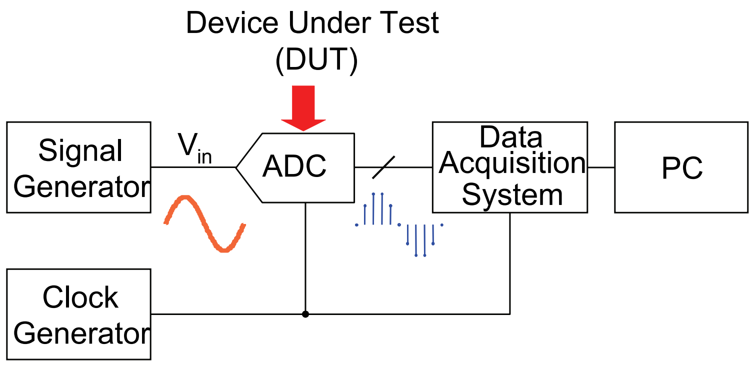
EE247
Lecture 13

- Administrative issues
  - Midterm exam date changed to Tues. Oct. 28th
    - You can *only* bring one 8x11 paper with your own written notes (please do not photocopy)
    - No books, class notes or any other kind of handouts/notes, calculators, computers, PDA, cell phones....
    - Midterm includes material covered to end of lecture 14

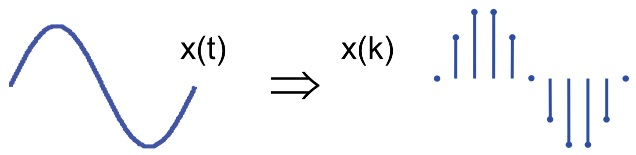
EE247
Lecture 13

- Data Converters
  - Data converter testing (continued)
    - Dynamic tests
      - Spectral testing → Reveals ADC errors associated with dynamic behavior i.e. ADC performance as a function of frequency
        - Direct Discrete Fourier Transform (DFT) based measurements utilizing sinusoidal signals
        - DFT measurements including windowing
    - Relationship between: DNL & SNR, INL & SFDR
    - Effective number of bits (ENOB)
  - D/A converter design
    - Architectures

ADC Spectral Test via Data Acquisition Sytem



Analyzing ADC Outputs via Discrete Fourier Transform (DFT)



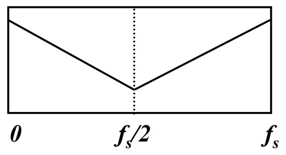
- Sinusoidal waveform has all its power at one single frequency
- An ideal, infinite resolution ADC would preserve ideal, single tone spectrum
- DFT used as a vehicle to reveal ADC deviations from ideality

Discrete Fourier Transform (DFT) Properties

- DFT of N samples spaced $T_s=1/f_s$ seconds:
  - N frequency bins from DC to f_s
  - Bin m represents frequencies at $m * f_s/N$ $[Hz]$
- DFT frequency resolution:
  - Proportional to f_s/N in [Hz/bin]
- DFT with $N = 2^k$ (k is an integer) can be found using a computationally more efficient algorithm named:
  - FFT → Fast Fourier Transform

DFT Magnitude Plots

- Because magnitudes of DFT bins (A_m) are symmetric around $f_S/2$, it is redundant to plot $|A_m|$'s for $m > N/2$



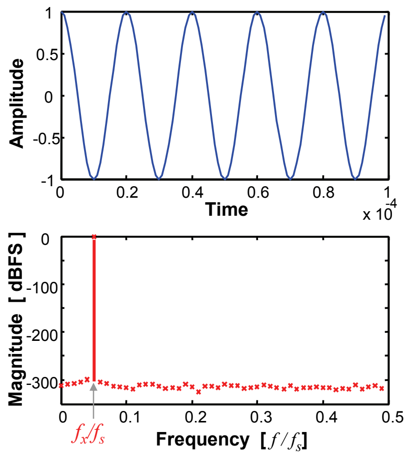
- Usually magnitudes are plotted on a log scale normalized so that a full scale sinusoidal waveform with rms value a_{FS} yields a peak bin of $0dBFS$:

$$|A_m| \ [dBFS] = 20 \log_{10} \frac{|A_m|}{a_{FS} \cdot N/2}$$

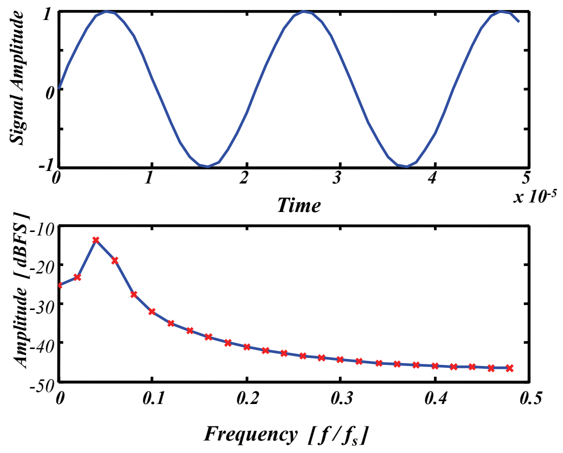
Matlab Example Normalized DFT

```
fs  = 1e6;
fx  = 50e3;
Afs = 1;
N   = 100;

% time vector
t = linspace(0, (N-1)/fs, N);
% input signal
y = Afs * cos(2*pi*fx*t);
% spectrum
s = 20 * log10(abs(dft(y)/N/Afs*2));
% drop redundant half
s = s(1:N/2);
% frequency vector (normalized to fs)
f = (0:length(s)-1) / N;
```



“Another” Example …

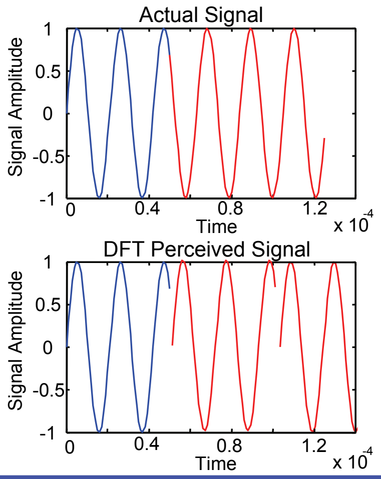


Even though the input signal is a pure sinusoidal waveform note that the DFT results does not look like the spectrum of a sinusoid …

Seems that the signal is distributed among several bins

DFT Periodicity

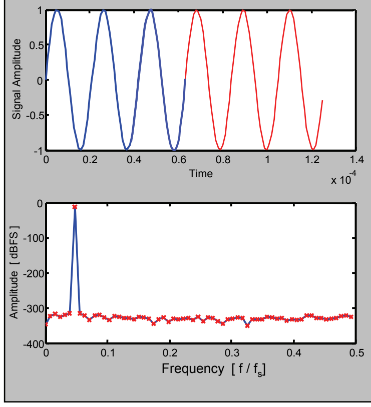
- The DFT implicitly assumes that time sample blocks repeat every N samples
- With a non-integer number of signal periods within the observation window, the input yields significant amplitude/phase discontinuity at the block boundary
- This energy spreads into other frequency bins as "spectral leakage"
- Spectral leakage can be eliminated by either
  1. Choice of integer number of sinusoids in each block
  2. Windowing



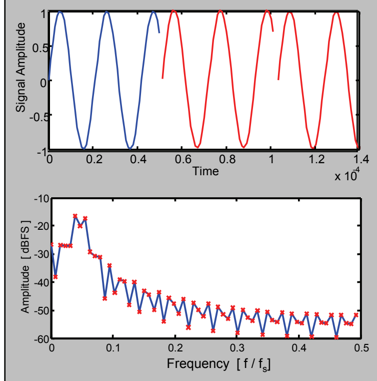
Frequency Spectrum

Integer # of Cycles versus Non-Integer # of Cycles

Integer number of cycles

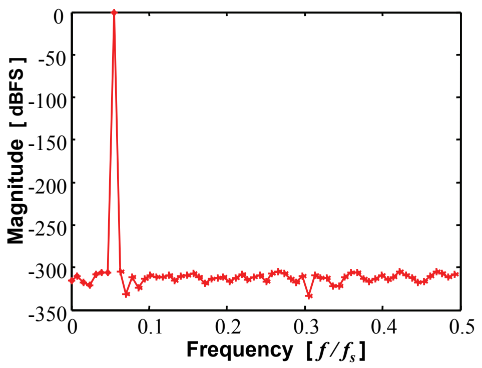


Non-integer number of cycles



Matlab Example
Integer Number of Cycles

```
fs  = 1e6;
Afs = 1;
N   = 2^7;
cycles=7;
fx=fs*cycles/N;
.
.
.
.
.
.
y = Afs * cos(2*pi*fx*t);
s = 20 * log10(abs(fft(y)/N/Afs*2));
```

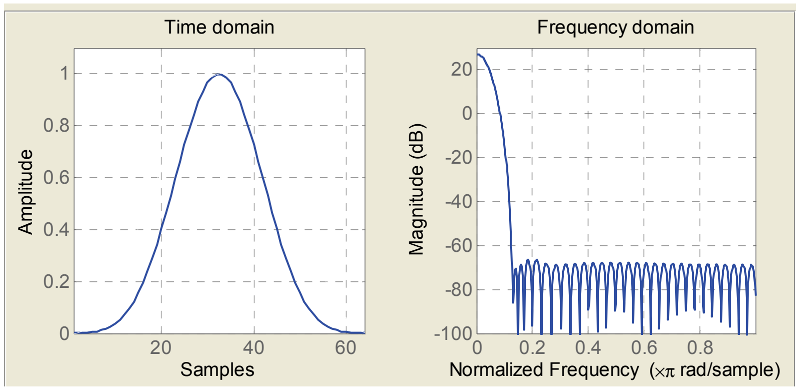


Notice: Range of test signals limited to $[(cycles) x f_s/N]$

Windowing

- Spectral leakage can be attenuated by “windowing” time samples prior to the DFT
  - Windows taper smoothly down to zero at the beginning and the end of the observation window
  - Time samples are multiplied by window coefficients on a sample-by-sample basis
    → Convolution in frequency domain
- Large number choices of various windows
  - Tradeoff: attenuation versus fundamental signal spreading to number of adjacent bins
- Window examples: Nuttall versus Hann

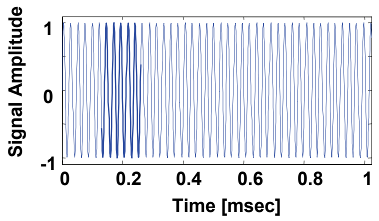
Example: Nuttall Window



- Time samples are multiplied by window coefficients on a sample-by-sample basis
- Multiplication in the time domain → convolution in the frequency domain

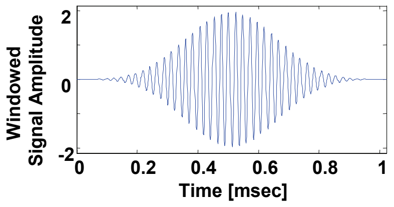
Windowed Data

- Signal before windowing



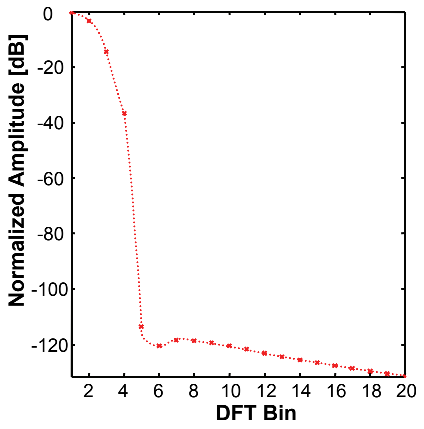
- Time samples are multiplied by window coefficients on a sample-by-sample basis

- Signal after windowing
  - Windowing removes the discontinuity at block boundaries



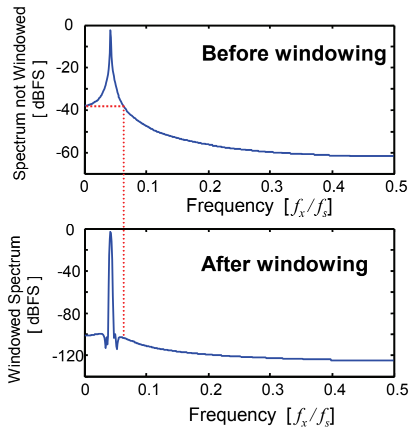
Nuttall Window DFT

- Only first 20 bins shown
- Response attenuated by -120dB for bins > 5
- Lots of windows to choose from (go by name of inventor- Blackman, Harris, Nutall…)
- Various window trade-off attenuation versus width (smearing of sinusoids)

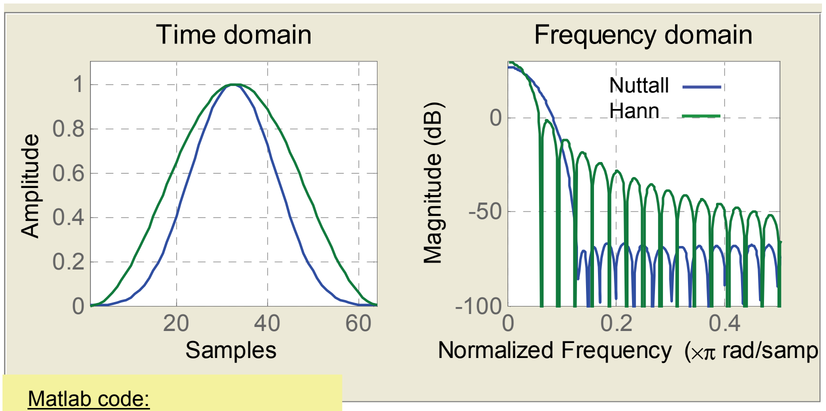


DFT of Windowed Signal Spectrum Before/After Windowing

- Windowing results in ~ 100dB attenuation of sidelobes
- Signal energy “smeared” over several (approximately 10) bins



Window
Nuttall versus Hann



Matlab code:

```
N=64;
wvtool(nuttallwin(N),hann(N));
```

Integer Cycles versus 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 → need to lock f_x to f_s (PLL)- not always possible
- Windowing
  - No restrictions on f_x → no need to have the signal locked to f_s
    → Good for measurements w/o having the capability to lock f_x to f_s
  - Signal energy and its harmonics distributed over several DFT bins – handle smeared-out harmonics with care!
  - Requires more samples for a given accuracy

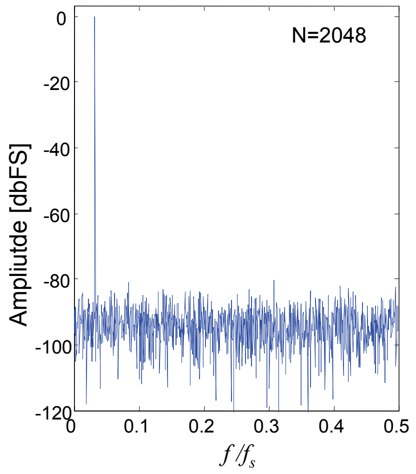
Example: ADC Spectral Testing

- ADC with B bits
- Full scale input =2

```
B = 10;
delta = 2/2^B;
y = cos(2*pi*fx/fs*[0:N-1]);
y=round(y/delta)*delta;
s = abs(fft(y)/N*2);
f = (0:length(s)-1) / N;
```

ADC Output Spectrum

- Input signal bin:
  - Bx @ bin # ($N * f_x/f_s + 1$) (Matlab arrays start at 1)
  - A_{signal} = 0dBFS
- SNR?

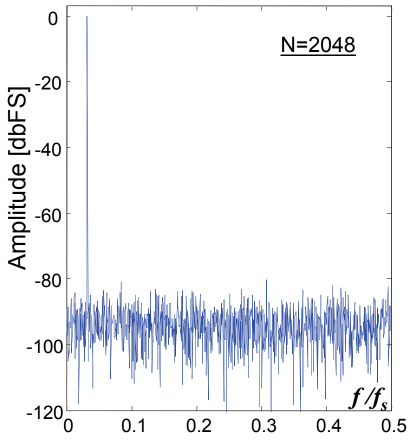


Simulated ADC Output Spectrum

- Noise bins: all except signal bin

```
bx = N*fx/fs + 1;
As = 20*log10(s(bx))
%set signal bin to 0
s(bx) = 0;
An = 10*log10(sum(s.^2))
SNR = As - An
```

- Matlab→SNR = 62dB (10 bits)
- Computed SQNR = 6.02xN+1.76dB=61.96dB



Note: In a real circuit including thermal/flicker noise → the measured total noise is the sum of quantization & noise associated with the circuit

Why is Noise Floor Not @ -62dB ?

- DFT bins act like an analog spectrum analyzer with bandwidth per bin of f_s/N
- Assuming noise is uniformly distributed, noise per bin:

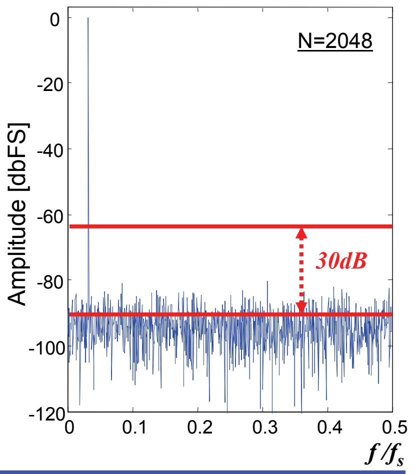
  (Total noise)/N/2

  →The DFT noise floor wrt total noise:

  $-10\log_{10}(N/2)$ [dB]

  below the actual noise floor
- For N=2048:

  $-10\log_{10}(N/2) = -30$ [dB]



DFT Plot Annotation

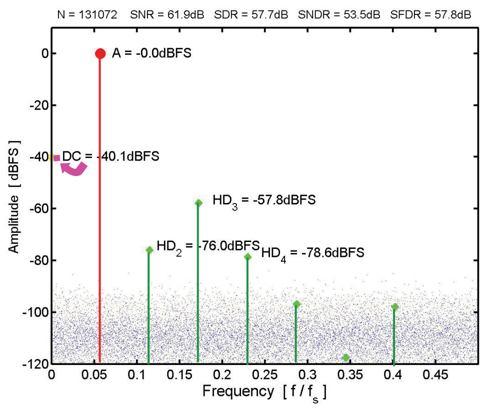
- Need to annotate DFT plot such that actual noise floor can be readily computed by one of these 3 ways:
  1. Specify how many DFT points (N) are used
  2. Shift DFT noise floor by $10\log_{10}(N/2)$ [dB]
  3. Normalize to "noise power in 1Hz bandwidth“ then noise is in the form of power spectral density

Spectral Performance Metrics
ADC Including Non-Idealities

- Signal S
- DC
- Distortion D
- Noise N

- Ideal ADC adds:
  – Quantization noise

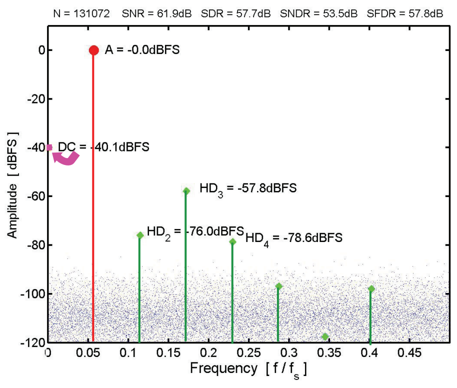
- Real ADC typically adds:
  – Thermal and flicker noise
  – Harmonic distortion associated with circuit nonlinearities



ADC Spectral Performance Metrics SNR

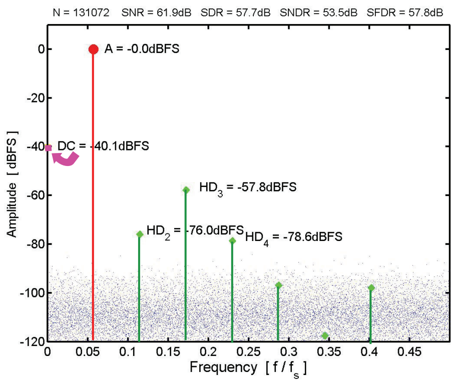
- Signal S
- DC
- Distortion D
- Noise N

- Signal-to-noise ratio
  SNR = 10log[(Signal Power) / (Noise Power)]
- In Matlab: Noise power includes power associated with all bins except:
  - DC
  - Signal
  - Signal harmonics



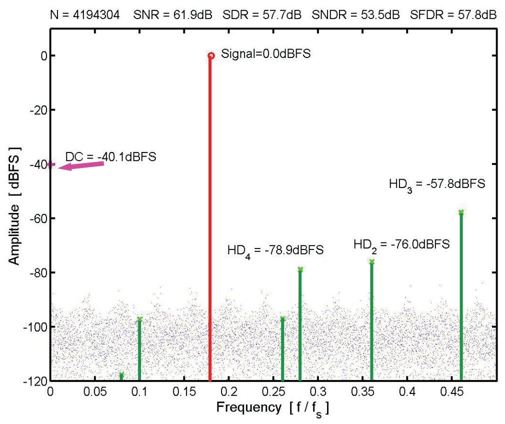
ADC Spectral Performance Metrics SDR & SNDR & SFDR

- SDR→ Signal-to-distortion ratio
  = 10log[(Signal Power) / (Total Distortion Power)]
- SNDR→ Signal-to-(noise+distortion)
  = 10log[S / (N+D)]
- SFDR→ Spurious-free dynamic range
  = 10log[(Signal)/ (Largest Harmonic)]
  → Typically SFDR > SDR



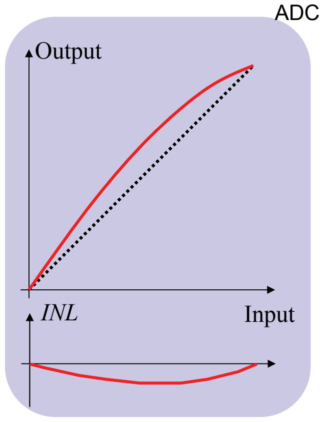
Harmonic Components

- At multiples of f_x
- Aliasing:
  - $f_{signal} = f_x = 0.18\ f_s$
  - $f_2 = 2\ f_0 = 0.36\ f_s$
  - $f_3 = 3\ f_0 = 0.54\ f_s$ $\rightarrow 0.46\ f_s$
  - $f_4 = 4\ f_0 = 0.72\ f_s$ $\rightarrow 0.28\ f_s$
  - $f_5 = 5\ f_0 = 0.90\ f_s$ $\rightarrow 0.10\ f_s$
  - $f_6 = 6\ f_0 = 1.08\ f_s$ $\rightarrow 0.08\ f_s$

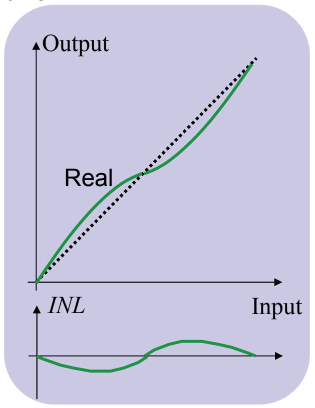


Relationship INL & SFDR/SNDR

ADC Transfer Curve

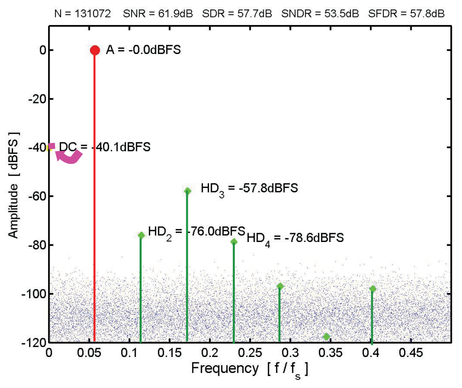


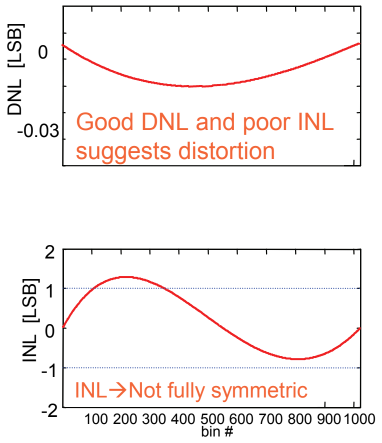
Quadratic shaped transfer function:
$\rightarrow$ Gives rise to **<u>even</u>** order harmonics



Cubic shaped transfer function:
$\rightarrow$ Gives rise to **<u>odd</u>** order harmonics

Frequency Spectrum versus INL & DNL

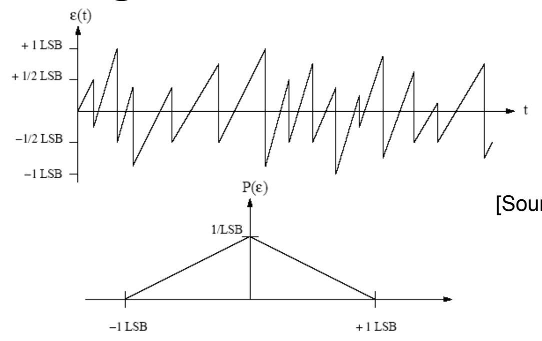




Relationship INL & SFDR/SNDR

- Nature of harmonics depend on "shape" of INL curve
- Rule of Thumb: SFDR $\cong$ 20log(2^B/INL)
  - E.g. 1LSB INL, 10b→ SFDR≅60dB
- Beware, this is of course only true under the same conditions at which the INL was taken, i.e. typically low input signal frequency

SNR Degradation due to DNL



[Source: Ion Opris]

- Uniform quantization error pdf was assumed for ideal quantizer over the range of: +/- Δ/2
- Let's now add uniform DNL over +/- Δ/2 and repeat math...
  - Joint pdf for two uniform pdfs → Triangular shape

SNR Degradation due to DNL

- To find total noise → Integrate triangular pdf:

$$\overline{e^2} = 2\int_0^{+\Delta}(1-e)\frac{e^2}{\Delta}de = \frac{\Delta^2}{6} \qquad \Rightarrow SNR = 6.02\cdot N - 1.25 \text{ [dB]}$$

3dB

- Compare to ideal quantizer:

$$\overline{e^2} = \int_{-\Delta/2}^{+\Delta/2}\frac{e^2}{\Delta}de = \frac{\Delta^2}{12} \qquad \Rightarrow SNR = 6.02\cdot N + 1.76 \text{ [dB]}$$

→Error associated with DNL reduces overall SNR

SNR Degradation due to DNL

- More general case:
  - Uniform quantization error ±0.5Δ
  - Uniform DNL error ± DNL [LSB]
  - Convolution yields trapezoid shaped joint pdf
  - SQNR becomes:

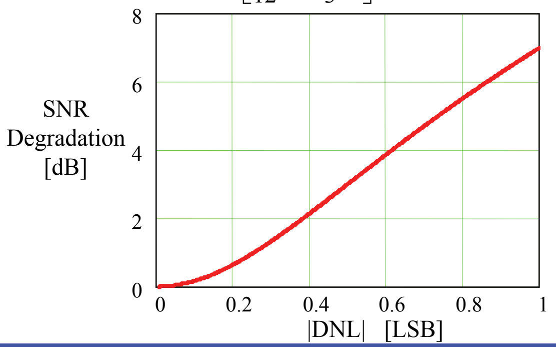
$$\boxed{SQNR = \frac{\frac{1}{2}\left(\frac{2^N\Delta}{2}\right)^2}{\frac{\Delta^2}{12}+\frac{DNL^2}{3}}}$$

SNR Degradation due to DNL

- Degradation in dB:

$$SQNR\_\deg = 1.76 - 10\log\left[\frac{\frac{1}{8}}{\frac{1}{12}+\frac{DNL^2}{3}}\right]$$

Valid only for cases where with no missing codes



Summary
INL & SFDR - DNL & SNR

INL & SFDR

- Depends on "shape" of INL
- Rule of Thumb:

$$\boxed{SFDR \cong 20\log(2^B/INL)}$$

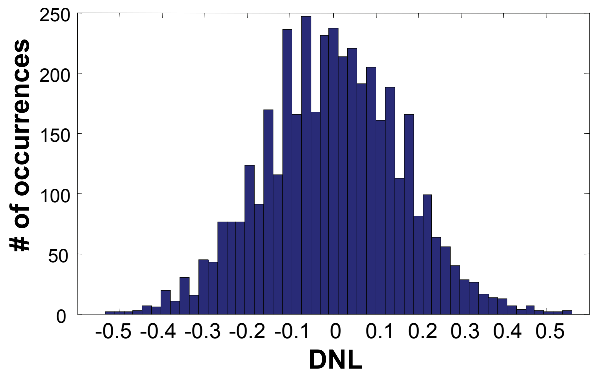
– E.g. 1LSB INL, 10b
→ SFDR≅60dB

DNL & SNR

Assumptions:
- DNL pdf →uniform
- No missing codes

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

Uniform DNL?



- DNL distribution of 12-bit ADC test chip
- Not quite uniform...

Effective Number of Bits (ENOB)

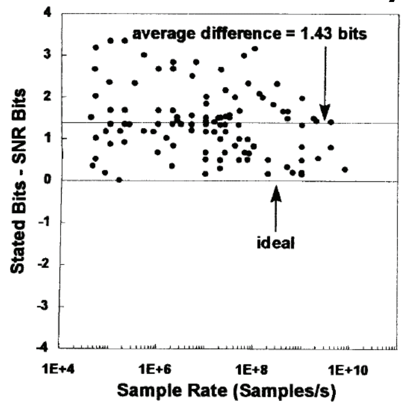
- Is a 12-Bit converter with 68dB SNDR really a 12-Bit converter?
- Effective Number of Bits (*ENOB*)

$$ENOB = \frac{SNDR - 1.76\text{dB}}{6.02\text{dB}}$$
$$= \frac{68 - 1.76}{6.02} = 11.0\text{Bits}$$

ENOB

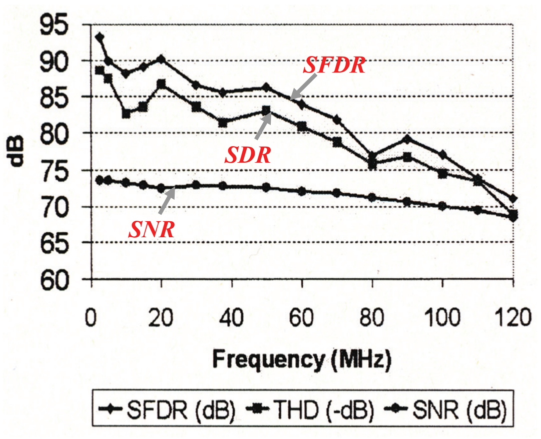
- At best, we get "ideal" ENOB only for negligible thermal noise, DNL, INL
- Low noise design is costly $\rightarrow$ 4x penalty in power per (ENOB-) bit or 6dB extra SNDR
- Rule of thumb for good performance /power tradeoff: ENOB < N-1

ENOB Survey



R. H. Walden, "Analog-to-digital converter survey and analysis," *IEEE J. on Selected Areas in Communications*, pp. 539-50, April 1999

Example: ADC Spectral Tests

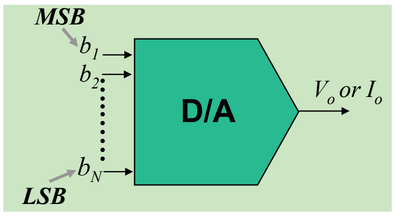


Ref: W. Yang et al., "A 3-V 340-mW 14-b 75-Msample/s CMOS ADC with 85-dB SFDR at Nyquist input," *IEEE J. of Solid-State Circuits,* Dec. 2001

D-to-A Converter Design

D/A Converter Transfer Characteristics

- An ideal digital-to-analog converter:
  - Accepts digital inputs b_1-b_n
  - Produces either an analog output voltage or current
  - Assumption (will be revisited)
    - Uniform, binary digital encoding
    - Unipolar output ranging from 0 to V_{FS}



Nomenclature:

$N = \#\ of\ bits$

$V_{FS} = full\ scale\ output$

$\Delta = min.\ step\ size \rightarrow 1LSB$

$$\Delta = \frac{V_{FS}}{2^N}$$

$$or\ N = log_2 \frac{V_{FS}}{\Delta} \rightarrow resolution$$

D/A Converters

- D/A architecture examples
  - Resistor string DAC
  - Charge Redistribution DAC
  - Current source type
- Static performance
  - Limited by component matching
  - Architectures
    - Unit element
    - Binary weighted
    - Segmented
  - Dynamic element matching
- Dynamic performance
  - Limited by timing errors causing glitches

D/A Converters

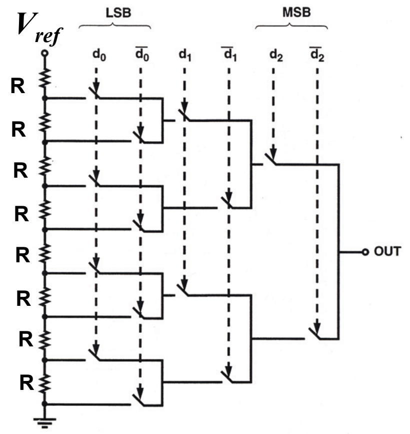
- Comprises *voltage* or *charge* or *current* based elements
- Examples for above three categories:
  - Resistor string → *voltage*
  - Charge redistribution → *charge*
  - Current source type → *current*

Resistor String DAC

Voltage based:

- A *B-bit* DAC requires: 2^B resistors in series
- All resistors equal

→ Generates 2^B equally spaced voltages ready to be chosen based on the digital input word



3-Bit Resistor String DAC

R-String DAC Example

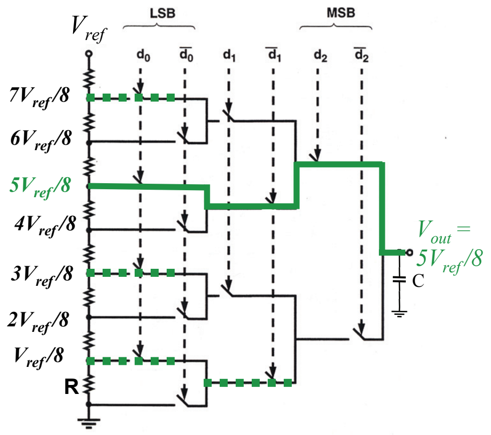
Example:

- Input code: [d2 d1 d0]=***101***

→ $V_{out}= 5V_{ref}/8$

- Assuming switch resistance << ***R:***

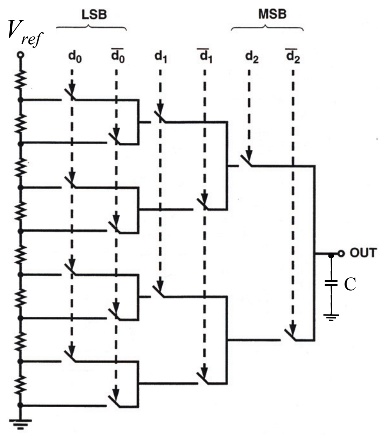
$\tau_{settling} = (3R||5R) \times C$
$=0.23 \times 8RC$



R-String DAC

- Advantages:
  - Takes full advantage of availability of almost perfect switches in MOS technologies
  - Simple, fast for <8-10bits
  - Inherently monotonic
  - Compatible with purely digital technologies
- Disadvantages:
  - 2^B resistors & ~$2\text{x}2^B$ switches for B bits → High element count & large area for B >10bits
  - High settling time for high resolution DACs:

    τ_{max} ~ 0.25 x 2^B RC



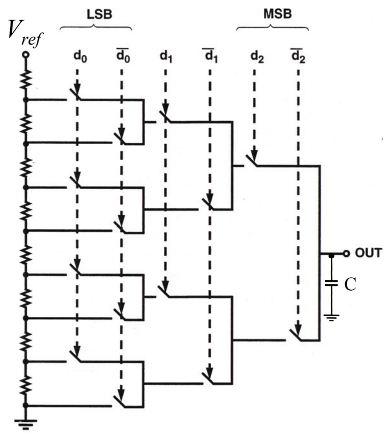
Ref:
M. Pelgrom, "A 10-b 50-MHz CMOS D/A Converter with 75-W Buffer," JSSC, Dec. 1990, pp. 1347

R-String DAC

•Choice of resistor value:
- Since maximum output settling time:

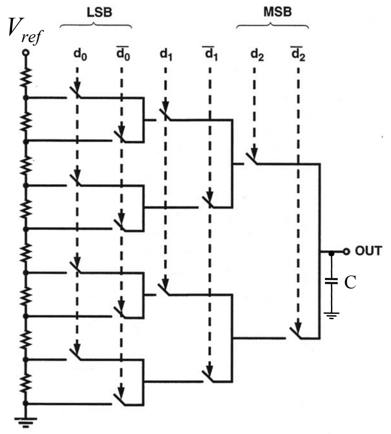
  τ_{max} ~ *0.25 x 2^B RC*
- Choice of resistor value directly affects DAC maximum operating speed
- Power dissipation: function of $V_{ref}^2/(Rx2^B)$

→ Tradeoff between speed and power dissipation



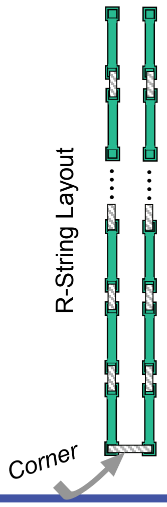
R-String DAC

- Resistor type:
  - Choice of resistive material important
  - Diffusion type R → high temp. co. & voltage co.
    - Results in poor INL/DNL
  - Better choice is poly resistor beware of poly R 1/f noise
  - At times, for high-frequency & high performance DACs, metal R (beware of high temp. co.) or thin film R is used



R-String DAC Layout Considerations

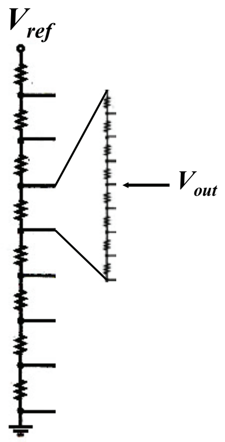
- Number of resistor segments → 2^B
  - E.g. 10-bit R-string DAC →1024 resistors
- Low INL/DNL dictates good R matching
- Layout quite a challenge
  - Good matching mandates all R segments either vertical or horizontal - not both
  - Matching of metal interconnect and contacts
  - Need to fold the string
    - Difficult to match corner segments to rest
    - Could result in large INL/DNL



R-String DAC Including Interpolation

Resistor string DAC
+ Resistor string interpolator increases resolution w/o drastic increase in complexity
e.g. 10bit DAC→ (5bit +5bit→ $2x2^5=2^6$ # of Rs) instead of direct 10bit→2^{10}

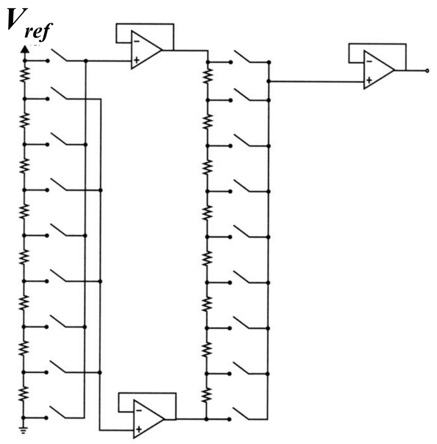
Considerations:
- ❑Main R-string loaded by the interpolation string
- ❑Large R values for interpolating string→ less loading but lower speed
- ❑Can use buffers



R-String DAC Including Interpolation

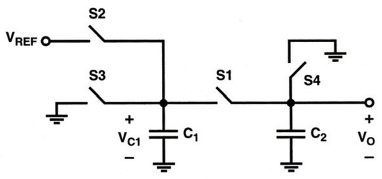
Use buffers to prevent loading of the main ladder

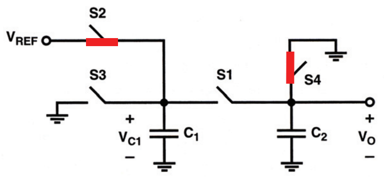
Issues:
- → Buffer DC offset
- → Buffer bandwidth limitation effect on overall speed

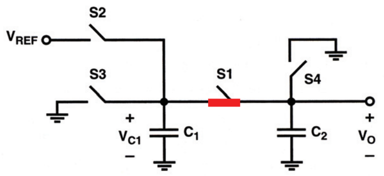


Charge Based: Serial Charge Redistribution DAC Simplified Operation

Nominally $C_1=C_2$



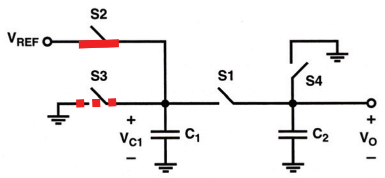


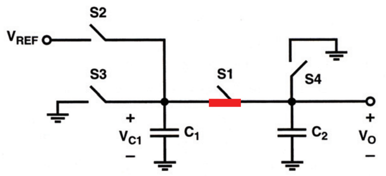


- Conversion sequence:
  - Initialize: Discharge C2 & charge C1 to V_{REF} → S2& S4 closed
  - Charge share: close S1→ $V_{C2}=V_{C1}=V_{REF}/2$

Serial Charge Redistribution DAC Simplified Operation (Cont'd)

Nominally $C_1=C_2$



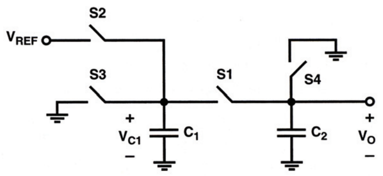


- Conversion sequence:
  - Next cycle
    - If S3 closed $V_{C1}=0$ then when S1 closes $V_{C1} = V_{C2} = V_{REF}/4$
    - If S2 closed $V_{C1}=V_{REF}$ then when S1 closes $V_{C1} =V_{C2} =V_{REF}/2+V_{REF}/4$

Serial Charge Redistribution DAC

- Nominally $C_1=C_2$
- Conversion sequence:
  - Discharge C1 & C2→ S3& S4 closed
  - For each bit in succession beginning with LSB, b_N:
    - S1 open- if $b_i=1$ C1 precharge to V_{REF} if $b_i=0$ discharged to GND
    - S2 & S3 & S4 open- S1 closed- Charge sharing C1 & C2

      → ½ of precharge on C1 +½ of charge previously stored on C2→ C2



$$V_o(1) = \frac{b_N}{2}V_{REF}$$

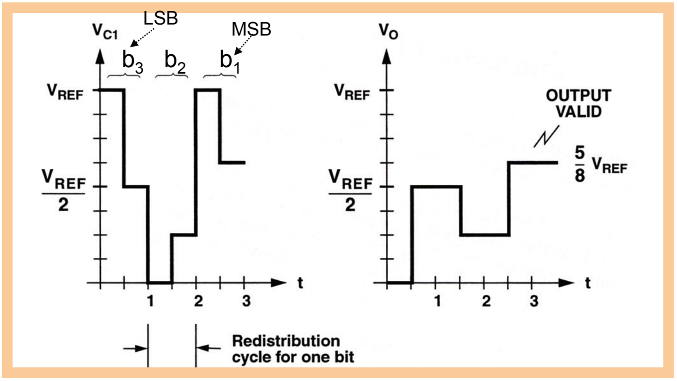
$$V_o(2) = \frac{1}{2}\left(b_{N-1} + \frac{b_N}{2}\right)V_{REF}$$

$$\vdots$$

$$V_o(N) = \left(\sum_{i=1}^{N}\frac{b_i}{2^i}\right)V_{REF}$$

Serial Charge Redistribution DAC

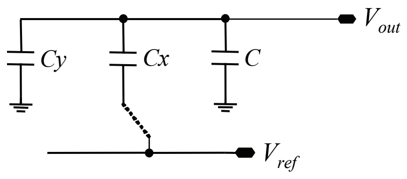
Example: Input Code 101



- Example input code 101→ output (1/8 +0/8 +4/8)V_{REF} =5/8 V_{REF}
- Very small area
- N redistribution cycles for N-bit conversion → quite slow

Parallel Charge Scaling DAC

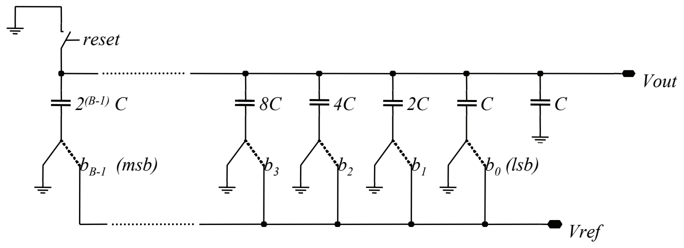
- DAC operation based on capacitive voltage division



$$V_{out} = \frac{Cx}{Cx+Cy+C} V_{ref}$$

→ Make *Cx* & *Cy* function of incoming DAC digital word

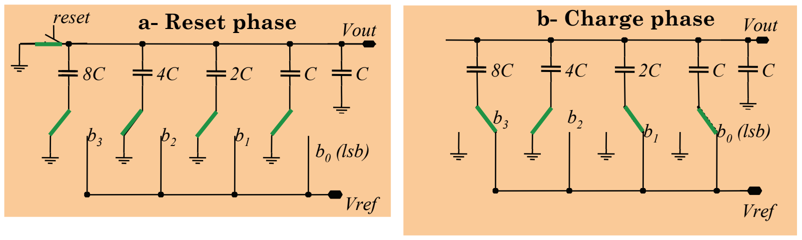
Parallel Charge Scaling DAC



- E.g. "Binary weighted"
- B+1 capacitors & switches (Cs built of unit elements → 2^B units of C)

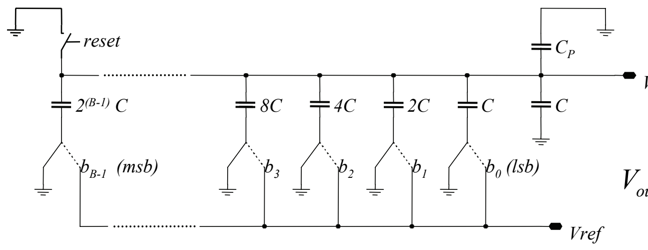
$$V_{out} = \frac{\sum_{i=0}^{B-1} b_i 2^i C}{2^B C} V_{ref}$$

Charge Scaling DAC
Example: 4Bit DAC- Input Code 1011



$$V_{out} = \frac{2^0 C + 2^1 C + 2^3 C}{2^4 C} V_{ref} = \frac{11}{16} V_{ref}$$

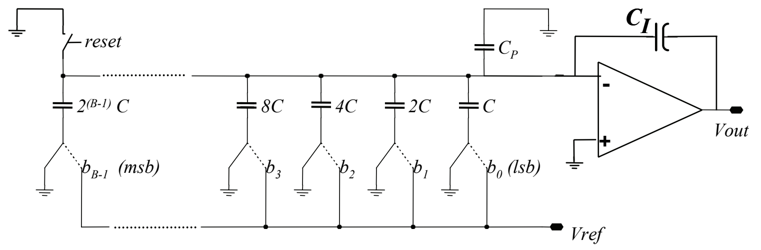
Charge Scaling DAC



$$V_{out} = \frac{\sum_{i=0}^{B-1} b_i 2^i C}{2^B C + C_P} V_{ref}$$

- Sensitive to parasitic capacitor @ output
  - If C_p constant → gain error
  - If C_p voltage dependant → DAC nonlinearity
- Large area of caps for high DAC resolution (10bit DAC ratio 1:512)
- Monotonicity depends on element matching (more later)

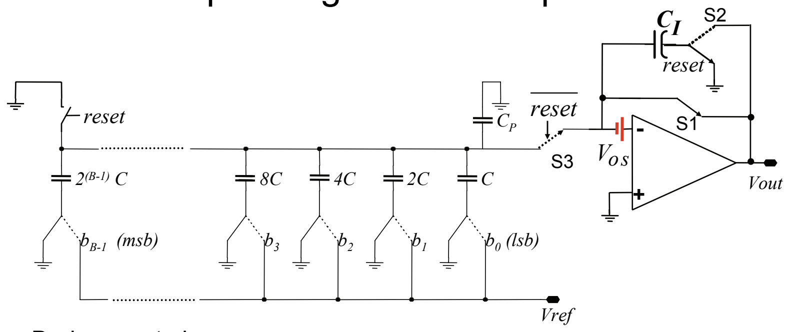
Parasitic Insensitive Charge Scaling DAC



$$V_{out} = \frac{\sum_{i=0}^{B-1} b_i 2^i C}{C_I} V_{ref} \ , \quad C_I = 2^B C \ \rightarrow V_{out} = \frac{\sum_{i=0}^{B-1} b_i 2^i}{2^B} V_{ref}$$

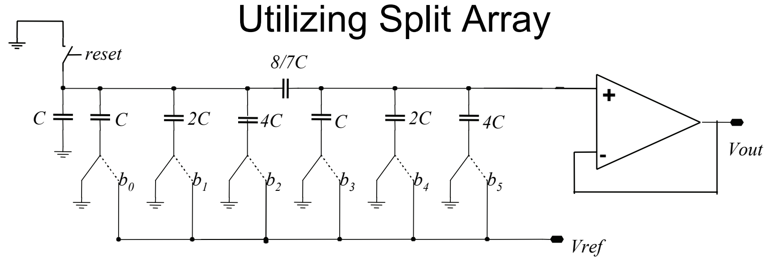
- Opamp helps eliminate the parasitic capacitor effect by producing virtual ground at the sensitive node since C_P has zero volts at start & end
  - Issue: opamp offset & speed

Charge Scaling DAC Incorporating Offset Compensation



- During reset phase:
  - Opamp disconnected from capacitor array via switch S3
  - Opamp connected in unity-gain configuration (S1)
  - C_I Bottom plate connected to ground (S2)
  - $V_{out} \sim - V_{os} \rightarrow V_{CI} = -V_{os}$
- This effectively compensates for offset during normal phase

Charge Scaling DAC Utilizing Split Array



$$C_{series} = \frac{\sum all\, LSB\, array\, C}{\sum all\, MSB\, array\, C} C$$

- Split array→ reduce the total area of the capacitors required for high resolution DACs
  - E.g. 10bit regular binary array requires 1024 unit Cs while split array (5&5) needs 64 unit Cs
  - Issue: Sensitive to parasitic capacitor

Charge Scaling DAC

- Advantages:
  - Low power dissipation → capacitor array does not dissipate DC power
  - Output is sample and held → no need for additional S/H
  - INL function of capacitor ratio
  - Possible to trim or calibrate for improved INL
  - Offset cancellation almost for free
- Disadvantages:
  - Process needs to include good capacitive material → not compatible with standard digital process
  - Requires large capacitor ratios
  - Not inherently monotonic (more later)