2^k q(k) \implies q(1)=8, q(2)=4, q(3)=0, q(4)=0, q(5)=0$$

b) Calculate the required step:

$$p(1) = 2^{N-1} = 16$$

$$p(k+1) = -q(k) + 2^{M-k-1} - \sum_{i=k+1}^{M-1} 2^{i-k-1} q(i)$$



$$k=1 \rightarrow p(2) = -q(1) + 2^{6-2} - \sum_{i=2}^5 2^{i-2} q(i) = -q(1) + 16 - [q(2) + 2q(3) + 2^2 q(4) + 2^3 q(5)] = -8 + 16 - 4 = 4$$

$$k=2 \rightarrow p(3) = -q(2) + 2^{6-3} - \sum_{i=3}^5 2^{i-3} q(i) = -q(2) + 8 - [q(3) + 2q(4) + 2^2 q(5)] = -4 + 8 - 0 = 4$$

$$k=3 \rightarrow p(4) = -q(3) + 2^{6-4} - \sum_{i=4}^5 2^{i-4} q(i) = -q(3) + 4 - [q(4) + 2q(5)] = 0 + 4 - 0 = 4$$

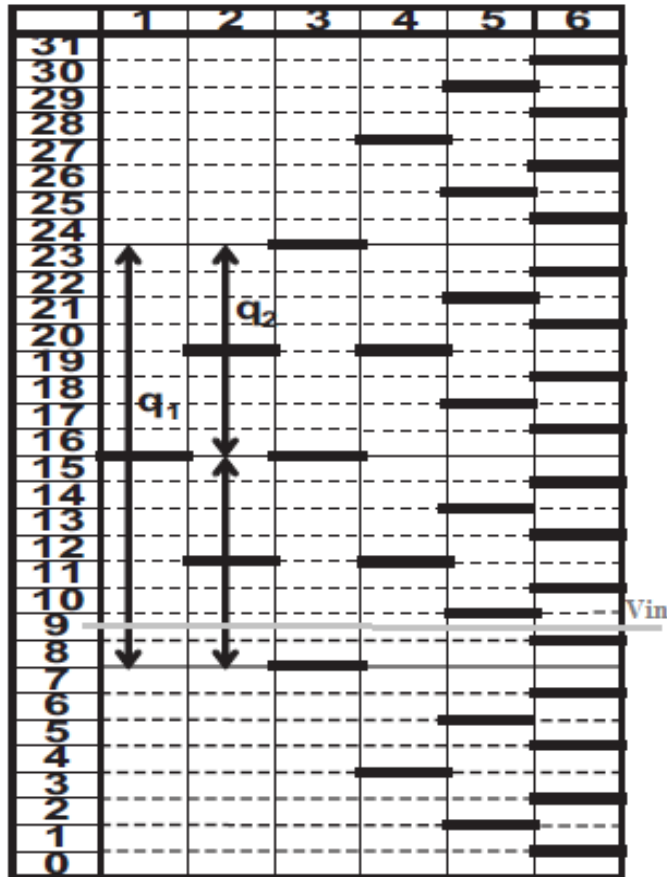
$$k=4 \rightarrow p(5) = -q(4) + 2^{6-5} - \sum_{i=5}^5 2^{i-5} q(i) = -q(4) + 2 - [q(5)] = 0 + 2 - 0 = 2$$

$$k=5 \rightarrow p(6) = -q(5) + 2^{6-6} = -q(5) + 1 = 0 + 1 = 1$$

Ex.1: incomplete settling < 1/4 Full-Scale (cont'd)

c) Calculate the N-bit output code D_{out} from the M-bit conversion code B:

No settling errors



$$P(k) = [16 \quad 4 \quad 4 \quad 4 \quad 2 \quad 1]$$

$$B(i) = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1]$$

$\rightarrow b(M)$

$$D_{out} = \sum_{i=2}^M b(i-1) \cdot 2p(i) + b(M)$$

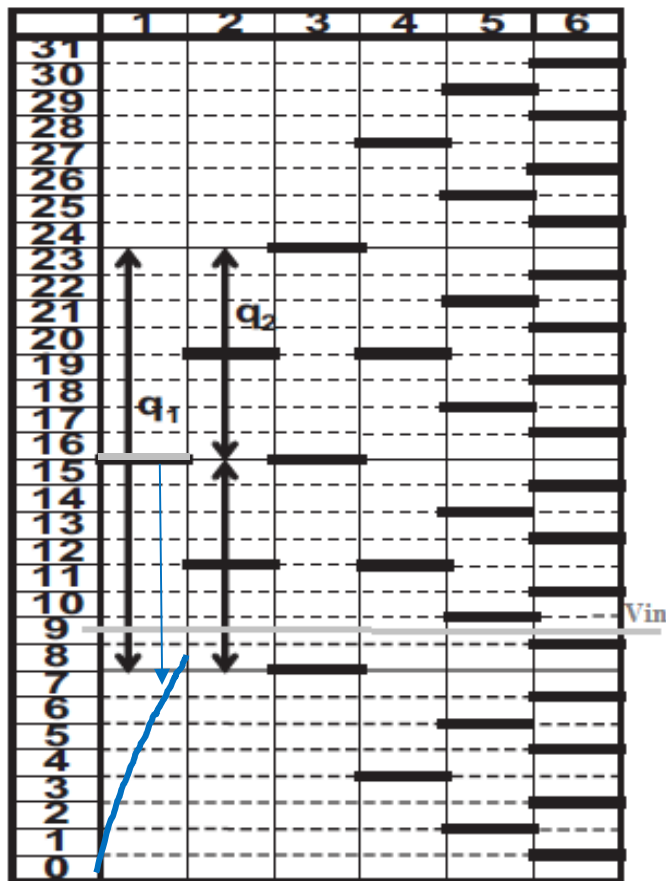


$$D_{out} = 0 \cdot 2 \cdot 4 + 0 \cdot 2 \cdot 4 + 1 \cdot 2 \cdot 4 + 0 \cdot 2 \cdot 2 + 0 \cdot 2 \cdot 1 + 1 = 9 \quad \text{OK}$$

Ex.1: incomplete settling < 1/4 Full-Scale (cont'd)

c) Calculate the N-bit output code D_{out} from the M-bit conversion code B:

Incomplete settling (8LSB error) of first decision level



$$P(k) = [16 \ 4 \ 4 \ 4 \ 2 \ 1]$$

$$B(i) = [1 \ 0 \ 0 \ 0 \ 0 \ 1]$$

$b(M)$

$$D_{out} = \sum_{i=2}^M b(i-1) \cdot 2p(i) + b(M)$$



$$D_{out} = 1 \cdot 2 \cdot 4 + 0 \cdot 2 \cdot 4 + 0 \cdot 2 \cdot 4 + 0 \cdot 2 \cdot 2 + 0 \cdot 2 \cdot 1 + 1 = 9 \quad \text{OK}$$

Settling of DAC output

Since large settling errors can be tolerated, generalized non-binary algorithm are faster than conventional non-binary ones, and much faster than classical binary ones.

BINARY SEARCH ALGORITHM

STEP 1	STEP 2	STEP 3	STEP N-1	STEP N
--------	--------	--------	-------	----------	--------

Exact DAC settling in each step → long time

NON-BINARY SEARCH ALGORITHM

STEP 1	STEP 2	STEP 3	...	STEP M-1	STEP M
--------	--------	--------	-----	----------	--------

Incomplete DAC settling → short time

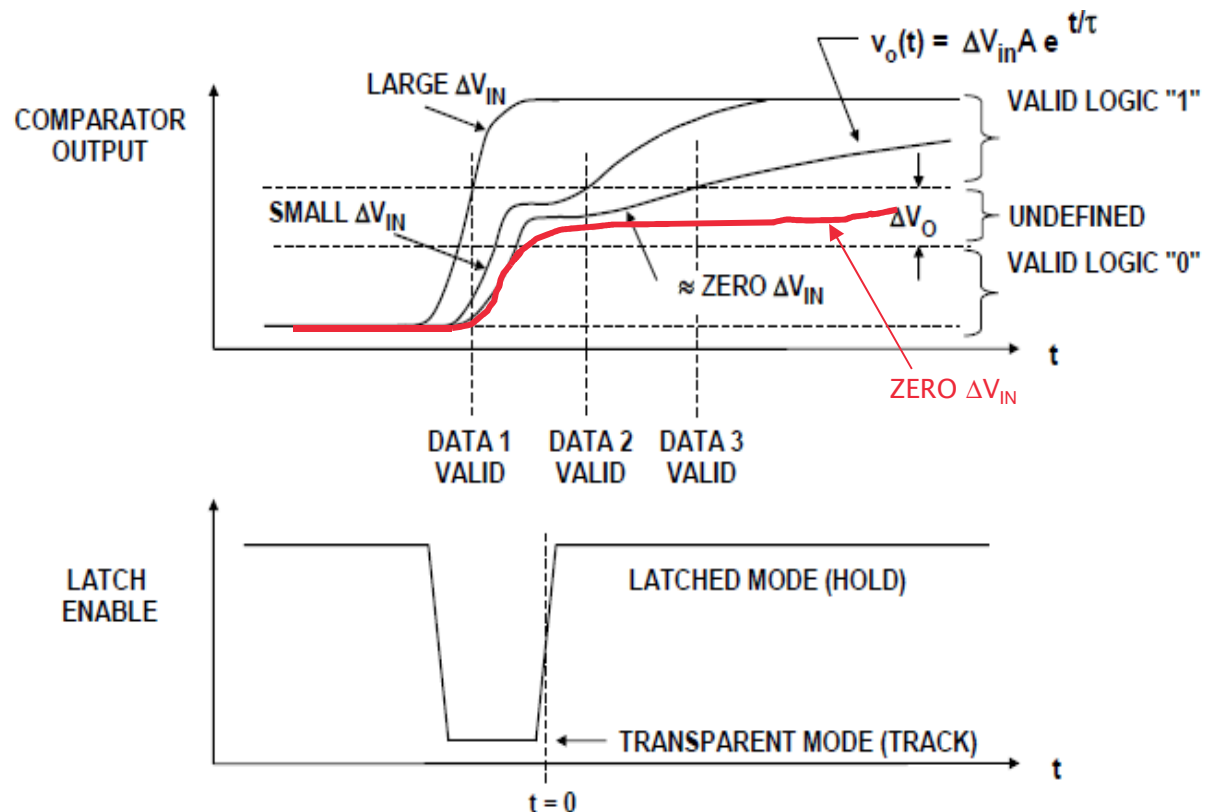
14b SAR ADC CONVERSION SPEED COMPARISON

Algorithm	Binary	Conventional	Generalized
Time slot for each step	8.4τ	2.2τ	1.2τ
Number of steps	14	17	22
Total conversion time	109.2τ	35.2τ	25.2τ

Metastability in ADCs

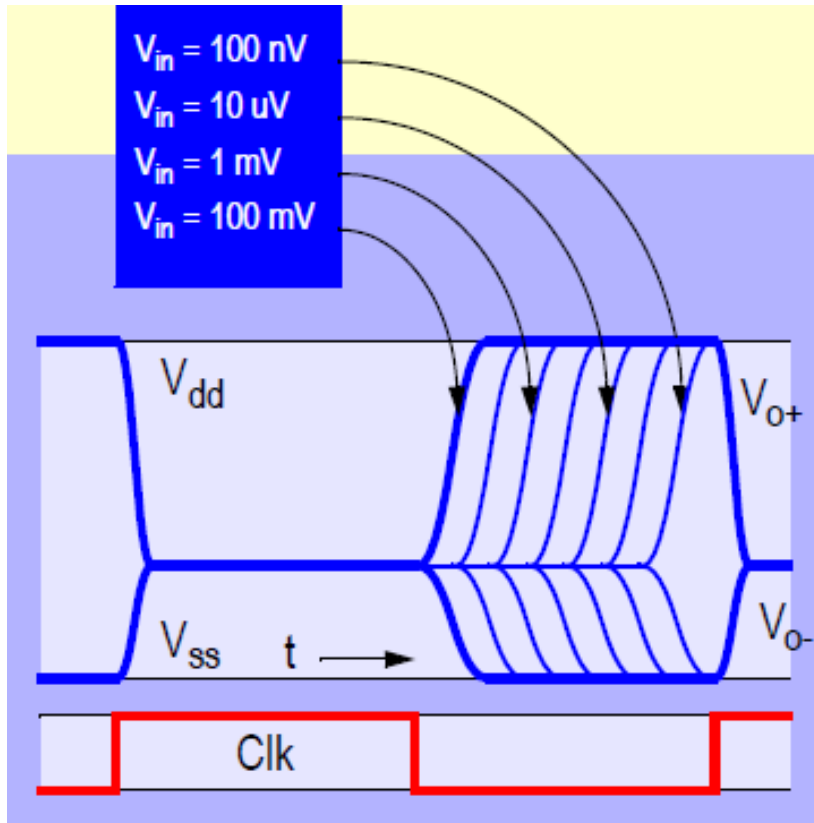
Metastability caused by comparators (I)

A latched comparator is said to be in a metastable state if it is not able to produce a valid decision, i.e. a decision that can be interpreted by subsequent digital logic, in an available time.



Metastability caused by comparators (II)

- Bit Error Rate (BER) example -



$$1\text{LSB} = 100\text{mV}$$

Min input signal achieving full output swing with the available time is 100nV

$$\text{BER} = 100\text{nV} / 100\text{mV} = 10^{-6}$$

Metastability caused by comparators (III)

This issue is usually severe in high-speed SAR ADCs due to their serial conversion scheme, which includes the regeneration and the reset process of the comparator in a feedback loop, thus significantly reducing the available time for the regeneration.

Analysis of the behavior of metastability has shown that the probability of the output remaining in a metastable region decreases exponentially with time. This is the reason why timer-based, i.e. with loop time-out, asynchronous ADCs are less prone to metastability than synchronous ones.

Any comparator has a finite probability to enter in a metastable state, probability that can be made very low through proper circuit design techniques but never made equal to zero.

If these errors were confined to the least significant bit, the consequent problems would not be serious, but it can be shown that metastability in the most significant bit of a SAR ADC can result in an error of a quarter of full scale.

Performance limitations in ADCs

Performance limitations in ADCs

- Sampling switch kT/C noise
- Sampling distortion
- Sampling switch charge injection

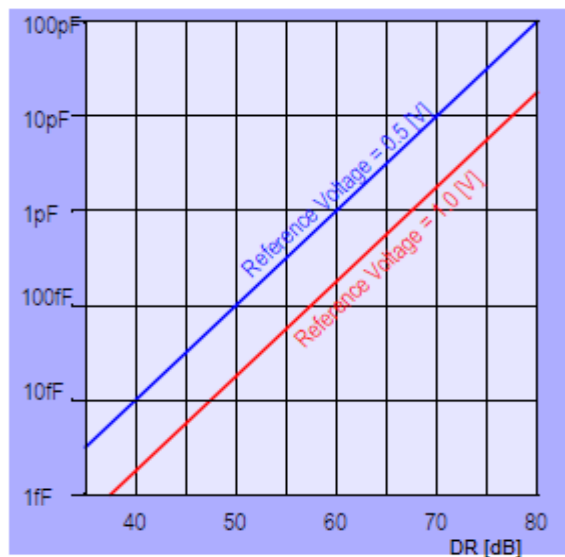
Sampling switch kT/C noise

Sampling capacitor is usually chosen to have a kT/C noise lower than the quantization noise with some noise margin (NM), unless its noise is deliberately used as dithering.

$$DR = \frac{V_{in_max}}{\sqrt{NM \frac{kT}{C_{TOT}}}} = \frac{\frac{V_R}{2\sqrt{2}}}{\sqrt{NM \frac{kT}{C_{TOT}}}}$$

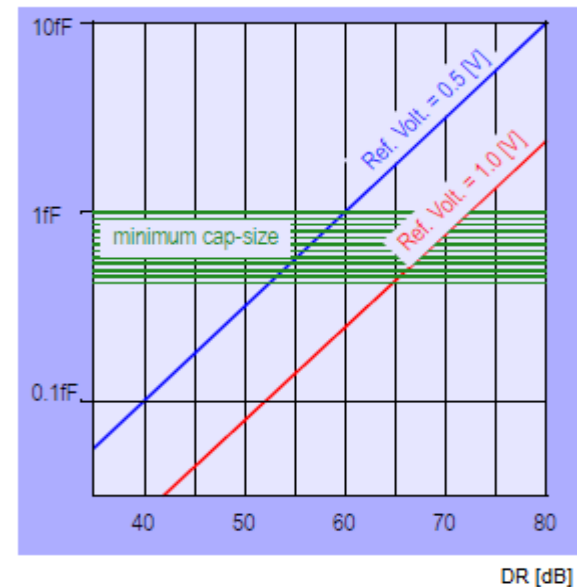
$$C_{TOT} = \frac{8kT DR^2}{V_R^2} NM$$

C_{TOT}



$$C_{LSB} = \frac{C_{TOT}}{2^N} = \frac{C_{TOT}}{DR}$$

C_{LSB}



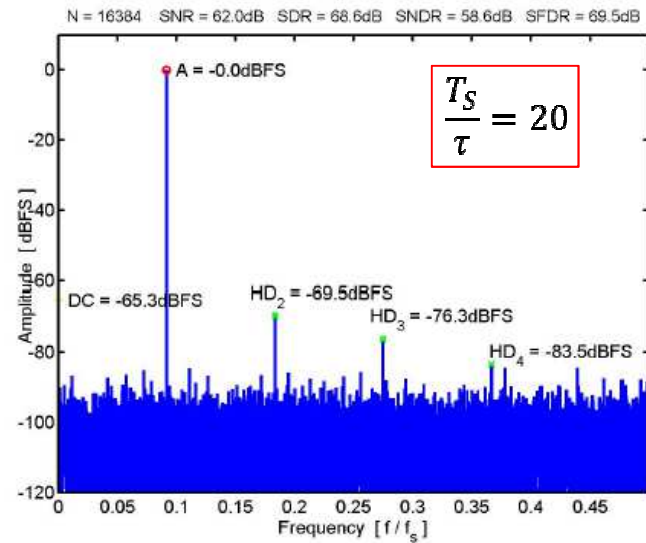
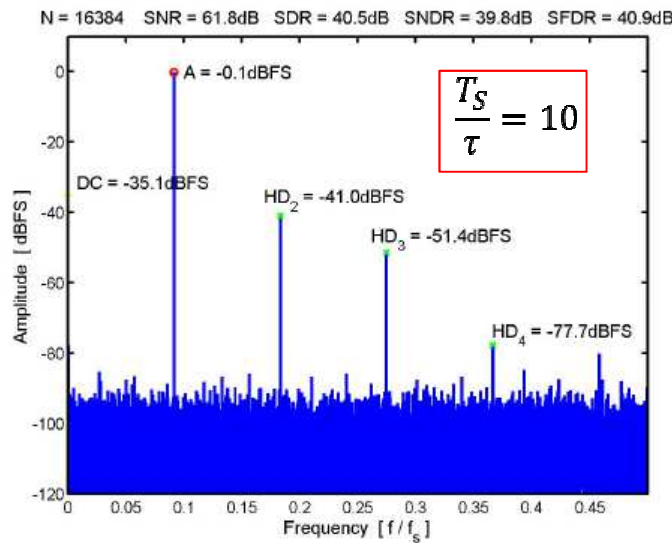
Sampling distortion (I)

Settling time of the voltage across sampling capacitor is impacted by finite non-linear switch resistance and must be commensurate to ADC resolution to avoid sampling distortion.

$$\tau \ll \frac{T_S}{2} \frac{1}{\ln 2^N} \implies R_{SW} \ll \frac{T_S}{2C} \frac{1}{\ln 2^N}$$

$$R_{SW} = \frac{1}{\mu C_{OX} \frac{W}{L} (V_{GS} - V_{TH})} = \frac{1}{\mu C_{OX} \frac{W}{L} (V_{CC} - V_{in} - V_{TH})}$$

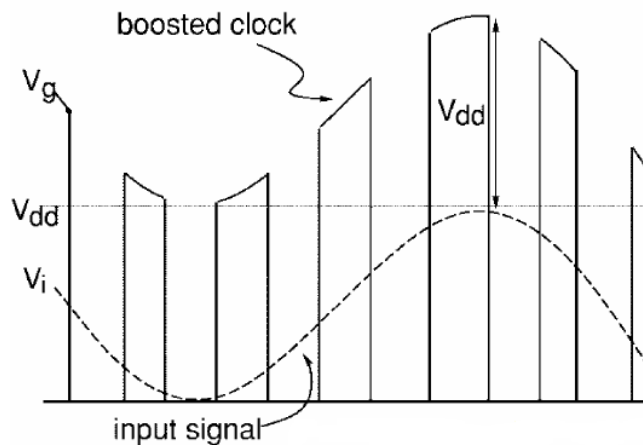
Example 10b ADC



Sampling distortion (II)

Solutions:

- increase switch control voltage $V_C \rightarrow$ HD2 decreases by (V_{C1}/V_{C2}) ; HD3 decreases by $(V_{C1}/V_{C2})^2$
- clock voltage multiplier needed.
- increase switch width \rightarrow distortion largely decreases
- switch charge injection and non-linear S/D junction capacitances increase.
- constant and maximum V_{GS} sampling \rightarrow distortion strongly reduced
- clock boosting needed.



Sampling switch charge injection (I)

Switch charge injection \rightarrow switch charge sharing + clock feedthrough

Clock fall time \gg MOS speed \rightarrow channel charge flows into low impedance signal source
 \rightarrow clock feedthrough is the only source of error.

Clock fall time \ll MOS speed \rightarrow channel charge divides equally between source and drain
 \rightarrow half channel charge + clock feedthrough are the sources of error.

In general (and real) cases error/distortion is function of many parameters:

- clock fall time
- input voltage level
- source impedance
- sampling capacitance
- switch size and junction capacitances

Sampling switch charge injection (II)

Solutions:

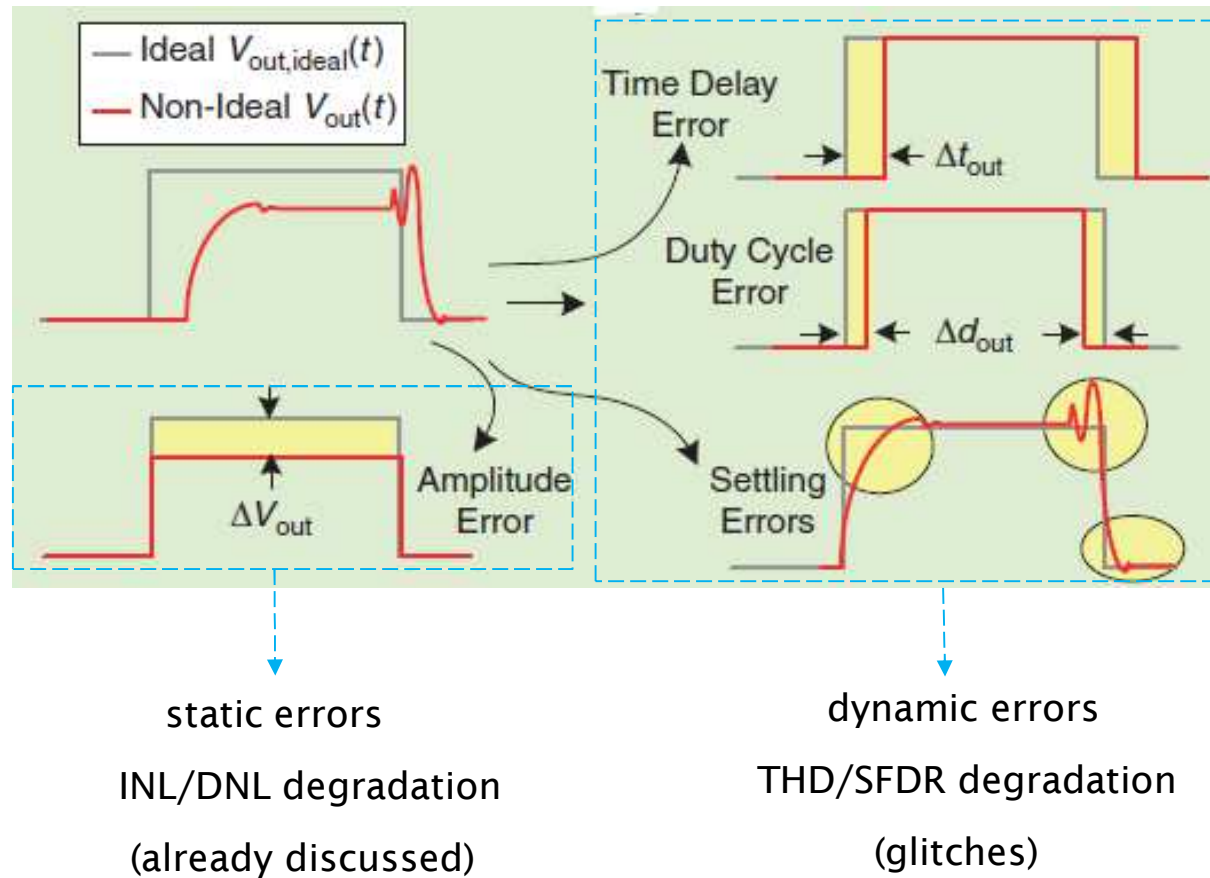
- reduce switch width → charge injection is reduced linearly but
 - switch time constant is increased → sampling distortion increases → not a viable solution.
- complementary switches → almost no impact when input signal is close to its max and min values and minor improvements when input signal moves around its average value (p-chs and n-chs have different widths and threshold voltages) → not a viable solution.
- addition of dummy switches → effective only if exactly half of the charge goes to the dummy
 - need to create the same environment on both sides, i.e. same input/output node impedance
 - good matching between clock fall/rise is required
 - limited by switch mismatches.
- Use fully-differential structures (do not use dummy switches in this case).
- Use bottom-plate sampling → the reset switch of the comparator input node opens first
 - no signal dependent charge injected by sampling switch
 - constant charge can be eliminated using differential switching.

Dynamic performance limitations in DACs

Dynamic performance limitations in DACs

- Glitch definition
- Glitch effects
- Main glitch causes and possible improvements
- Compensation techniques

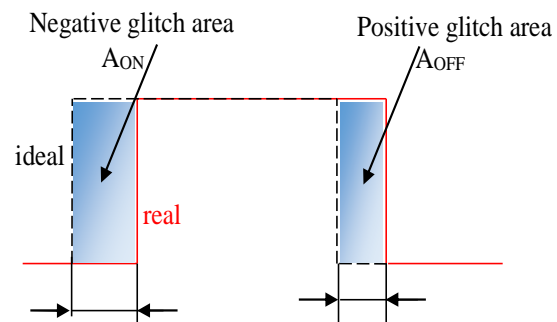
DAC waveform errors



Glitch definition

Let's define glitch as the area difference between real and ideal steps (voltage, current, or charge), whatever can be its origin.

Absolute glitch definition



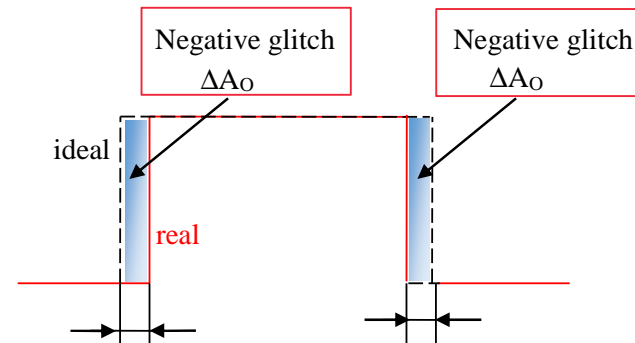
$$A_{ON} \neq A_{OFF}$$



$$A_{ON} = A_O + \Delta A_O$$

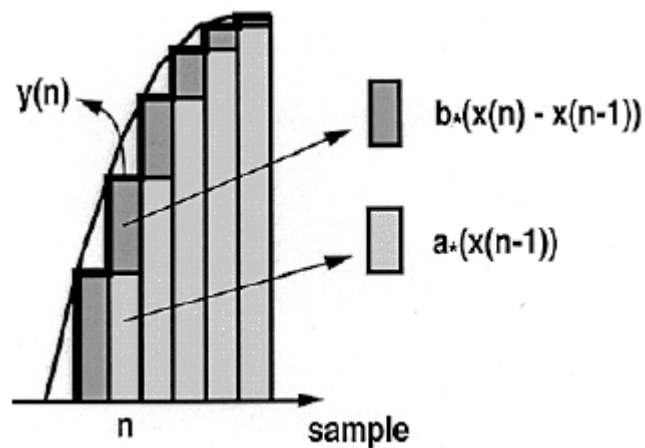
$$A_{OFF} = A_O - \Delta A_O$$

Relative glitch definition



Glitch effects (I)

If $A_{ON} = A_{OFF}$, i.e. $\Delta A_O = 0$, glitches result to only a small filter action on the signal in both thermometer-coded (or segmented) and binary-weighted architectures.

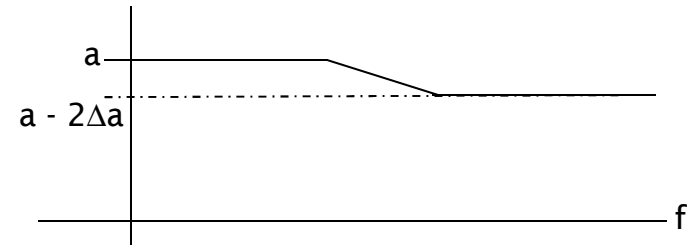


$$y(n) = a \cdot x(n-1) + b \cdot [x(n) - x(n-1)]$$

If $b=a \rightarrow$ no glitches $\rightarrow y(n) = a \cdot x(n)$

If $b=a-\Delta a \rightarrow$ equal positive and negative glitches $\rightarrow y(n) = a \cdot x(n) - \Delta a \cdot [x(n) - x(n-1)]$

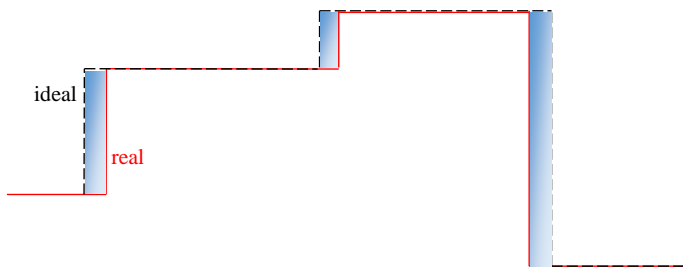
$$\frac{Y(z)}{X(z)} = a - \Delta a \cdot (1 - z^{-1})$$



Glitch effects (II)

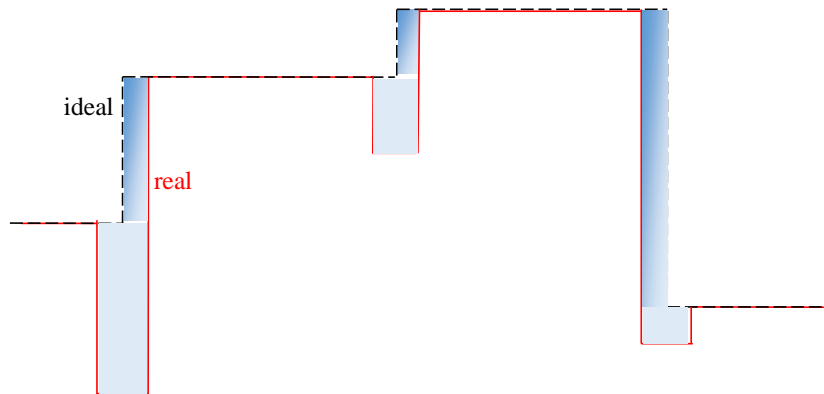
If $A_{ON} \neq A_{OFF}$, i.e. $\Delta A_O \neq 0$, glitches add harmonic and spurious tones on the signal, degrading the THD and SFDR !

Thermometer-coded architecture



Glitches are linearly related to the modulus of the first derivative of the signal, resulting to mainly THD and IMD.

Binary-weighted architecture



Glitches are non-linearly related to the signal value and its derivative resulting to much higher THD, IMD, and spurious tones.

Things become worse if signal frequency increases because its derivative is higher, and if sampling frequency increases since the ratio between glitch and signal areas is larger.

Main glitch causes and possible improvements (I)

a) The imperfect synchronization (time skew) of the control signals of the switches.



Place a synchronization block (latch) immediately in front of the switches (*) in order to circumvent any delay introduced by the digital decoding logic. At layout level, assure that the connections between latch and switching transistors are identical.

b) The digital signal feedthrough via the C_{DG} of the switch transistors in current steering DACs.

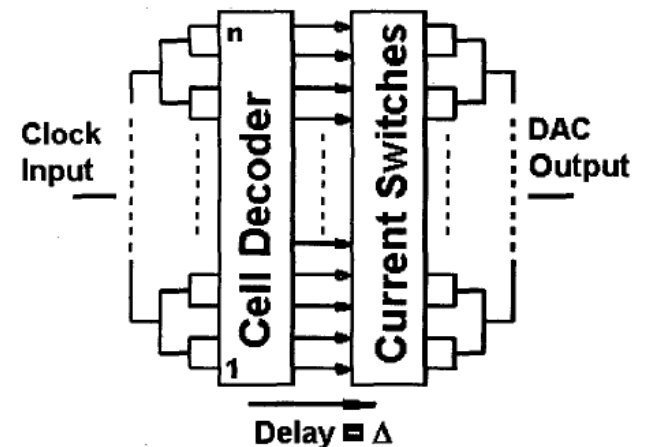


Use switching transistors as small as possible, possibly in saturation region; reduce the voltage swing at the gates of the switches; use dummy switching transistors for first order compensation.

(*) Pay attention to match clock delays by exploiting binary tree distribution of the clock.

Pay attention to data dependent clock loading (clock driver sees different loads into the latches depending whether the content of the latch is changing or not) that leads to code-dependent clock rise/fall times and switching times [22].

Moreover, if possible, use output collection tree from each cell.



Main glitch causes and possible improvements (II)

c) The voltage change at the drain of the current source transistors in current steering DACs.



If switches are simultaneously in the off-state for a short period of time during switching, a discharge of the drain capacitance of the current source transistors takes place leading to a deterioration of the dynamic performance. This problem is reduced by using a cascoded transistor for each current source, and special switch driver circuits that assure a short on-on of switching transistors. To “eliminate” it, differential quadrature switching (DQS) can be used [22].

d) The output voltage coupling to the source node of the switches in current steering DACs.



The signal attenuation of the output swing seen at the common source node of the differential switches is given by the switch $g_m r_{ds}$ (typical values of this ratio are about 20). This coupling modulates the switch bias voltage, resulting to a code-dependent switching time. This problem can be alleviated by using switching transistors in saturation region.

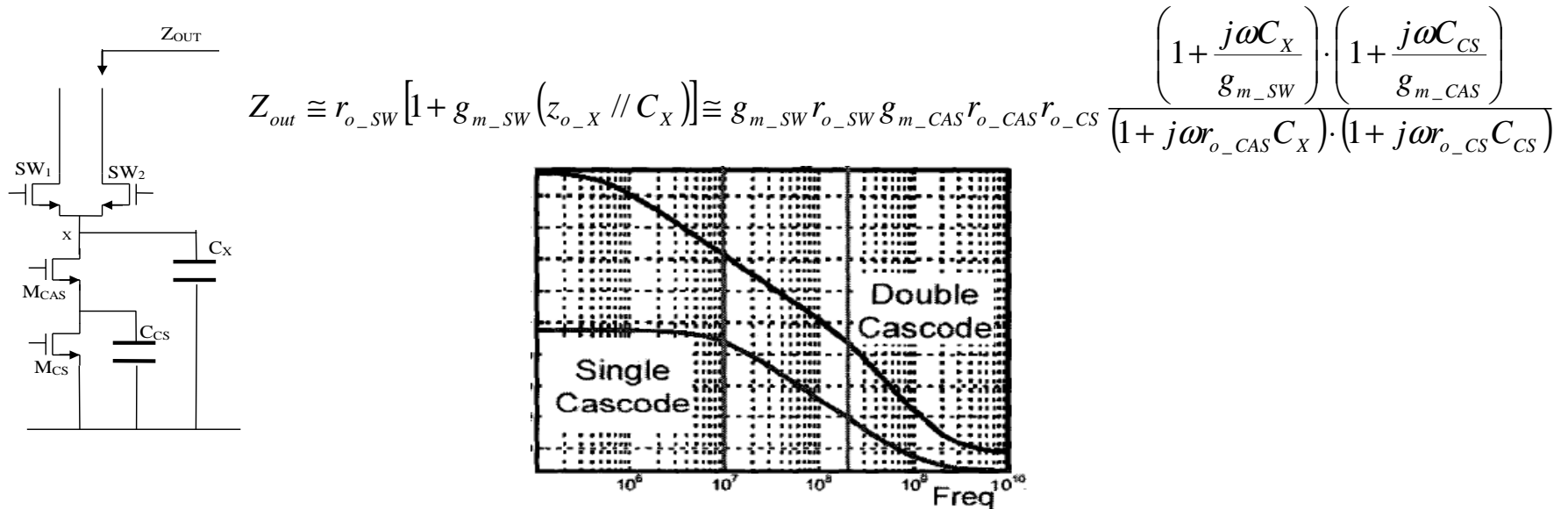
e) The code-dependent settling time constants.



Replicate the elementary unity cell (capacitors, switches and drivers in case of capacitive DACs) or add extra resistors in case of resistive DAC [7] in order to balance as much as possible the time constants independently from the selected code.

Z_{out} frequency dependency and possible improvements

In current steering DACs, the output impedance Z_{out} has frequency dependency (reduction)



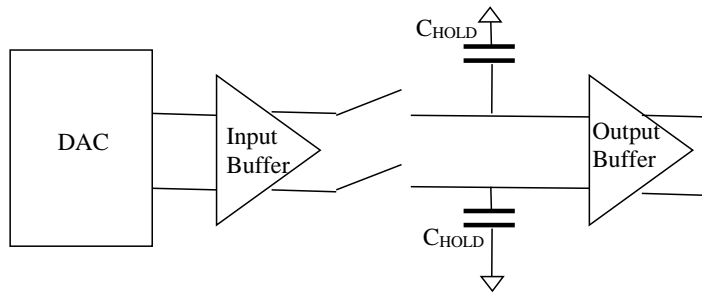
It can be demonstrated that, in case of full swing output, harmonic distortions are given by [23]:

$$\begin{aligned} \text{Single-ended case} & \left\{ \begin{aligned} \frac{HD2}{S} &\cong \frac{R_L N}{4Z_{out}} && \text{Single tone input} \\ \frac{IM2}{S} &\cong 2 \frac{R_L N}{4Z_{out}} && \text{Two tone input} \end{aligned} \right. \\ \text{Differential case} & \left\{ \begin{aligned} \frac{HD3}{S} &\cong \left(\frac{R_L N}{4Z_{out}} \right)^2 && \text{Single tone input} \\ \frac{IM3}{S} &\cong 3 \left(\frac{R_L N}{4Z_{out}} \right)^2 && \text{Two tone input} \end{aligned} \right. \end{aligned}$$

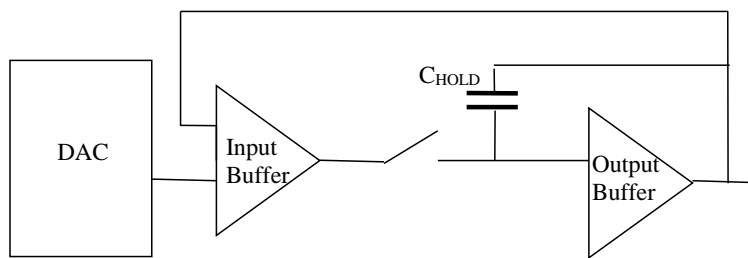
Compensation techniques: deglitching circuit

Since dynamic linearity problems are associated to the switching behavior, a track-and-Hold (T/H) circuit [24] can be used to hold the output constant while switching is occurring, and track once the current sources have settled to their steady state value. Only static DAC characteristics shows up at the output, and dynamic ones are eliminated.

The problem is that the T/H circuit introduces its own dynamic nonlinearities and settling time which ultimately limit DAC performance.



Nonlinear (open-loop) or fixed (closed-loop) pedestal errors when switch changes from track to hold.

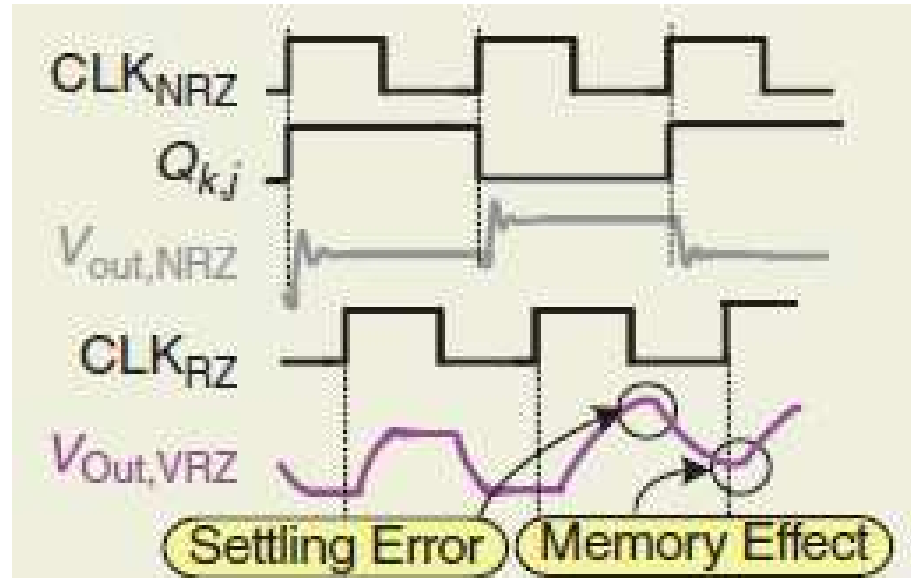
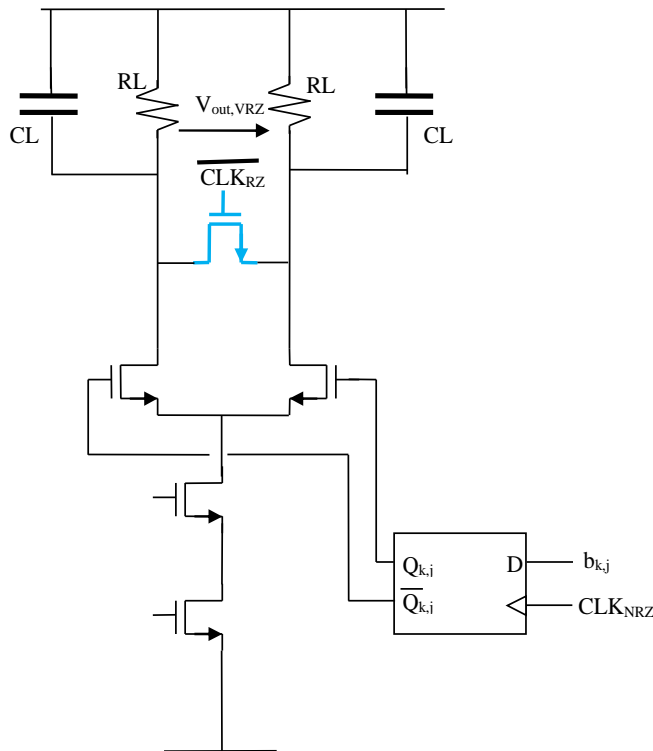


Nonlinear track mode errors (switch and buffers).

For these reasons, T/H circuit are less and less employed at the output of high-speed high-resolution DACs.

Compensation techniques: V return-to-zero switching

This technique avoids the use of buffers and hold capacitors in the output signal path [25].

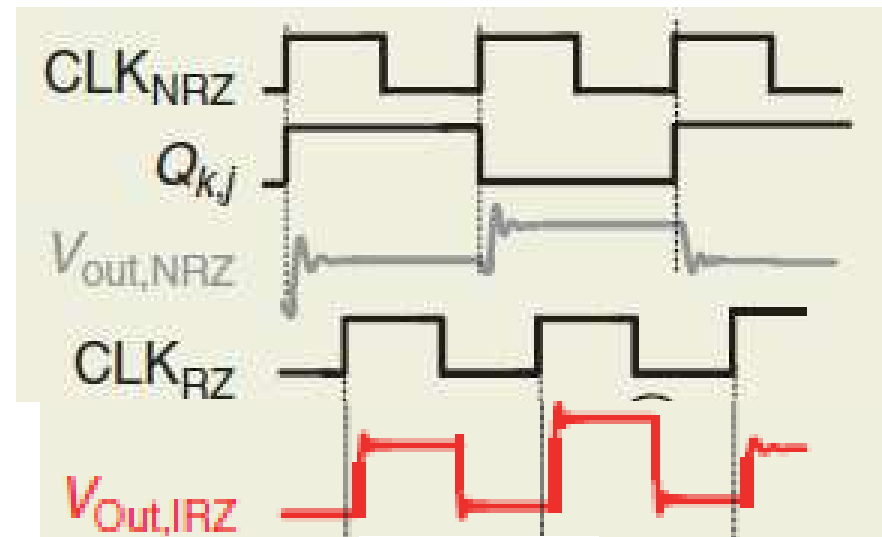
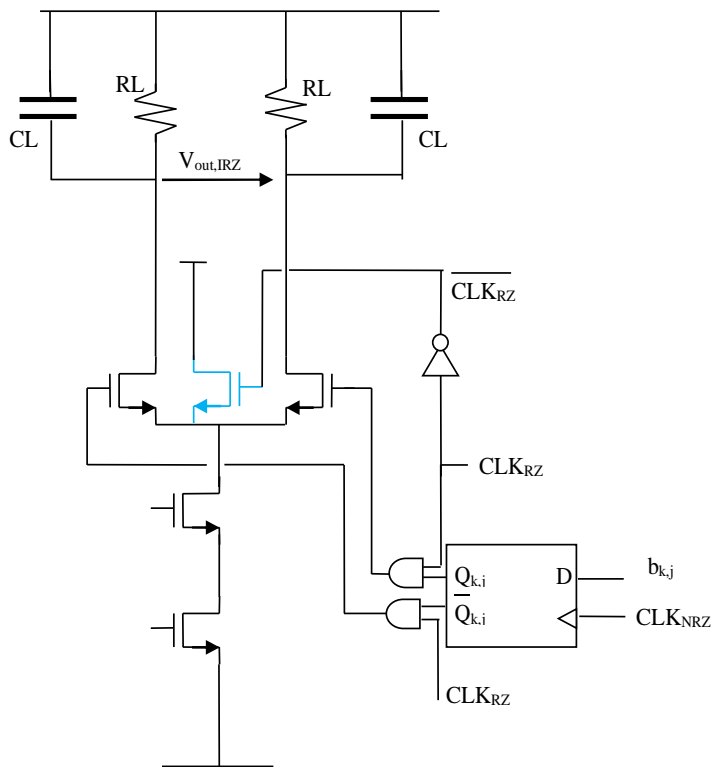


Based on the switch size, either the nonlinear switch ON resistance or capacitance dominates the output time constant. As a result, if the output does not fully charge to its desired value (settling error) or fail to completely reset to zero (memory effect), this data-dependent error ultimately limits the achievable SFDR.

There is a loss of 6dB on signal amplitude and it is more sensitive to clock jitter.

Compensation techniques: I return-to-zero switching

This technique not only avoids the use of buffers and hold capacitors in the output signal path, but also breaks the speed-linearity tradeoff of the V-RTZ approach at the expense of a more complex RZ clock distribution.



Since this technique requires the RZ clock to be routed to each individual cell, it introduces its own mismatch errors that ultimately limits the achievable SFDR.

There is a loss of 6dB on signal amplitude and it is more sensitive to clock jitter.

Power-supply decoupling

Best practices

1) Use separated analog and digital power-supply pins/ball.

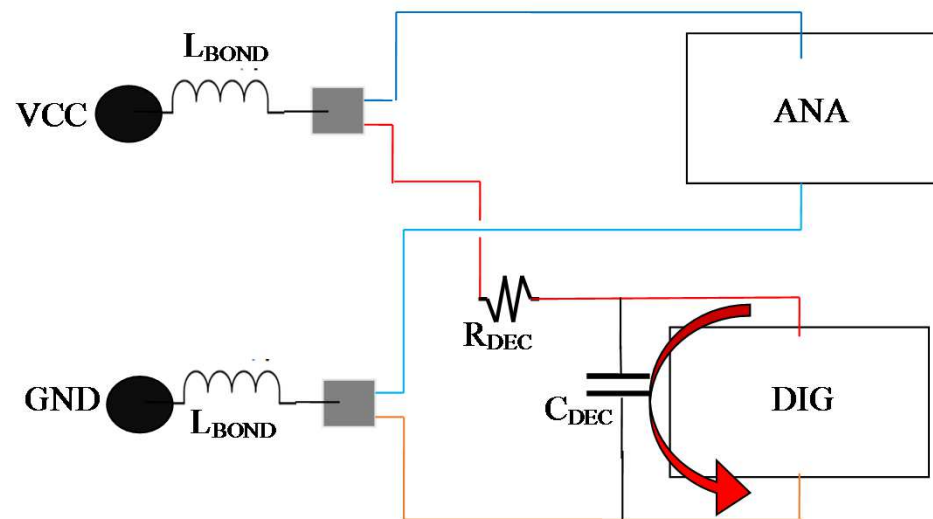
4 pins/balls → 4 pads and 4 bonding wires

2) Use separated analog and digital power-supply pads.

2 pins/balls → 4 pads and 4 bonding wires

3) Use star-routed analog and digital power-supplies with decoupling circuit

2 pins/balls → 2 pads and 2 bonding wires



Data converter validation

Static testing

- DAC testing [26]
- ADC testing
 - code boundary servo [26]
 - direct testing
 - histogram testing [26]

DAC testing

Very simple → apply digital codes and use a good voltmeter to measure the analog output:

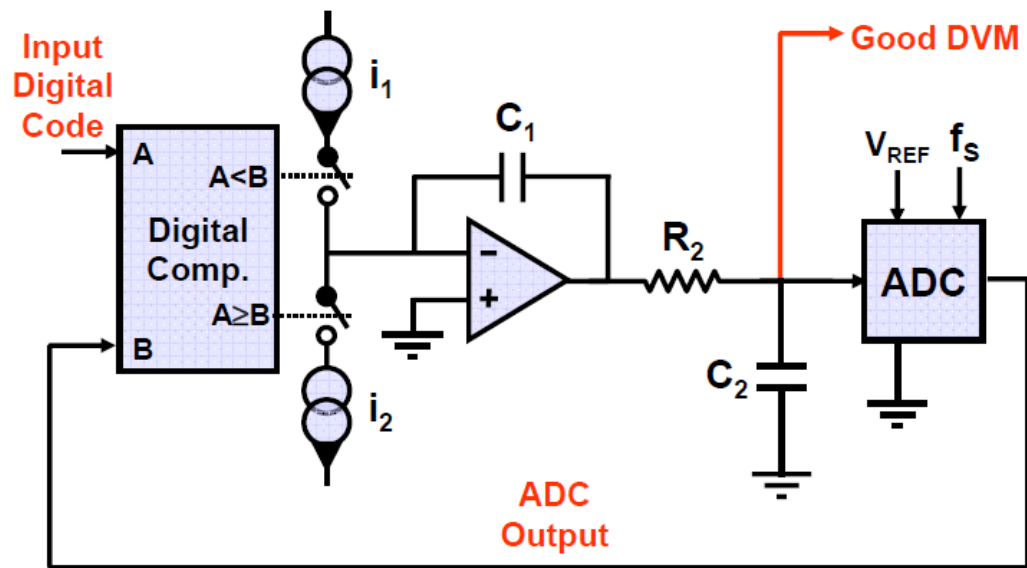
- many measurements for each code
- average measurements over multiple 50/60 Hz cycles to filter out ac line couplings

ADC testing

Not as simple as DACs → need to find transition levels, i.e. input voltages at all code boundaries.

- Code boundary servo → adjust ADC voltage source to find exact code trip points.
- Direct testing → use precision dc generator (calibrator) to force ADC input voltage.
- Histogram testing → Apply a signal with known distribution → analyze digital code distribution at ADC output → back calculate DNL and INL.

Code boundary servo

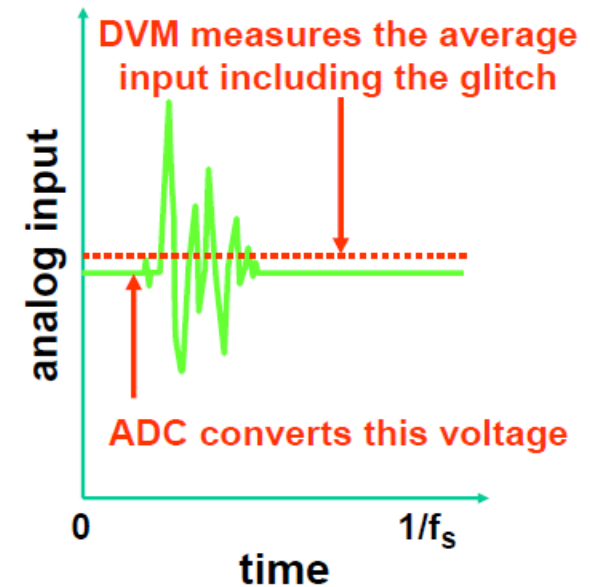
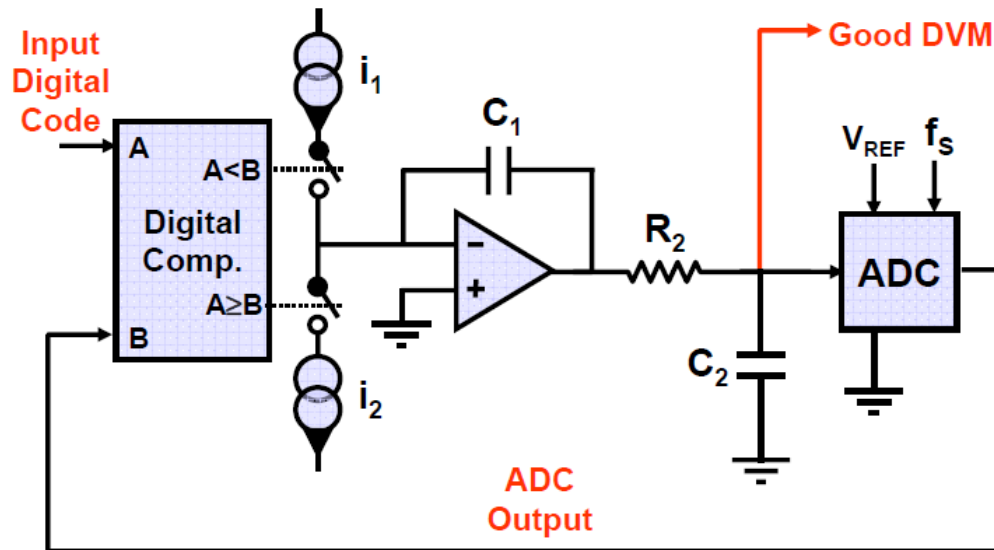


Very good digital voltmeter (DVM) measures the analog input voltage corresponding to the desired code boundary

- many measurements for each code
- average measurements over multiple 50/60 Hz cycles to filter out ac line couplings

i_1 and i_2 are small and C_1 is large in order that the ADC analog input can move only a small fraction of a LSB each sampling period and to minimize noise, resulting in a long integration time for the precision voltmeter.

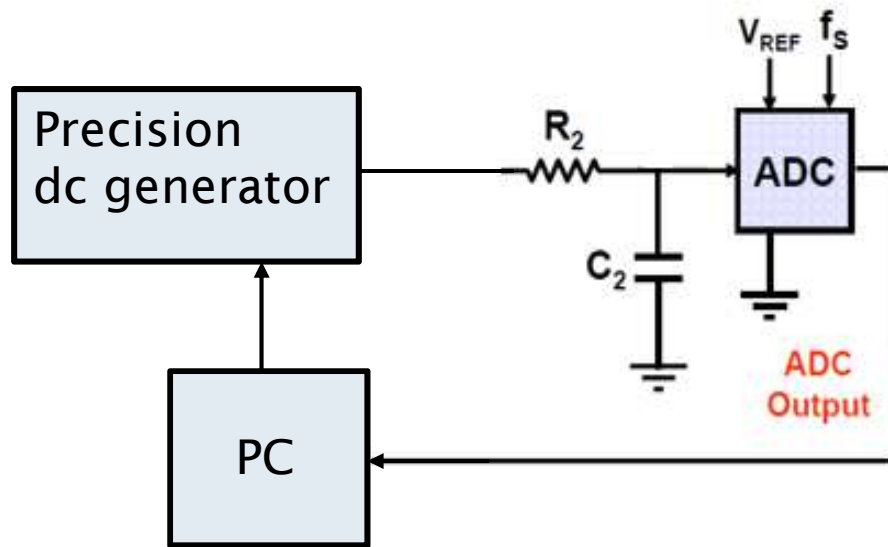
Code boundary servo issue



ADCs are notorious for kicking back high-frequency signal-dependent glitches to their inputs.

The difference between what ADC measures and what DVM measures is not ADC INL, but it is an error in the INL measurements → use a large C_2 to fix that at the expense of longer measurement time.

Direct testing



Precision dc generators can perform less than $10\mu\text{V}/\text{step}$ (6.5 digits)

- many measurements for each step
- average measurements over multiple 50/60 Hz cycles to filter out ac line couplings

Use a step size much lower than ADC LSB when sweep the generator over the FS range.
Code trip points are set when adjacent codes show equal probability (presence of ADC noise).

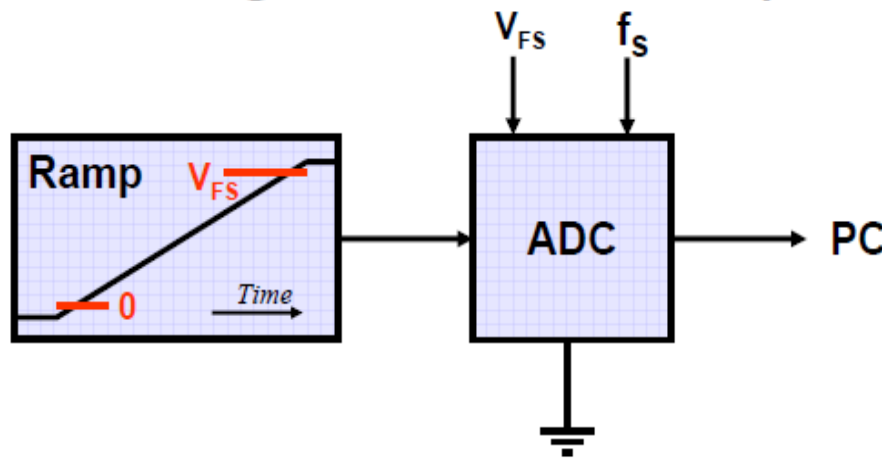
Very long measurement time.

Code boundary servo problem is avoided.

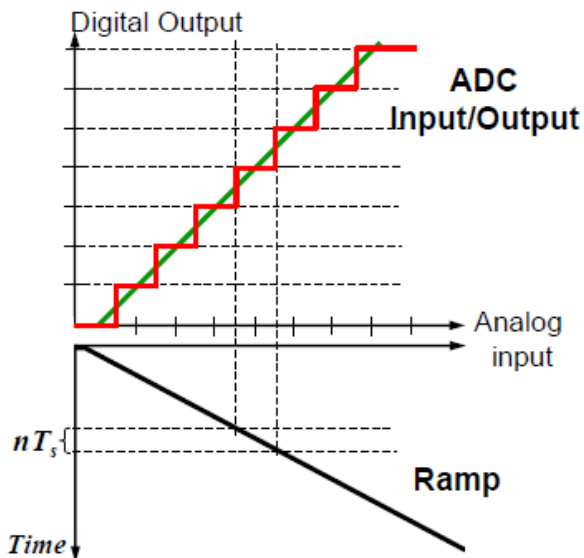
Histogram testing (I)

- ramp input signal -

For the histogram test to give accurate results, the input frequency must not be a sub-harmonic of the sampling frequency to avoid false missing codes and larger DNL errors.



Ramp slope is adjusted to provide a large number of output/code, e.g. 100 outputs of each code to get about 1/100 LSB resolution.



Example: 3b ADC

$$1\text{ LSB} = 10\text{mV}$$

$$\text{Ramp slope} = 10\mu\text{V}/\mu\text{s}$$

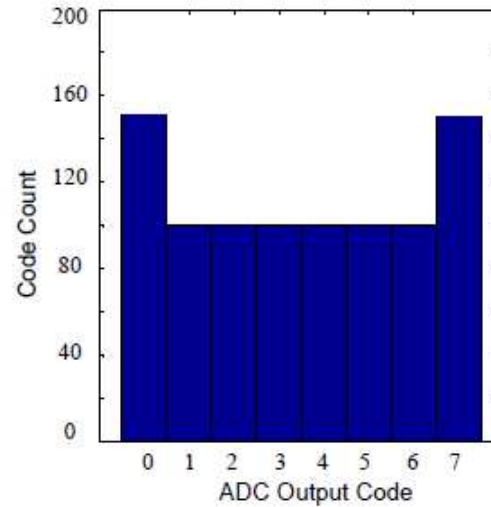
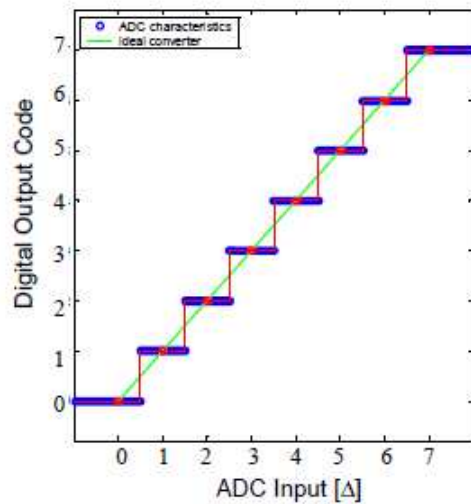
$$\text{Each ADC code} \rightarrow 1\text{ ms}$$

$$f_s = 100\text{kHz} \rightarrow T_s = 10\mu\text{s}$$

$$n = 1\text{ ms}/10\mu\text{s} = 100\text{samples/code}$$

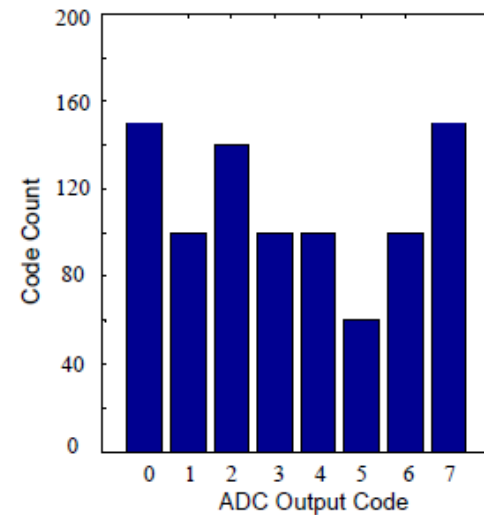
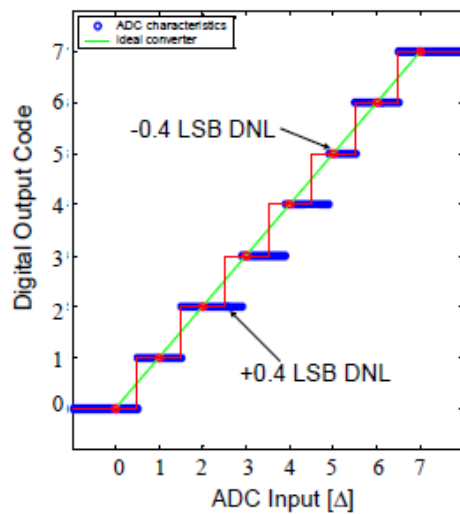
Histogram testing (II)

- ramp input signal -



Ramp histogram

ideal 3b ADC

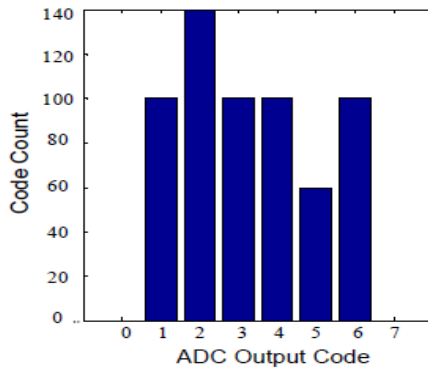


Ramp histogram

non-ideal 3b ADC

Histogram testing (III)

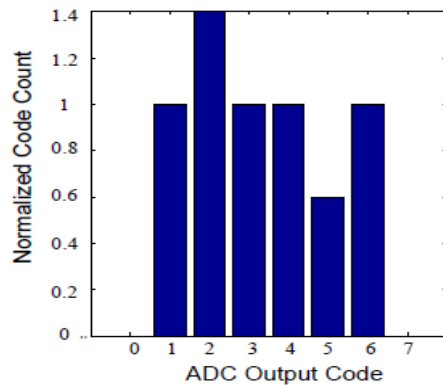
- ramp input signal -



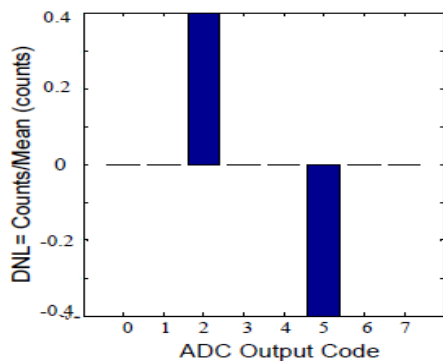
DNL extraction from histogram

Remove over-range bins from histogram (0 and full-scale)

Compute average count/bin (100 in this case)



Normalize, i.e. divide by average count/bin (ideal bins have exactly the average count, which after normalization is equal to 1)

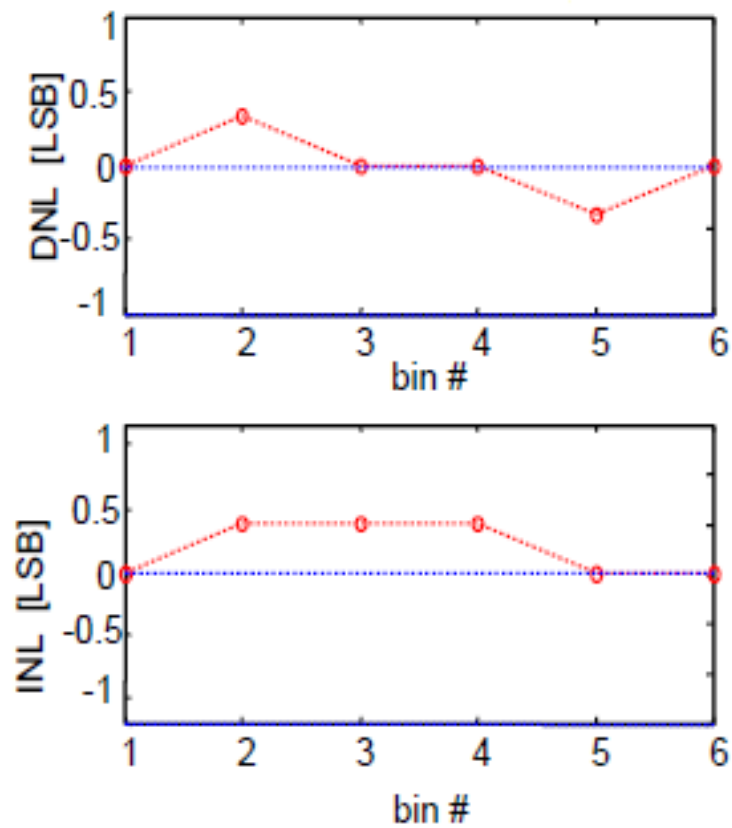


Subtract 1 from the normalized code count

Histogram testing (IV)

- ramp input signal -

DNL and INL extraction from histogram



Histogram testing (I)

- sinusoidal input signal -

- Very linear ramp generators are not readily available.
- High frequency noise, present on the triangular waveform, cannot be removed by filtering.



ramp linearity is affected.

For these reasons, a sinewave rather than a triangular waveform is often used as an input to the ADC when making histogram testing:

- Sinusoidal generators are more common
- Sinewaves can be generated with extremely high linearity and low noise with appropriate filtering.

Unlike the triangular wave, sinewave input does not yield equal probability for all codes:

- at sinewave midpoint crossings $\rightarrow dV/dt \rightarrow \max \rightarrow$ least number of samples
- At sinewave amplitude peaks $\rightarrow dV/dt \rightarrow \min \rightarrow$ highest number of samples

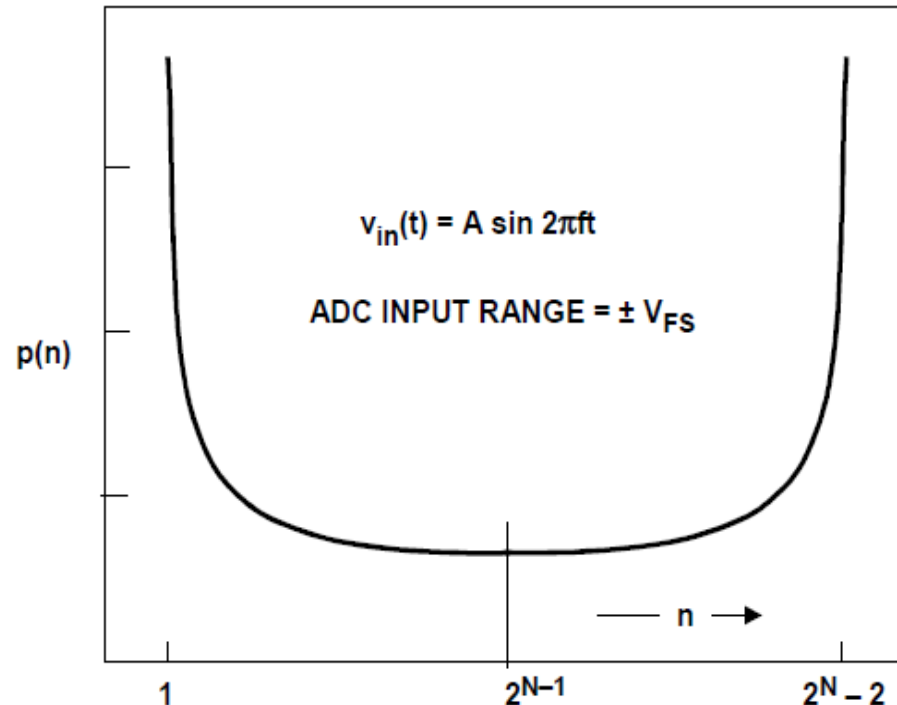


Need to correct for the sinewave probability density function (PDF) shape

Histogram testing (II)

- sinusoidal input signal -

Sinewave PDF



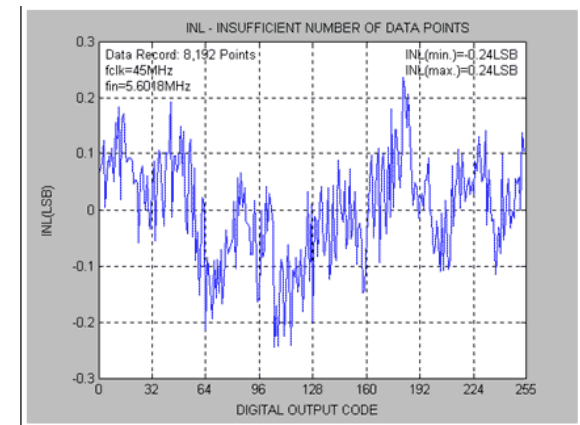
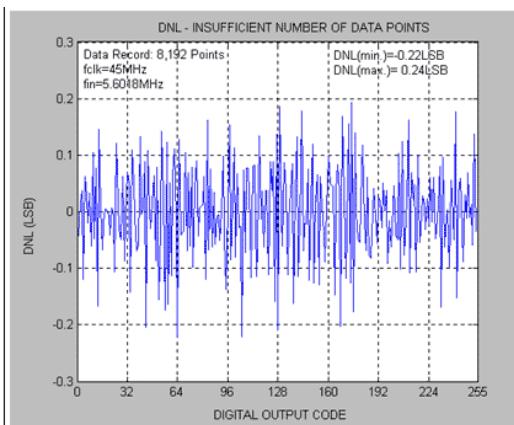
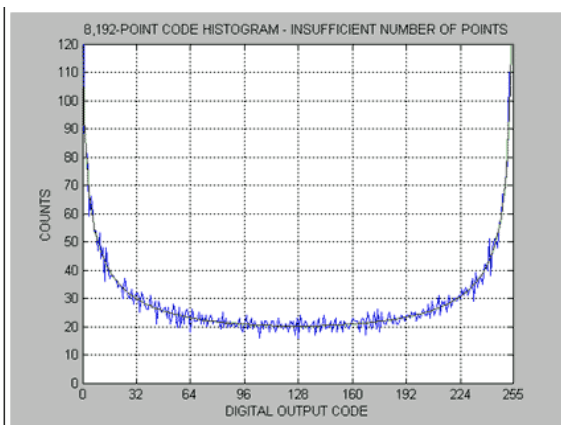
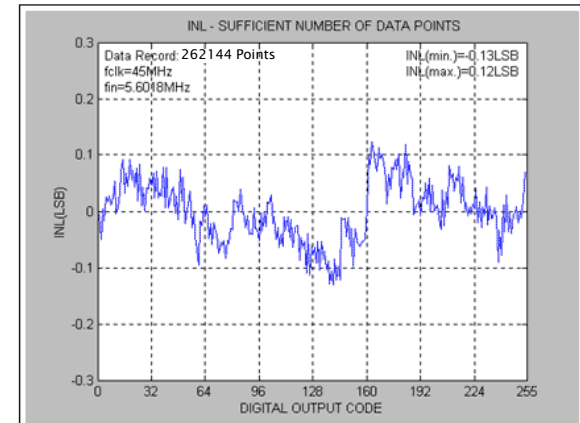
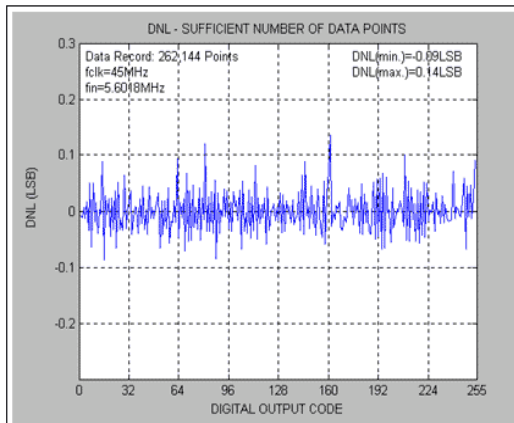
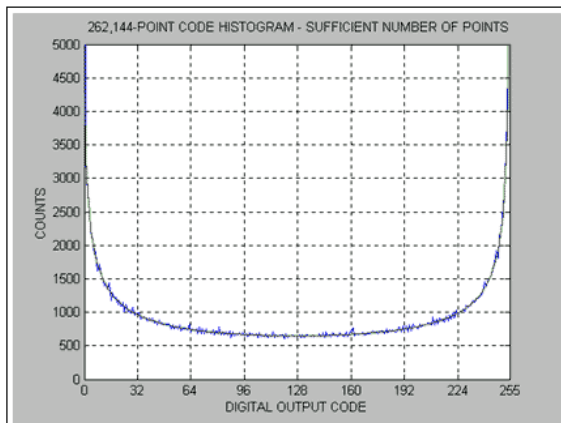
The deviation between measured and ideal histograms at each output code can be used to determine data converter's DNL as follows:

$$\text{DNL (LSB)} = [P_{\text{meas}}(n) / P_{\text{ideal}}(n)] - 1$$

$$p(n) = \frac{1}{\pi} \left[\sin^{-1} \left(\frac{V_{FS} (n - 2^{N-1})}{A \cdot 2^N} \right) - \sin^{-1} \left(\frac{V_{FS} (n - 1 - 2^{N-1})}{A \cdot 2^N} \right) \right]$$

Histogram testing limitations (I)

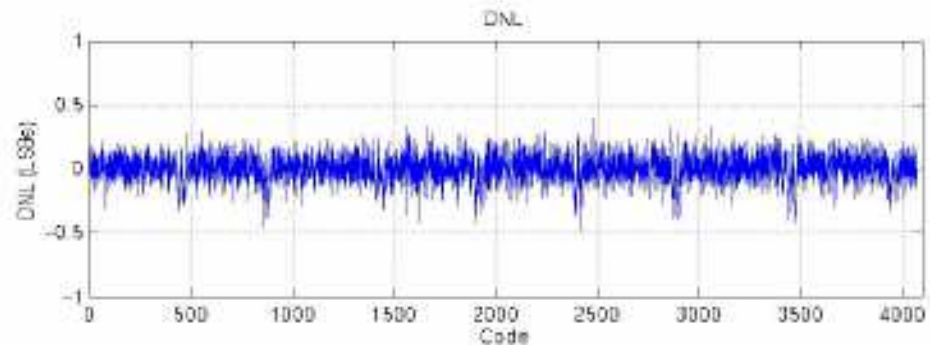
- The histogram method requires the capture of fairly large data records. The number of samples required depends on the resolution of the ADC, the desired confidence level of the measurement, and the size of the DNL error.



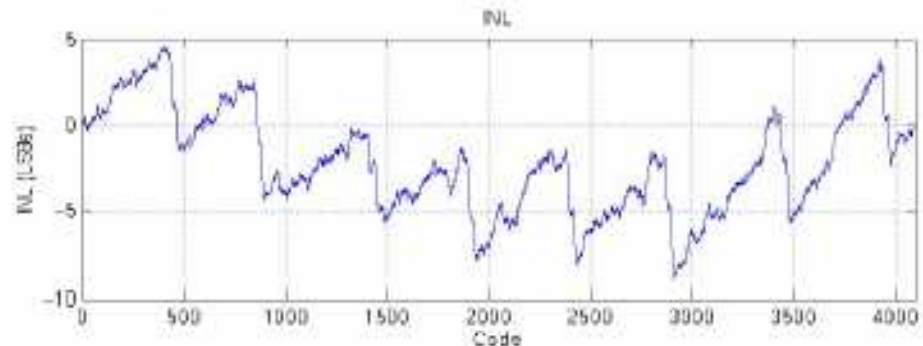
Histogram testing limitations (II)

- Histogram testing assumes monotonicity → code flips are not detected!
- Dynamic sparkle codes produce only minor DNL/INL errors → Look at ADC output to detect!
- Input referred noise is not detected → histogram test eliminates the effects of noise by averaging it over all the code bins!
- DNL/INL errors hidden by the noise →

DNL extracted from histogram
is 'smeared out' by noise



Measured ADC INL
(no histogram testing used)

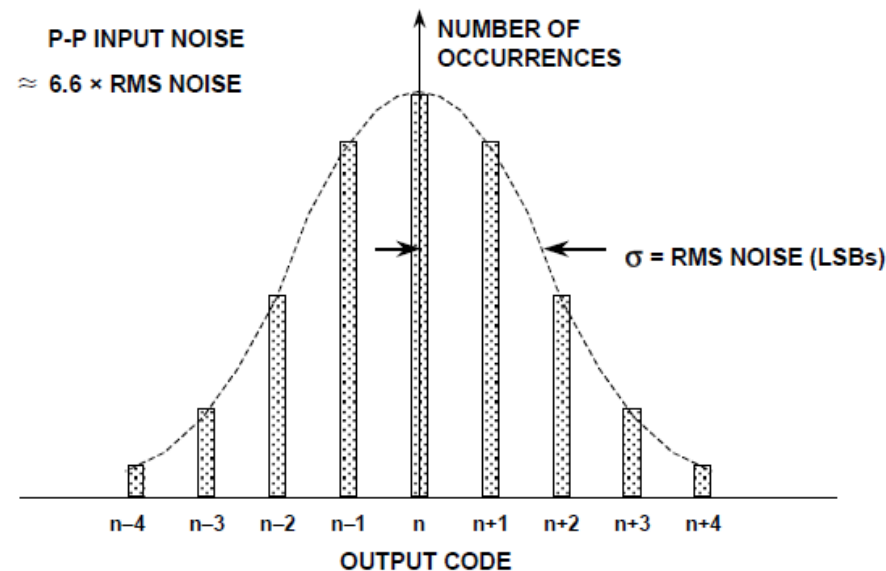


Histogram testing - dc input signal -

Applying a dc input (the actual value is not critical, but a voltage near mid-scale can be used for convenience), and performing an histogram allows to measure ADC input referred noise.

If the peak-to-peak input referred ADC noise is less than 1 LSB, the output samples should all correspond to a single code value. If the dc input happens to fall exactly on a code transition, the samples will be divided between two adjacent codes. However, the dc input can be offset by $\frac{1}{2}$ LSB in order to produce a single code at the output.

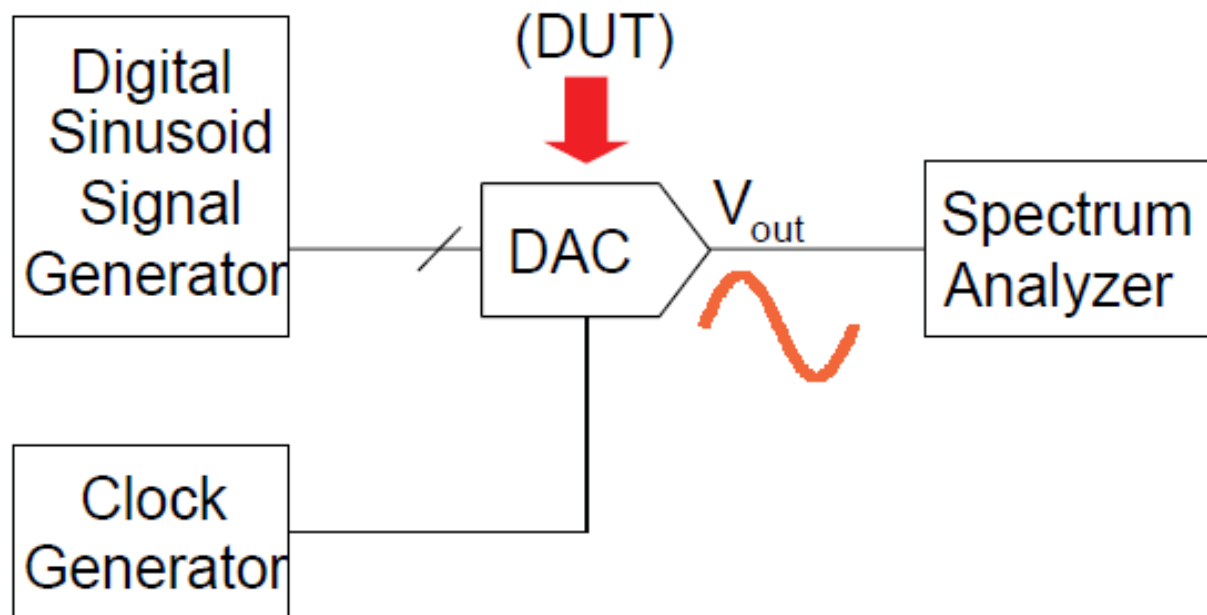
The effect of input-referred noise is to spread the output samples over a number of code bin. Assuming the noise is Gaussian, the input-referred noise is simply the standard deviation of the distribution.



Dynamic (spectral) testing

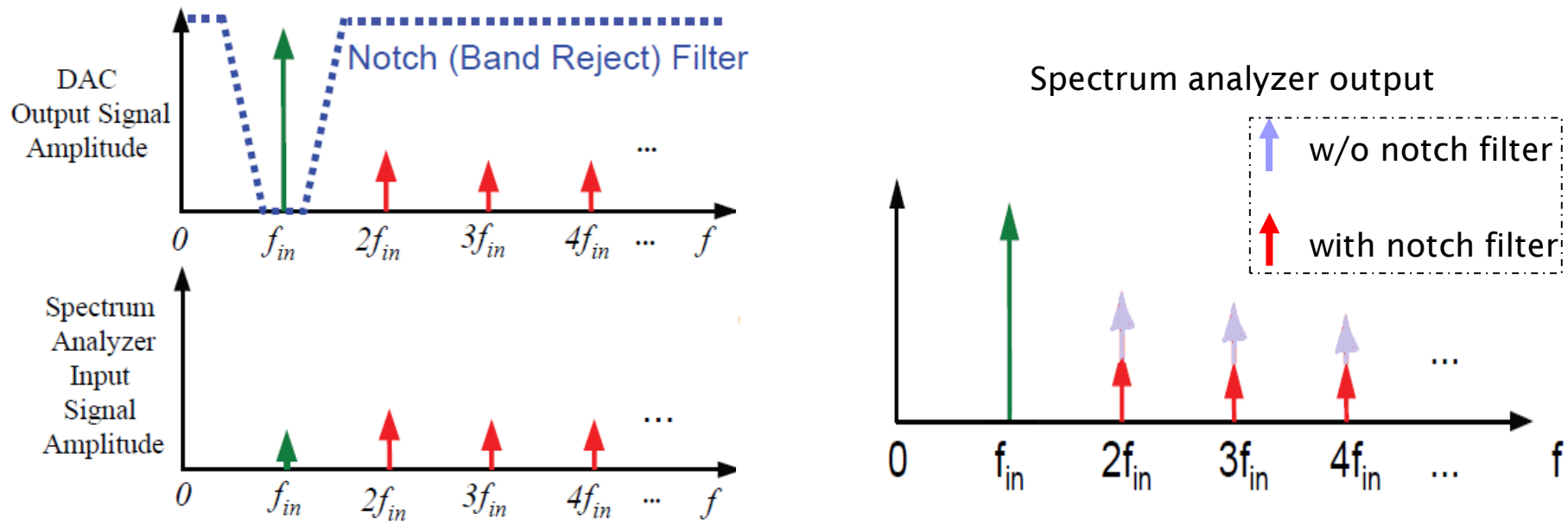
- DAC testing [26]
- ADC testing [26]
 - direct ADC spectral test via DAC
 - ADC spectral test via data acquisition system

DAC testing (I)



- Input sinusoid → need to have significantly better spectral purity compared to DAC linearity.
- Spectrum analyzer → need to have better linearity than DAC or use notch filter (see next slide)

DAC testing (II)



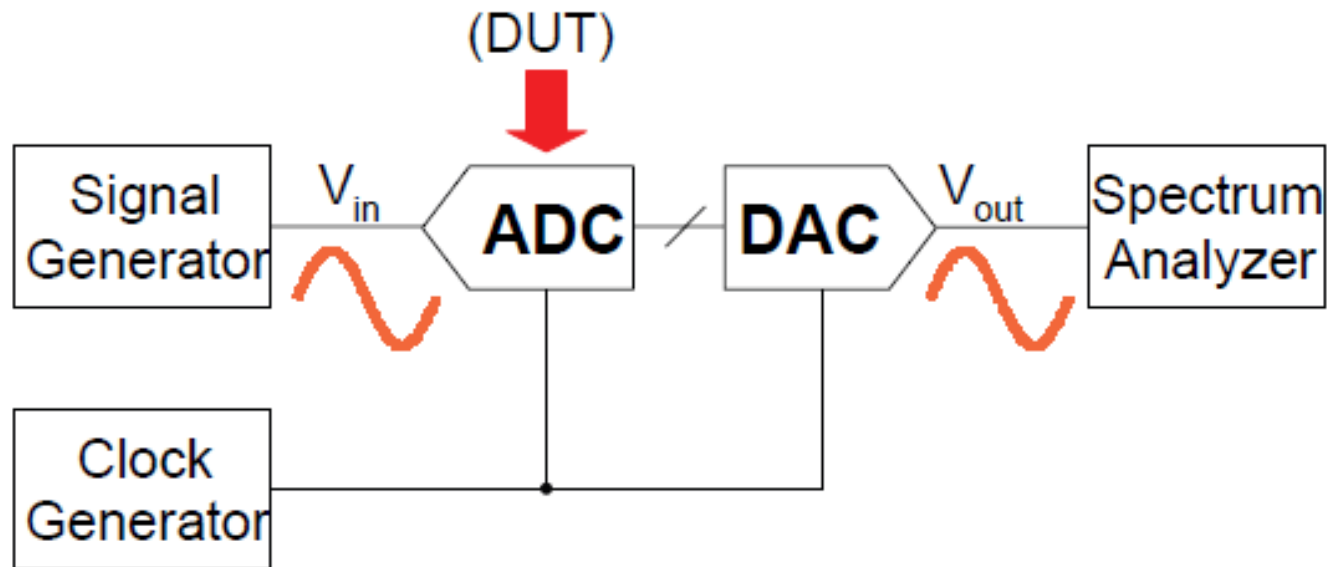
- Measure fundamental signal level.
- Notch out fundamental signal to avoid driving spectrum analyzer (*) into non-linear region.
- Measure the harmonic content of the DAC output.

(*) Audio analyzers have internal notch filters, whereas spectrum analyzers need external ones !

ADC testing

- Direct ADC spectral test via DAC
- ADC spectral test via data acquisition system

Direct ADC spectral test via DAC (I)



- Input sinusoid → need to have significantly better linearity compared to ADC one.
- Need a DAC with much better performance (at least 2 bits) compared to ADC under test.
- Beware of DAC output $\sin x/x$ frequency shaping.
- Spectrum analyzer → need to have better linearity than ADC.
- May need to introduce band-pass and notch filters to address input sinusoid and spectrum analyzer issues, respectively.

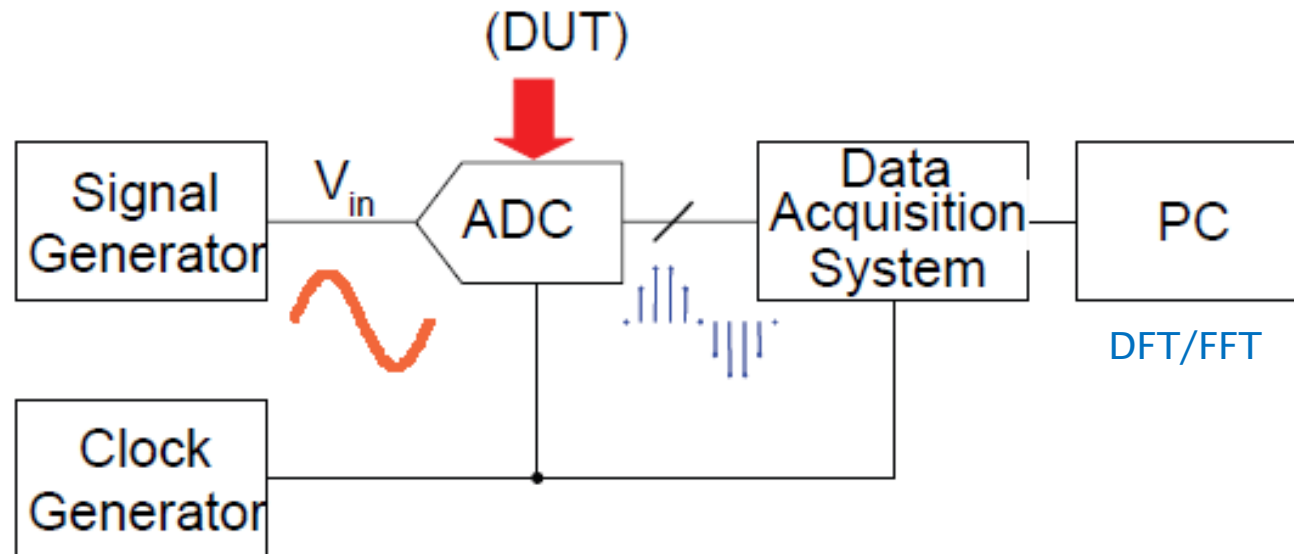
Direct ADC spectral test via DAC (II)

The back-to-back ADC test can serve as a quick check of overall ADC ac performance for resolutions of up to 10-12 bits.

It is also useful in testing end-to-end performance in systems which use a reconstruction DAC in conjunction with an ADC, as in audio applications.

To evaluate high performance ADCs with more than 12-bits of resolution requires the use of digital techniques. However, for this type of ADCs, the back-to-back test may still be useful for quick checks of functionality.

ADC spectral test via data acquisition system

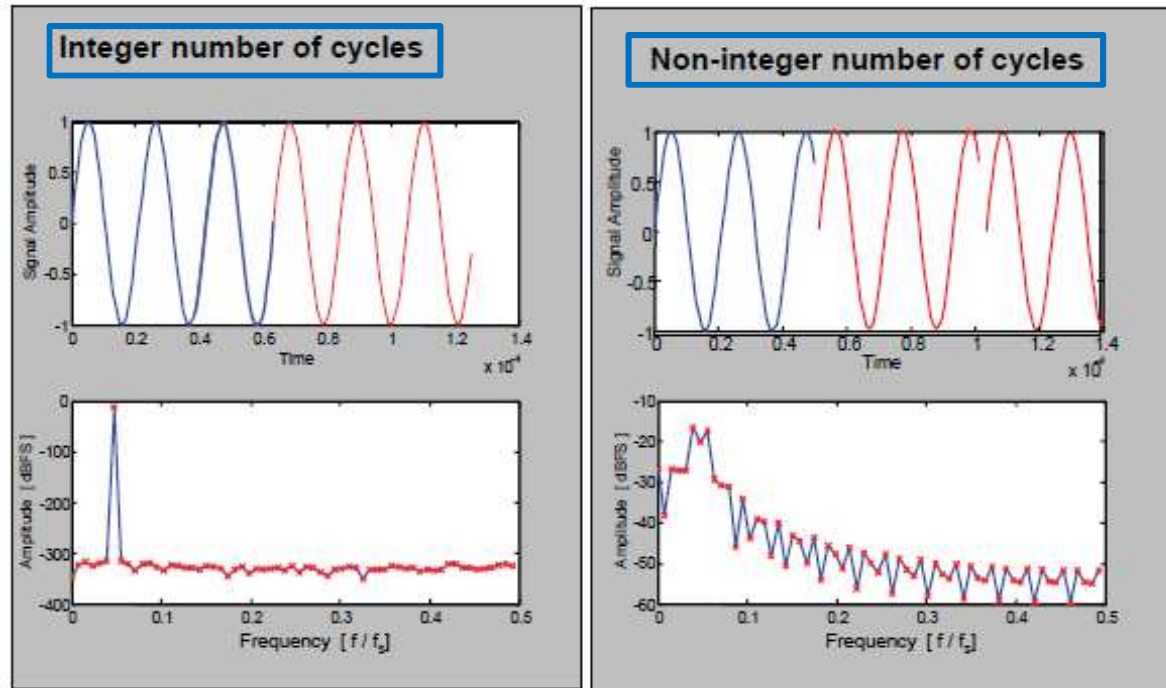


DFT of N samples spaced $T_S = 1/f_S$ seconds:

- N frequency bins from dc to f_S or $N/2$ frequency bins from dc to $f_S/2$
- each bin has width = f_S/N
- bin # n represents frequencies at nf_S/N

DFT with $N = 2^P$ (P is an integer) can be found using a computationally efficient algorithm named Fast Fourier Transform (FFT)

DFT periodicity



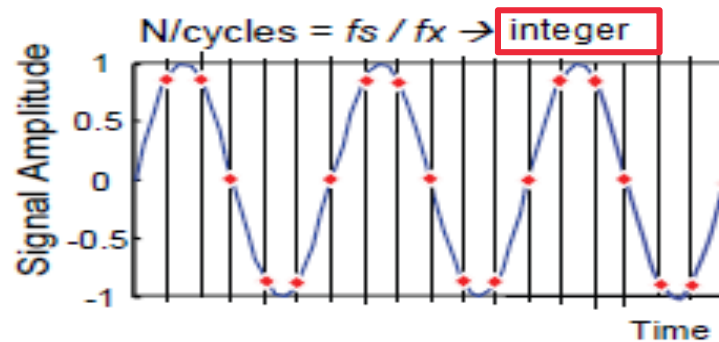
The DFT implicitly assumes that time sample blocks repeat every N samples

- With a non-integer number of signal periods within the observation window, the input yields significant amplitude/phase discontinuity at the block boundary
- This energy spreads into other frequency bins as “spectral leakage”
- Spectral leakage can be eliminated by either
 - Choice of integer number of sinusoid periods in the time sample block
 - Windowing

Integer number of cycles

No frequency spectrum leakage \rightarrow integer number M of cycles in the time window $\rightarrow T_X = \frac{T_{win}}{M}$

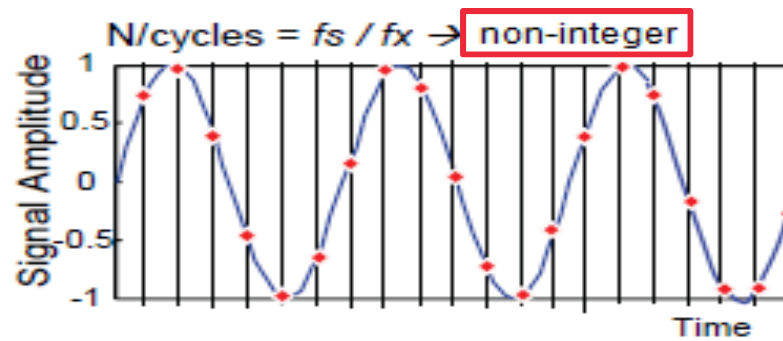
For random quantization noise \rightarrow non-integer $N/\text{cycles} = f_s / f_x \rightarrow T_S = \frac{T_{win}}{N} = \frac{M}{N} T_X$



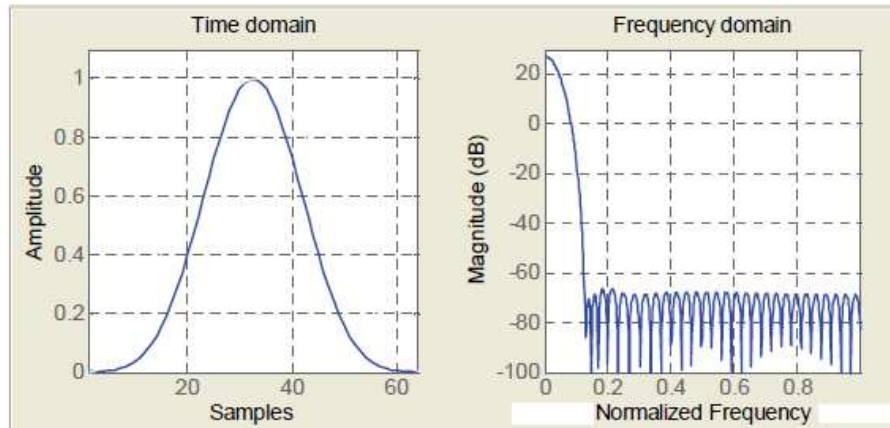
Non-integer number

\swarrow

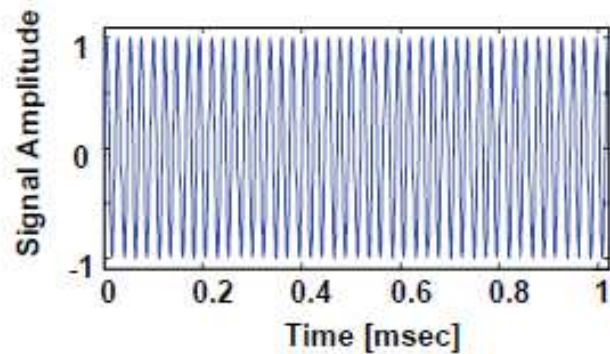
$$\frac{N}{M}$$



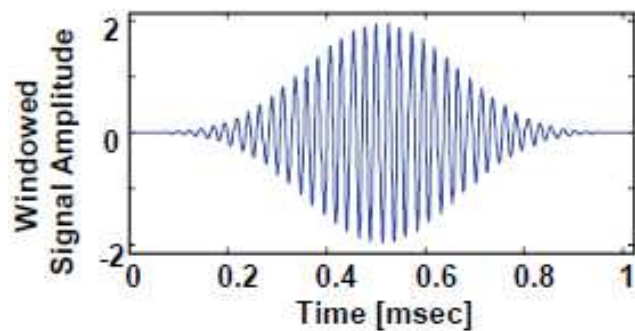
Windowing (I)



Window example (Nuttall, $N=64$)

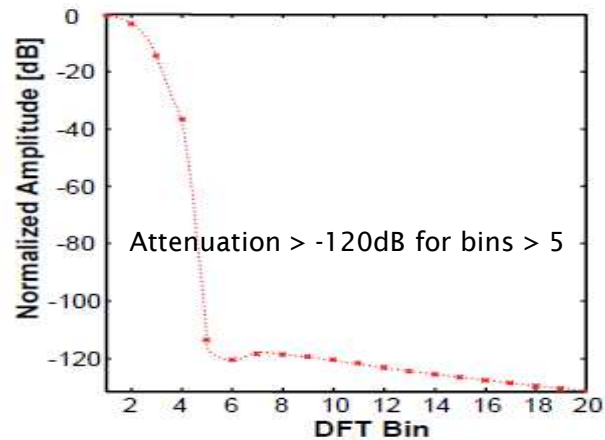


Signal before windowing.



Windowing smoothens down to zero at the beginning and the end of the observation window, removing discontinuity at time block boundaries.

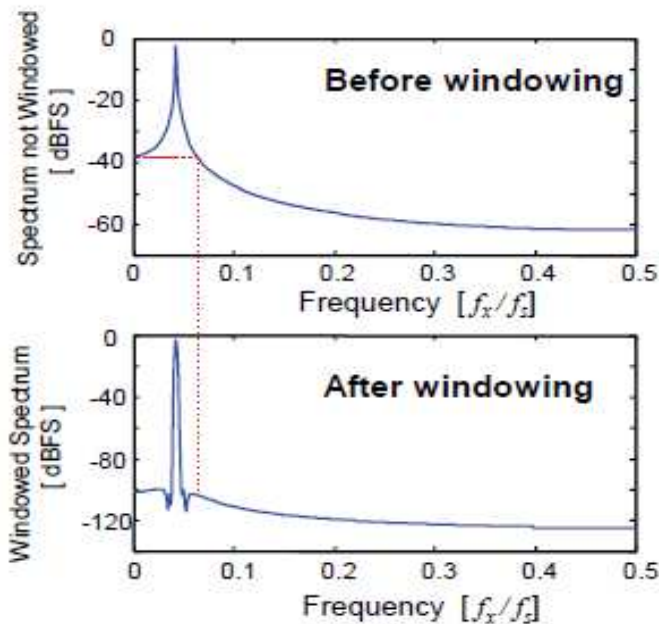
Windowing (II)



Time samples are multiplied by window coefficients



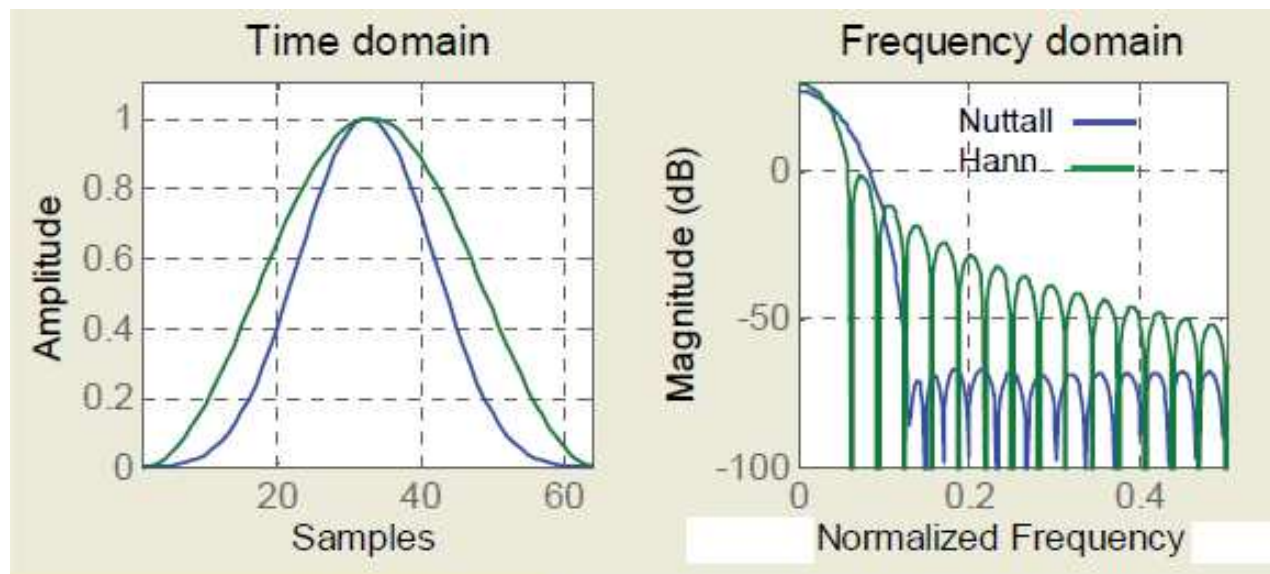
convolution in frequency domain



Windowing results in >100dB attenuation of sidelobes
Signal energy "smeared" over several bins

Windowing (III)

Windows type tradeoff → attenuation versus fundamental signal spreading to number of adjacent bins

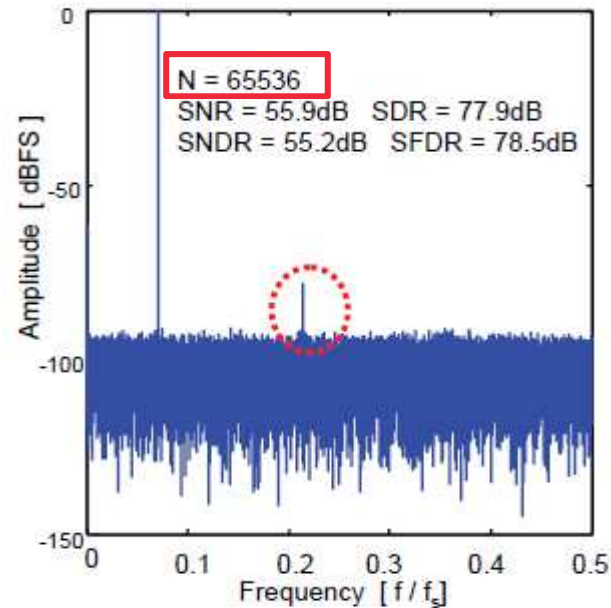
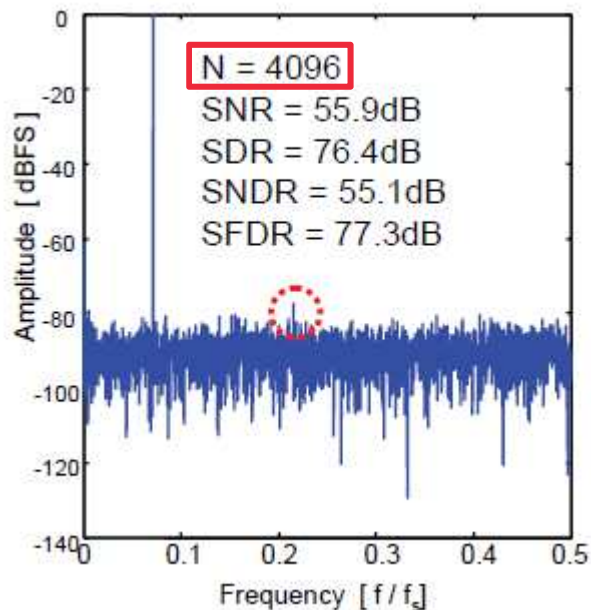


Integer number of cycles vs windowing

- Integer number of cycles
 - Signal energy for a single sinusoid falls into single DFT bin
 - Requires careful choice of f_x
 - Ideal for simulations
 - Measurements \rightarrow need to lock f_x to f_s ---- not always possible
- Windowing
 - No restrictions on $f_x \rightarrow$ no need to lock f_x to $f_s \rightarrow$ good for measurements
 - Signal energy and its harmonics distributed over several DFT bins
 - Normally requires more samples for a given accuracy

Some more hints about DFT

N should be large enough so that distortion products can be distinguished from the DFT noise floor.



Beware that averaging several DFTs does not further reduce the noise floor !

It simply reduce the variations between the individual noise spectral component amplitudes on the screen.

Some more hints about data converters

- A data converter does not just have one "input" or "output" pin but:
 - Clock
 - Power supply, ground
 - Reference voltage
- How to get the bits off chip?
 - "Full swing" CMOS signaling works well for $f_{\text{CLK}} < 100\text{MHz}$
 - Prefer Low Voltage Differential Signaling (LVDS) for faster data converters
 - LVDS vs CMOS:
 - Higher speed, more power efficient at high speed
 - Less crosstalk/EMI
 - Two pins/bit

References

- [1] Communication in the Presence of Noise - C.E. Shannon - Proceedings IRE , Jan. 1949.
- [2] Course on clock jitter – G. Nicollini – 2014.
- [3] Spectra of Quantized Signals – W.R. Bennett – *Bell System technical Journal*, July 1948.
- [4] Increasing the Spurious-Free Dynamic Range of an Integrated Spectrum Analyzer - Oude Alink - MSc. Thesis, Nov. 2008
- [5] A 10b, 500MSample/s CMOS DAC in 0.6mm² – C. H. Lin and K. Bult - *IEEE JSSC*, December 1998.
- [6] Formulation of INL and DNL Yield Estimation in Current-Steering D/A Converters – Y. Cong, and R. L. Geiger – *Proc. ISCAS*, May 2002.
- [7] A 10b 50MHz CMOS D/A Converter with 75 Ω Buffer – M. J. M. Pelgrom - *IEEE JSSC*, December 1990.
- [8] A 10b, 70MS/s CMOS D/A Converter – Y. Nakamura, et al. - *IEEE JSSC*, April 1991.
- [9] A 12-bit, Intrinsic Accuracy High-Speed CMOS DAC – J. Bastos, et al. - *IEEE JSSC*, December 1998
- [10] A 14bit Intrinsic Accuracy Q² *Random Walk* CMOS DAC – G. A. M. Van der Plas et al. - *IEEE JSSC*, December 1999.
- [11] A Self-Calibration Technique for Monolithic High-resolution D/A Converters – D. W. J. Groeneveld et al. - *IEEE JSSC*, December 1989.
- [12] A 1.5V 14bit 100MS/s Self-Calibrated DAC – Y. Cong and R.L Geiger - *IEEE JSSC*, December 2003.
- [13] A 16b 400MS/s DAC with <80dBc IMD to 300MHz and <160dBm/Hz Noise Power Spectral Density – W. Schofield et al. – *Proc. ISSCC*, Feb. 2003.
- [14] A Self-Calibrating 15 bit CMOS A/D Converter– H-S- Lee et al - *IEEE JSSC*, December 1984.
- [15] A Self-Calibrating SAR ADC for Automotive Microcontrollers– C. Burgio et al. – *Proc. Advances in Analog Circuit Design (AACD)*, 2016
- [16] An 820 μ W 9b 40MS/s Noise-Tolerant Dynamic SAR ADC in 90nm Digital CMOS – V. Giannini et al. – *Proc. ISSCC*, Feb. 2008.

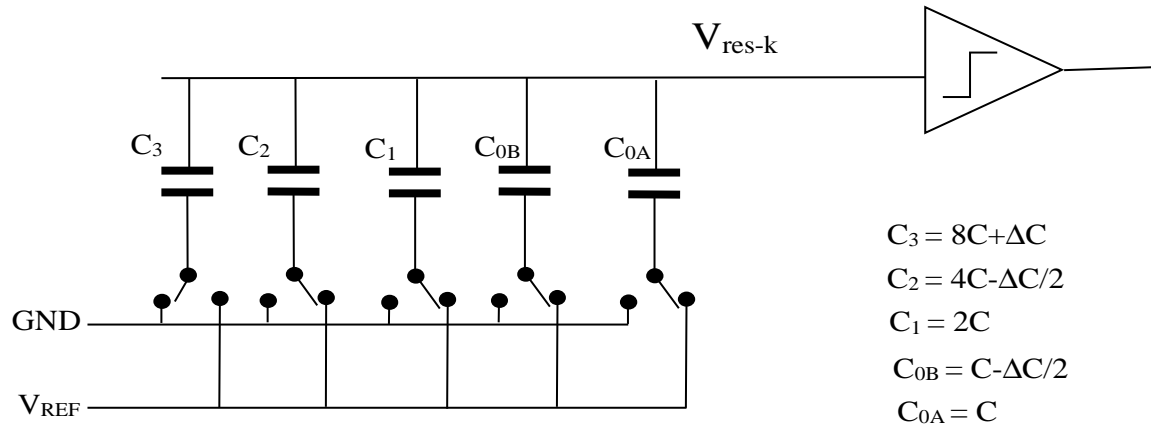
G. Nicollini

References (cont,d)

- [17] 1.05V On-Request 10b General-Purpose ADC in 65nm Digital CMOS Technology – M. Zamprogno et al. – *Proc. ICECS*, Dec. 2011.
- [18] A 12-bit non-calibrating noise-immune redundant SAR ADC for System-on-a-chip – A. Shrivastava – *Proc. ISCAS*, May 2006.
- [19] A 0.015mm² 63fJ/conversion-step 10-Bit 220MS/s SAR ADC with 1.5b/step Redundancy and Digital Metastability Correction – R. Vitek et al. – *Proc. CICC*, Sept. 2012.
- [20] A 1.2V 10b 20MS/s Non-Binary Successive Approximation ADC in 0.13μm CMOS – F. Kuttner – *Proc. ISSCC*, Feb. 2002.
- [21] SAR ADC Algorithm with Redundancy – T. Ogawa et al. – *Proc. ISCAS*, May 2008.
- [22] A Study of Error Sources in Current Steering Digital-to-Analog Converters – D. Mercer ,– *Proc. CICC*, 2004.
- [23] SFDR-Bandwidth Limitations for High speed High Resolution Current Steering CMOS D/A Converters – A. Van den Bosch, et al.,– *Proc. ICECS*, 1999.
- [24] Fully bipolar, 120-Msample/s 10-b track-and-hold circuit, - P. Vorenkamp and J. Verdaasdonk, *IEEE JSSC*, July 1992.
- [25] A 144, 100-Ms/s CMOS DAC designed for spectral performance, - A.R. Bugeja, et al. , *IEEE JSSC*, Dec. 1999
- [26] Data converters, - H. Khorramabadi, *EECS 247 lecture 13*, UC Berkeley, 2006.
- [27] Advanced Engineering Course on High-Performance Data converters, - K. Bult, *Mead education*, Lausanne, 2018.

Appendix 1: calibration with single-correction DAC

Example with N=4:



$$V_{res-3} = \frac{+(8+\Delta)V_R - (8-\Delta)V_R}{16} = 2\Delta \cdot \frac{V_R}{16}$$

→

$$V_{err-3} = 0.5 \cdot (2\Delta) = \Delta \quad \text{OK}$$

$$V_{res-2} = \frac{+(4+\frac{\Delta}{2})V_R - (4-\frac{\Delta}{2})V_R}{16} = 0 \cdot \frac{V_R}{16}$$

→

$$V_{err-2} = 0.5 \cdot (0 - \Delta) = -\frac{\Delta}{2} \quad \text{OK}$$

$$V_{res-1} = \frac{+2V_R - (2-\frac{\Delta}{2})V_R}{16} = \frac{\Delta}{2} \cdot \frac{V_R}{16}$$

→

$$V_{err-1} = 0.5 \cdot (\frac{\Delta}{2} - \Delta + \frac{\Delta}{2}) = 0 \quad \text{OK}$$

$$V_{res-0B} = \frac{+(1-\frac{\Delta}{2})V_R - V_R}{16} = -\frac{\Delta}{2} \cdot \frac{V_R}{16}$$

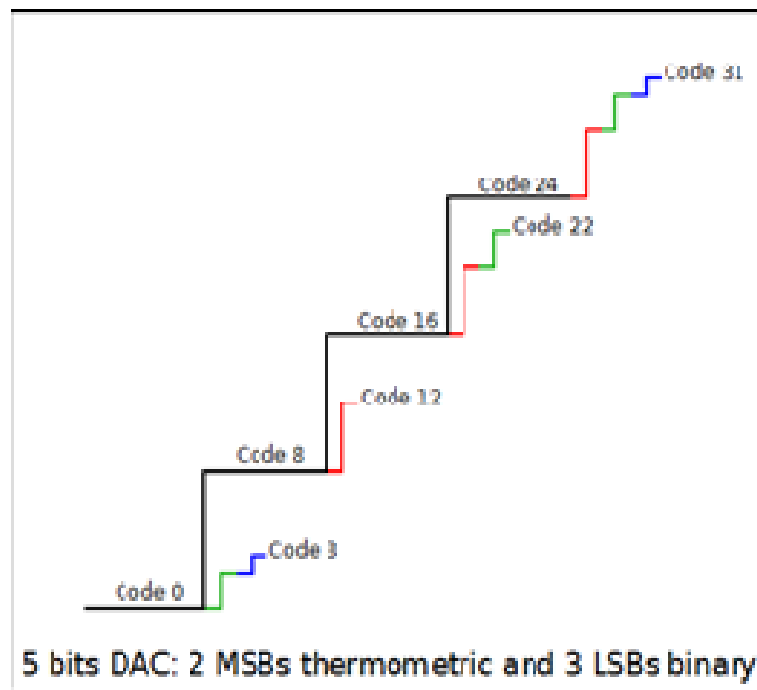
→

$$V_{err-1} = 0.5 \cdot (-\frac{\Delta}{2} - \Delta + \frac{\Delta}{2} - 0) = -\frac{\Delta}{2} \quad \text{OK}$$

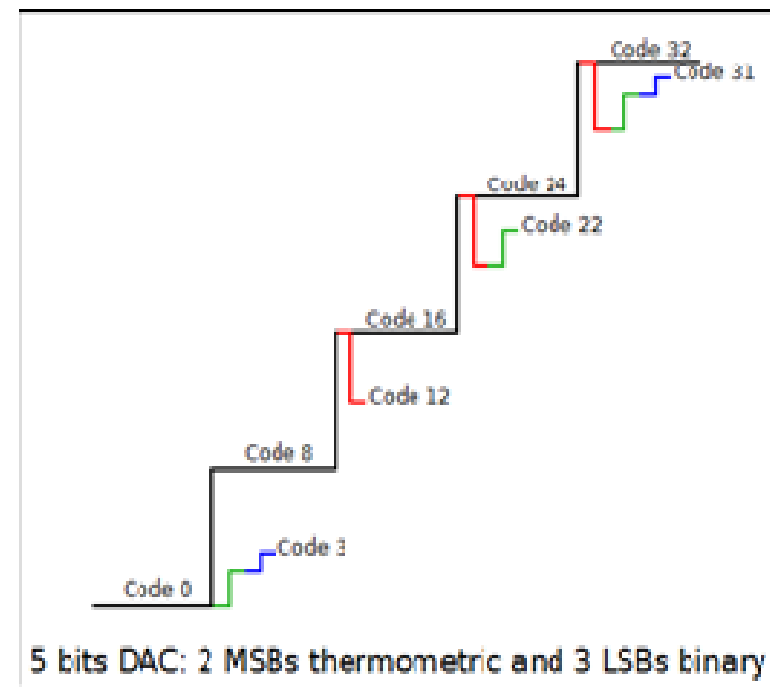
Appendix 2: Thermometric DAC with 2^P instead of $2^P - 1$ caps

Example with $N=5 \rightarrow$ 2MSB thermometric and 3LSB binary-weighted

Conventional approach



new approach



Appendix 3: redundant conversion step implementation

