# EE247
# Lecture 9

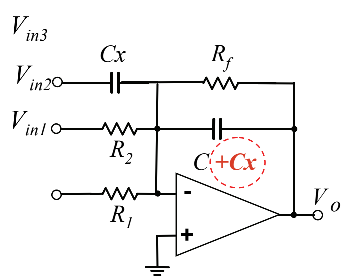
- Continuous-time filters (continued)
  - Various Gm-C filter implementations
  - Performance comparison of various continuous-time filter topologies
- Switched-capacitor filters
  - Emulating a resistor by using a switched capacitor
  - Tradeoffs in choosing sampling rate
  - Effect of sample and hold
  - Switched-capacitor network electronic noise
  - Switched-capacitor integrators
    - DDI integrators
    - LDI integrators



# Correction to Lecture 4 Slide # 48
## Transmission Zero Generation Opamp-RC Integrator

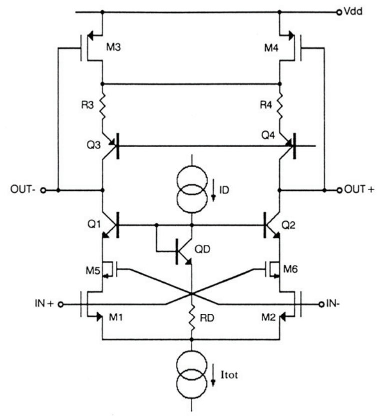
$$V_o = -\frac{1}{s(C+C_x)}\left[\frac{V_{in1}}{R_1}+\frac{V_{in2}}{R_2}+\frac{V_o}{R_f}\right]$$

$$-V_{in3}\times\frac{C_x}{C+C_x}$$



# BiCMOS Gm-C Integrator

- M1,2 → triode mode
- Q1,2 → hold $V_{ds}$ of M1,2 constant
- Current source ID used to tune filter critical frequency by varying $V_{ds}$ of M1,2 and thus controlling *gm* of M1,2
- M3, M4 operate in triode mode and added to provide common-mode feedback
- Needs higher supply voltage compared to the previous design since quite a few devices are stacked vertically

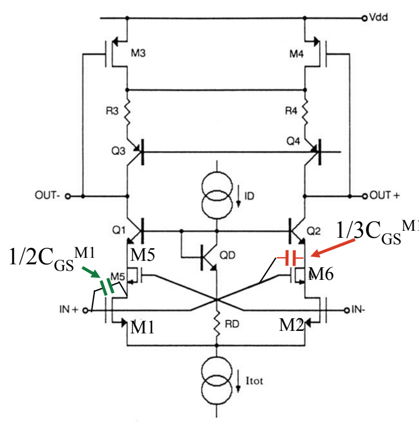



**Ref:** R. Alini, A. Baschirotto, and R. Castello, "Tunable BiCMOS Continuous-Time Filter for High-Frequency Applications," *IEEE Journal of Solid State Circuits,* Vol. 27, No. 12, pp. 1905-1915, Dec. 1992.



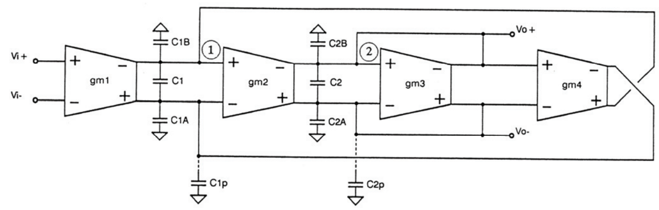
# BiCMOS Gm-C Integrator

- M5 & M6 configured as capacitors- added to compensate for RHP zero due to Cgd of M1,2 (moves it to LHP) size of M5,6 → 1/3 of M1,2




**Ref:** R. Alini, A. Baschirotto, and R. Castello, "Tunable BiCMOS Continuous-Time Filter for High-Frequency Applications," *IEEE Journal of Solid State Circuits,* Vol. 27, No. 12, pp. 1905-1915, Dec. 1992.

# BiCMOS Gm-C Filter For Disk-Drive Application




- Using the integrators shown in the previous page
- Biquad filter for disk drives
- *gm1=gm2=gm4=2gm3*
- *Q=2*
- Tunable from 8MHz to 32MHz

**Ref:** R. Alini, A. Baschirotto, and R. Castello, "Tunable BiCMOS Continuous-Time Filter for High-Frequency Applications," *IEEE Journal of Solid State Circuits,* Vol. 27, No. 12, pp. 1905-1915, Dec. 1992.



# Summary Continuous-Time Filters

- Opamp RC filters
  - Good linearity → High dynamic range (*60-90dB*)
  - Only discrete tuning possible
  - Medium usable signal bandwidth (<*10MHz*)
- Opamp MOSFET-C
  - Linearity compromised (typical dynamic range *40-60dB*)
  - Continuous tuning possible
  - Low usable signal bandwidth *(<5MHz*)
- Opamp MOSFET-RC
  - Improved linearity compared to Opamp MOSFET-C (D.R. *50-90dB*)
  - Continuous tuning possible
  - Low usable signal bandwidth *(<5MHz*)
- Gm-C
  - Highest frequency performance -at least an order of magnitude higher compared to other integrator-based active filters *(<100MHz*)
  - Dynamic range not as high as Opamp RC but better than Opamp MOSFET-C (*40-70dB*)

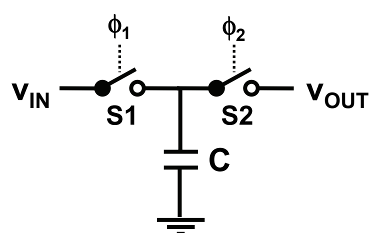
# Switched-Capacitor Filters

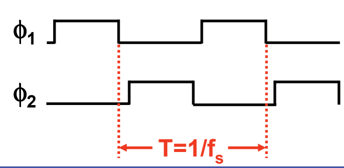
- SC filters are sampled-data type circuits operating with continuous signal amplitude & quantized time
- Emulating resistor via switched-capacitor network
- $1^{st}$ order switched-capacitor filter
- Switch-capacitor filter considerations:
  - Issue of aliasing and how to prevent aliasing
  - Tradeoffs in choice of sampling rate
  - Effect of sample and hold
  - Switched-capacitor filter electronic noise



# Switched-Capacitor Resistor

- Capacitor C is the "switched capacitor"
- Non-overlapping clocks $\phi_1$ and $\phi_2$ control switches S1 and S2, respectively
- $v_{IN}$ is sampled at the falling edge of $\phi_1$
  - Sampling frequency $f_S$
- Next, $\phi_2$ rises and the voltage across C is transferred to $v_{OUT}$
- Why does this behave as a resistor?



# Switched-Capacitor Resistors

- Charge transferred from $v_{IN}$ to $v_{OUT}$ during each clock cycle is:

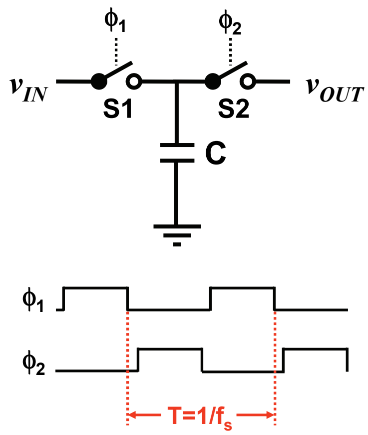
$$Q = C(v_{IN} - v_{OUT})$$

- Average current flowing from $v_{IN}$ to $v_{OUT}$ is:

$$i = Q/t = Q \cdot f_s$$

Substituting for $Q$:

$$i = f_S C(v_{IN} - v_{OUT})$$






# Switched-Capacitor Resistors

$$i = f_S C(v_{IN} - v_{OUT})$$

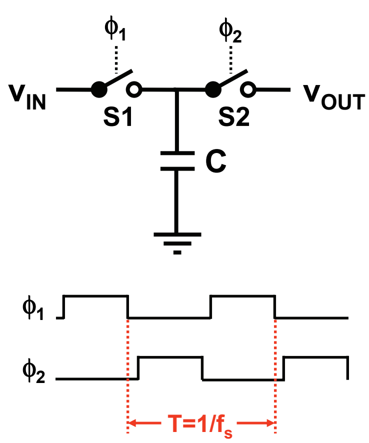
With the current through the switched-capacitor resistor proportional to the voltage across it, the equivalent "switched capacitor resistance" is:

$$R_{eq} = \frac{v_{IN} - v_{OUT}}{i} = \frac{1}{f_S C}$$

*Example:*

$$f_S = 100kHz, C = 0.1pF$$
$$\rightarrow R_{eq} = 100Mega\Omega$$

Note: Can build large time-constant in small area

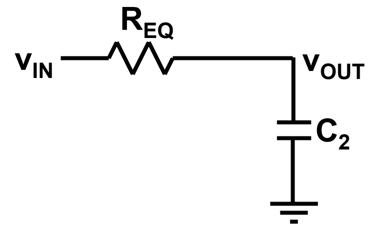


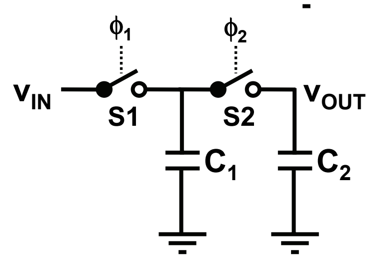
# Switched-Capacitor Filter

- Let's build a "switched- capacitor " filter …
- Start with a simple RC LPF
- Replace the physical resistor by an equivalent switched-capacitor resistor
- 3-dB bandwidth:

$$\omega_{-3dB} = \frac{1}{R_{eq}C_2} = f_s \times \frac{C_1}{C_2}$$

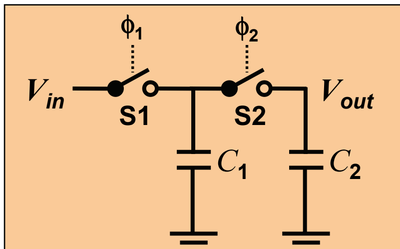
$$f_{-3dB} = \frac{1}{2\pi} f_s \times \frac{C_1}{C_2}$$



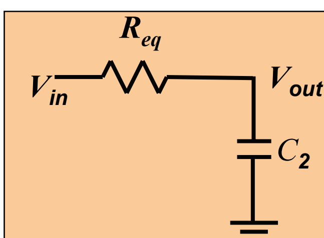



# Switched-Capacitor Filter Advantage versus Continuous-Time Filter




$$f_{-3dB} = \frac{1}{2\pi} f_s \times \frac{C_1}{C_2}$$

- Corner freq. proportional to:
  System clock (accurate to few ppm)
  C ratio accurate → < 0.1%



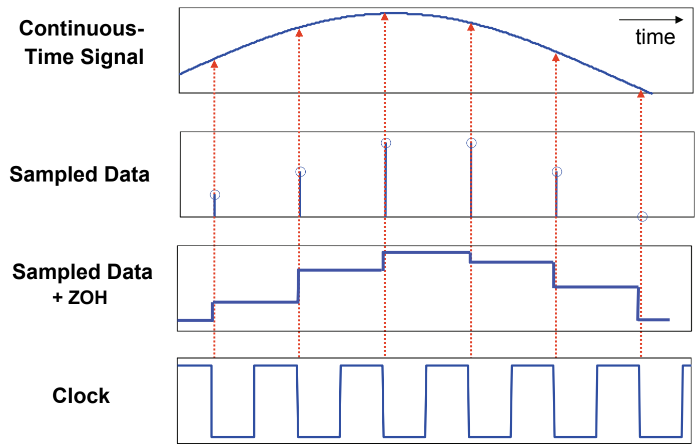

$$f_{-3dB} = \frac{1}{2\pi} \times \frac{1}{R_{eq}C_2}$$

- Corner freq. proportional to:
  Absolute value of Rs & Cs
  Poor accuracy → 20 to 50%

***Main advantage of SC filters → inherent corner frequency accuracy***

# Typical Sampling Process

## Continuous-Time(CT) ⇒ Sampled Data (SD)



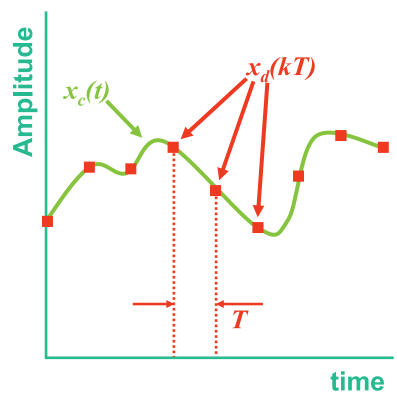



# Uniform Sampling

Nomenclature:

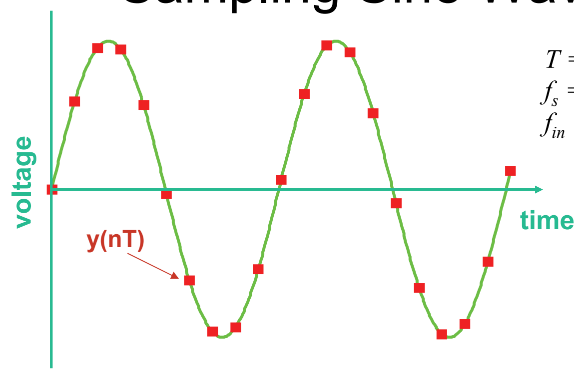
Continuous time signal $x_c(t)$
Sampling interval $T$
Sampling frequency $f_s = 1/T$
Sampled signal $x_d(kT) = x(k)$

- Problem: Multiple continuous time signals can yield exactly the same discrete time signal
- Let's look at samples taken at 1µs intervals of several sinusoidal waveforms …




Note: Samples are the waveform values at $kT$ instances and undefined in between

# Sampling Sine Waves




$T = 1\mu s$
$f_s = 1/T = 1MHz$
$f_{in} = 101kHz$

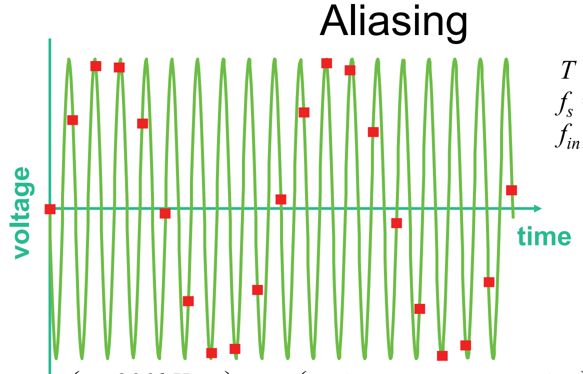
$$v(t) = cos\,(2\pi.f_{in}.t)$$

Sampled-data domian $\rightarrow t \rightarrow n.T$ or $t \rightarrow n/f_s$

$$v(n) = cos\left(2\pi.\frac{f_{in}}{f_s}.n\right) = cos\left(2\pi.\frac{101kHz}{1MHz}.n\right)$$



# Sampling Sine Waves
# Aliasing

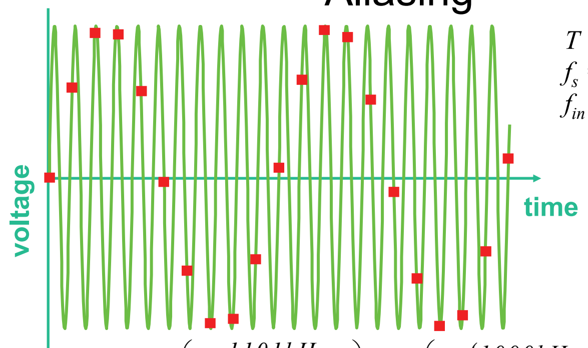



$T = 1\mu s$
$f_s = 1/T = 1MHz$
$f_{in} = 899kHz$

$$v(n) = cos\left(2\pi.\frac{899kHz}{1MHz}.n\right) = cos\left(2\pi.\frac{(1000kHz\text{-}101kHz)}{1MHz}.n\right) = cos\left(2\pi - \frac{101kHz}{1MHz}.n\right)$$

$$= cos\left(2\pi.\frac{101kHz}{1MHz}.n\right)$$

# Sampling Sine Waves
## Aliasing




$T = 1\mu s$
$f_s = 1/T = 1MHz$
$f_{in} = 1101kHz$

$$v(n) = \cos\left(2\pi.\frac{1101kHz}{1MHz}.n\right) = \cos\left(2\pi.\frac{(1000kHz + 101kHz)}{1MHz}.n\right)$$

$$= \cos\left(2\pi.\frac{101kHz}{1MHz}.n\right)$$



# Sampling Sine Waves

Problem:

Identical samples for:

$v(t) = \cos\,[2\pi f_{in} t\,]$

$v(t) = \cos\,[2\pi(n.f_s + f_{in})t\,]$

$v(t) = \cos\,[2\pi(n.f_s - f_{in})t\,]$

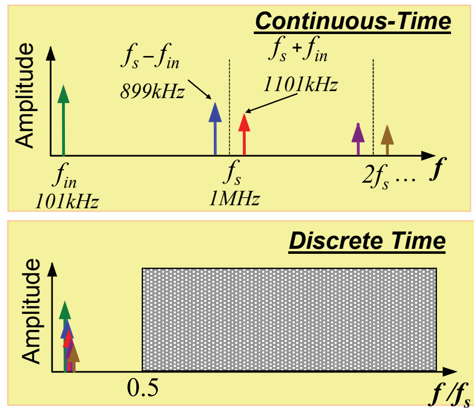
* (n-integer)

→Multiple continuous time signals can yield exactly the same discrete time signal

## Sampling Sine Waves Frequency Spectrum

Signal scenario before sampling

Signal scenario after sampling



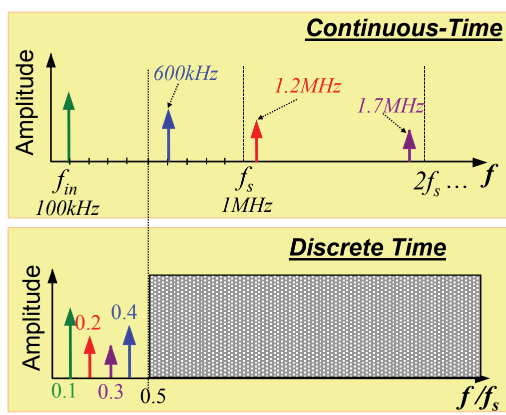

Key point: Signals @ $nf_S \pm f_{max\_signal}$ fold back into band of interest→Aliasing



## Sampling Sine Waves Frequency Spectrum

Signal scenario before sampling

Signal scenario after sampling




Key point: Signals @ $nf_S \pm f_{max\_signal}$ fold back into band of interest→Aliasing

# Aliasing

- Multiple continuous time signals can produce identical series of samples
- The folding back of signals from $nf_s \pm f_{sig}$ (n integer) down to $f_{fin}$ is called aliasing
  - Sampling theorem: $f_s > 2f_{max\_Signal}$
- If aliasing occurs, no signal processing operation downstream of the sampling process can recover the original continuous time signal



# How to Avoid Aliasing?

- Must obey sampling theorem:

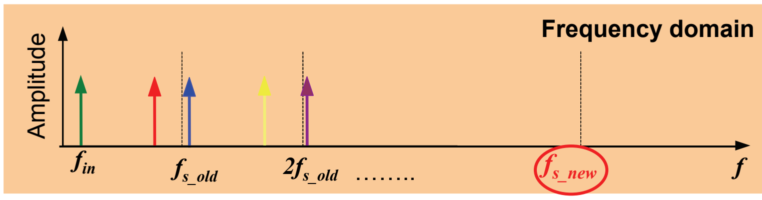
$$f_{max\text{-}Signal} < f_s/2$$

*Note:

Minimum sampling rate of $f_s = 2x f_{max\text{-}Signal}$ is called Nyquist rate

- Two possibilities:
  1. Sample fast enough to cover all spectral components, including "parasitic" ones outside band of interest
  2. Limit $f_{max\_Signal}$ through filtering → attenuate out-of-band components prior to sampling

## How to Avoid Aliasing?
## 1-Sample Fast

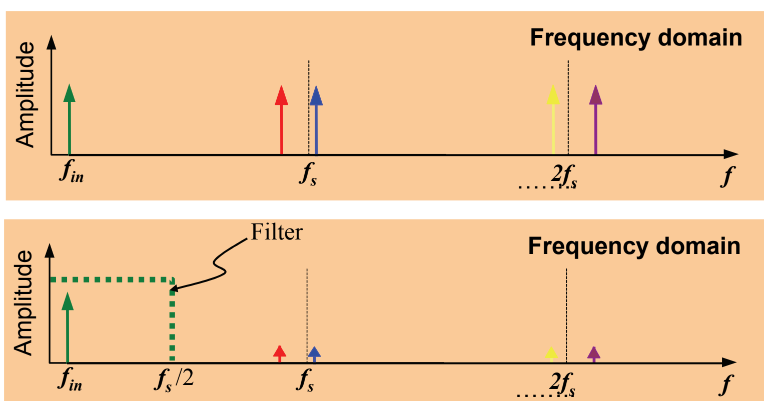



Push sampling frequency to *x2* of the highest frequency signal to cover all unwanted signals as well as wanted signals

→ In vast majority of cases not practical

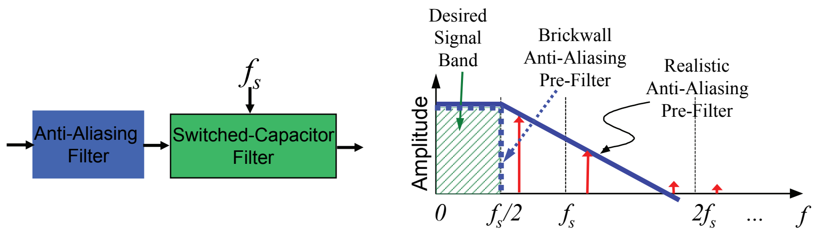


## How to Avoid Aliasing?
## 2-Filter Out-of-Band Signal Prior to Sampling




**Pre-filter signal to eliminate/attenuate signals above $f_s/2$- then sample**

# Anti-Aliasing Filter Considerations

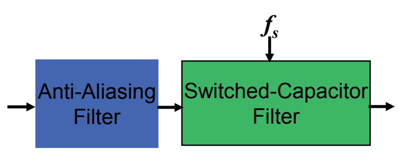



Case1- $B = f_{sig}^{max} = f_s/2$

- Non-practical since an extremely high order anti-aliasing filter (close to an ideal brickwall filter) is required
- Practical anti-aliasing filter →Non-zero filter "transition band"
- In order to make this work, we need to sample much faster than 2x the signal bandwidth
  →"Oversampling"

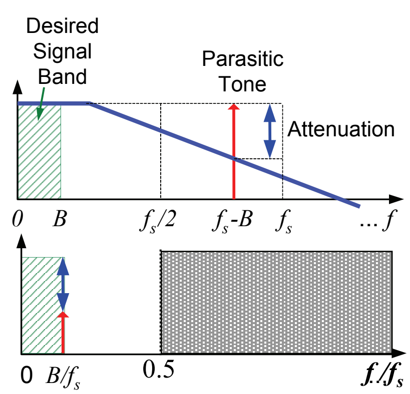


# Practical Anti-Aliasing Filter

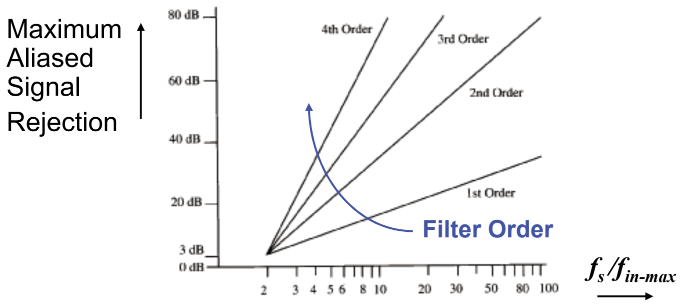



Case2 - $B = f_{max\_Signal} << f_s/2$

- More practical anti-aliasing filter
- Preferable to have an anti-aliasing filter with:
  →The lowest order possible
  →No frequency tuning required (if frequency tuning is required then why use switched-capacitor filter, just use the prefilter!?)



## Tradeoff
## Oversampling Ratio versus Anti-Aliasing Filter Order



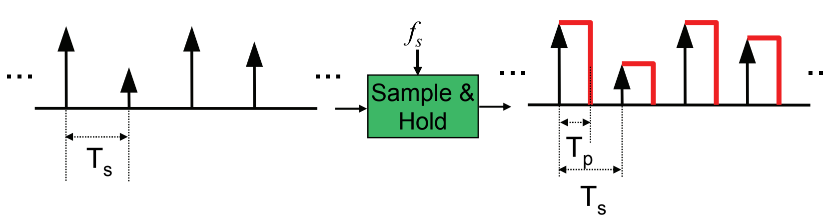

* *Assumption → anti-aliasing filter is Butterworth type (not a necessary requirement)*

→Tradeoff: Sampling frequency versus anti-aliasing filter order

Ref: R. v. d. Plassche, *CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters*, 2nd ed., Kluwer publishing, 2003, p.41



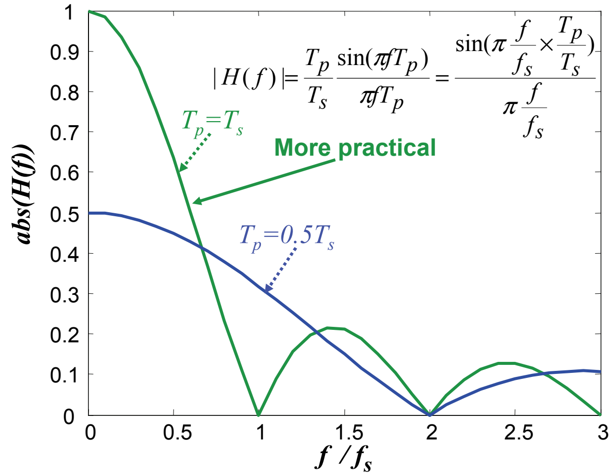
# Effect of Sample & Hold




- Using the Fourier transform of a rectangular impulse:

$$|H(f)| = \frac{T_p}{T_s}\frac{\sin(\pi f T_p)}{\pi f T_p} \longrightarrow \frac{\sin x}{x} \text{ shape}$$

# Effect of Sample & Hold on Frequency Response




$$|H(f)| = \frac{T_p}{T_s}\frac{\sin(\pi f T_p)}{\pi f T_p} = \frac{\sin(\pi \frac{f}{f_s} \times \frac{T_p}{T_s})}{\pi \frac{f}{f_s}}$$

$$|H(f=0)| = \frac{T_p}{T_s}$$

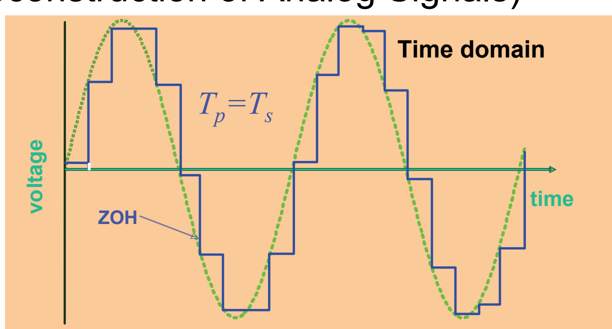
$$|H(f = n f_s \frac{T_s}{T_p})| = 0$$

$n \rightarrow$ integer

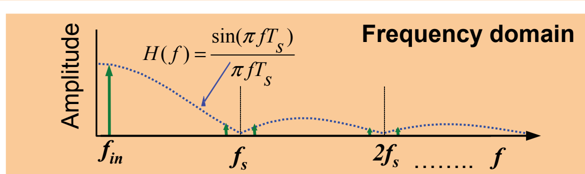


# Sample & Hold Effect (Reconstruction of Analog Signals)

$T_p=T_s$



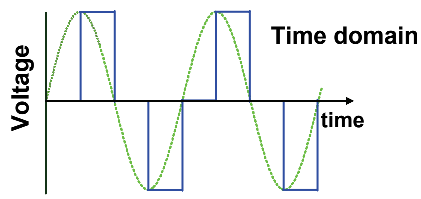

Magnitude droop due to *sinx/x* effect



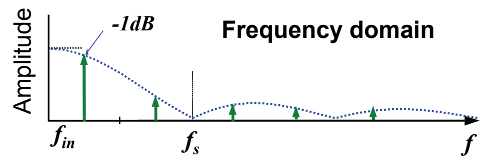
# Sample & Hold Effect (Reconstruction of Analog Signals)

Magnitude droop due to $sinx/x$ effect:

Case 1) $f_{sig}=f_s/4$




Droop= *-1dB*

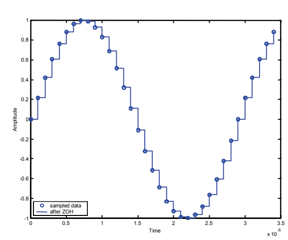





# Sample & Hold Effect (Reconstruction of Analog Signals)

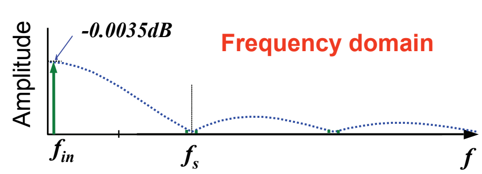
Magnitude droop due to $sinx/x$ effect:

Case 2) $f_{sig}=f_s/32$

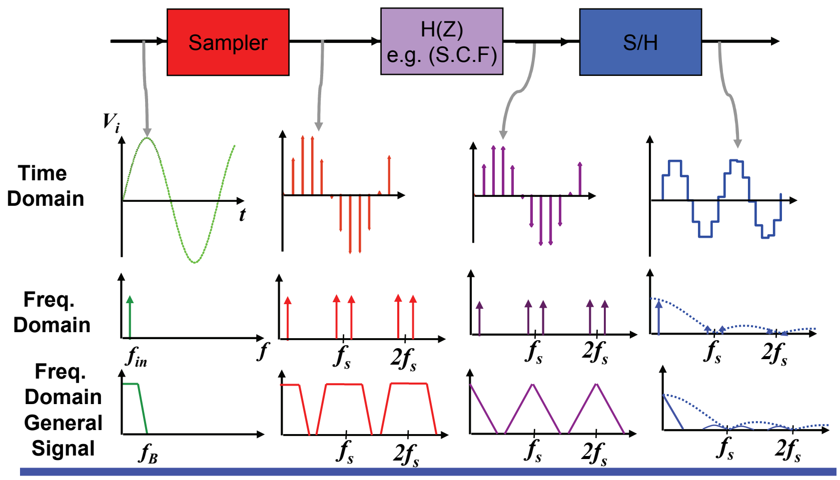



Droop= *-0.0035dB*

- ***Insignificant droop***
***→ High oversampling ratio desirable***



# Sampling Process Including S/H





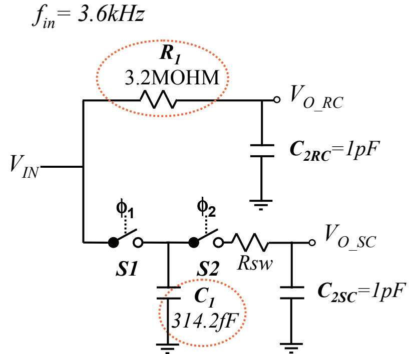

# 1st Order Filter S.C. versus C.T. Transient Analysis

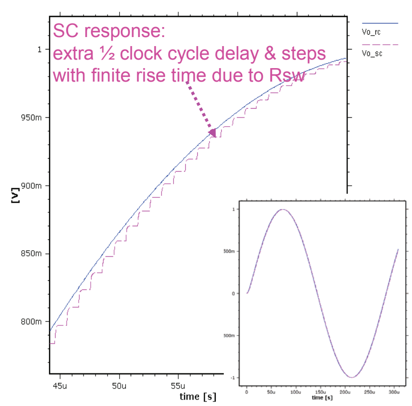
1st Order RC versus SC LPF

$f_s = 1MHz$

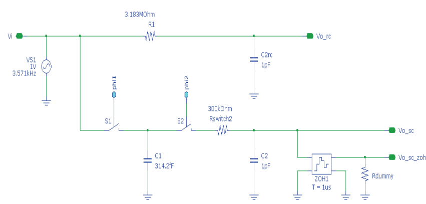
$f_{-3dB}$=50kHz

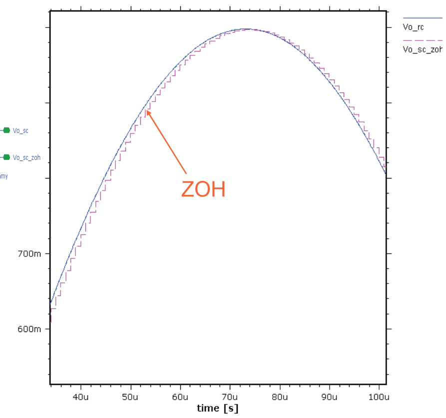
$f_{in}$= 3.6kHz



# 1st Order Filter Transient Analysis

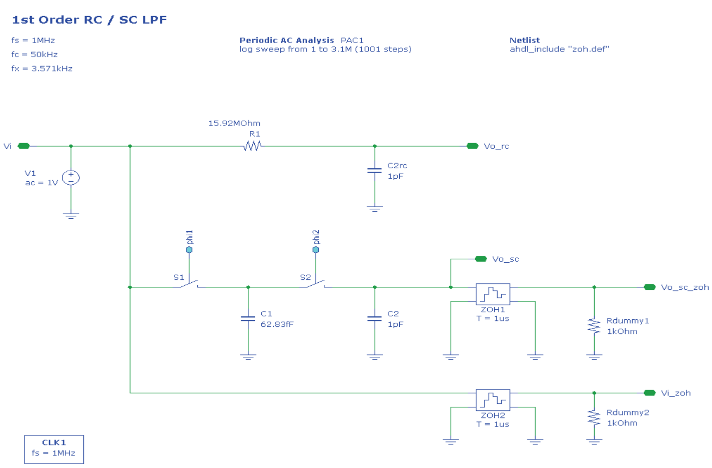


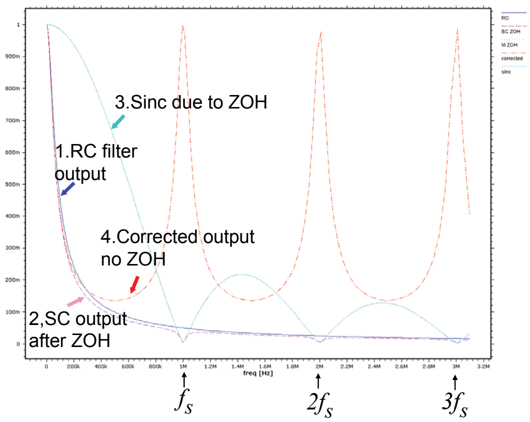

- ZOH: Emulates an ideal S/H→ pick signal after settling (usually at end of clock phase)
- Adds delay and sin(x)/x shaping
- When in doubt, use a ZOH in periodic ac simulations



# Periodic AC Analysis



# $1^{st}$ Order Filter Magnitude Response




1. RC filter output
2. SC output after ZOH
3. Output after single ZOH
4. Output w/o effect of ZOH
   - (2) over (3)
   - Repeats filter shape around $nf_s$
   - Identical to RC for $f << f_s/2$



# Periodic AC Analysis

- SPICE frequency analysis
  - ac linear, time-invariant circuits
  - pac linear, time-variant circuits
- SpectreRF statements

```
V1 ( Vi 0 ) vsource type=dc dc=0 mag=1 pacmag=1
PSS1 pss period=1u errpreset=conservative
PAC1 pac start=1 stop=1M lin=1001
```

- Output
  - Divide results by sinc($f/f_s$) to correct for ZOH distortion

# Spectre Circuit File

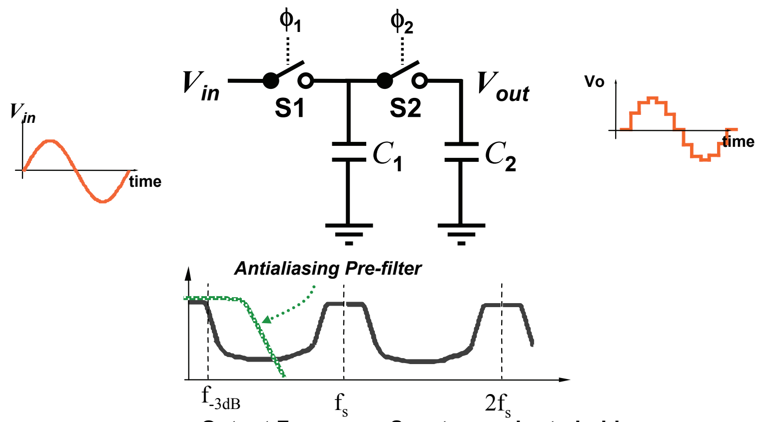
```
rc_pac
simulator lang=spectre
ahdl_include "zoh.def"

S1 ( Vi c1 phi1 0 ) relay ropen=100G rclosed=1 vt1=-500m vt2=500m
S2 ( c1 Vo_sc phi2 0 ) relay ropen=100G rclosed=1 vt1=-500m vt2=500m
C1 ( c1 0 ) capacitor c=314.159f
C2 ( Vo_sc 0 ) capacitor c=1p
R1 ( Vi Vo_rc ) resistor r=3.1831M
C2rc ( Vo_rc 0 ) capacitor c=1p
CLK1_Vphi1 ( phi1 0 ) vsource type=pulse val0=-1 val1=1 period=1u
                          width=450n delay=50n rise=10n fall=10n
CLK1_Vphi2 ( phi2 0 ) vsource type=pulse val0=-1 val1=1 period=1u
                          width=450n delay=550n rise=10n fall=10n
V1 ( Vi 0 ) vsource type=dc dc=0 mag=1 pacmag=1
PSS1 pss period=1u errpreset=conservative
PAC1 pac start=1 stop=3.1M log=1001
ZOH1 ( Vo_sc_zoh 0 Vo_sc 0 ) zoh period=1u delay=500n aperture=1n tc=10p
ZOH2 ( Vi_zoh 0 Vi 0 ) zoh period=1u delay=0 aperture=1n tc=10p
```



# ZOH Circuit File

```
// Copy from the SpectreRF Primer

module zoh (Pout, Nout, Pin, Nin) (period,
     delay, aperture, tc)

node [V,I] Pin, Nin, Pout, Nout;
parameter real period=1 from (0:inf);
parameter real delay=0 from [0:inf);
parameter real aperture=1/100 from (0:inf);
parameter real tc=1/500 from (0:inf);
{
integer n; real start, stop;
node [V,I] hold;
  analog {
    // determine the point when aperture
     begins
    n = ($time() - delay + aperture) / period
     + 0.5;
    start = n*period + delay - aperture;
    $break_point(start);

    // determine the time when aperture ends
    n = ($time() - delay) / period + 0.5;
    stop = n*period + delay;
    $break_point(stop);

    // Implement switch with effective series
    // resistence of 1 Ohm
    if ( ($time() > start) && ($time() <= stop))
      I(hold) <- V(hold) - V(Pin, Nin);
    else
      I(hold) <- 1.0e-12 * (V(hold) - V(Pin, Nin));

    // Implement capacitor with an effective
    // capacitance of tc
    I(hold) <- tc * dot(V(hold));

    // Buffer output
    V(Pout, Nout) <- V(hold);

    // Control time step tightly during
    // aperture and loosely otherwise
    if (($time() >= start) && ($time() <= stop))
      $bound_step(tc);
    else
      $bound_step(period/5);
  }
}
```

# First Order S.C. Filter




**Output Frequency Spectrum prior to hold**

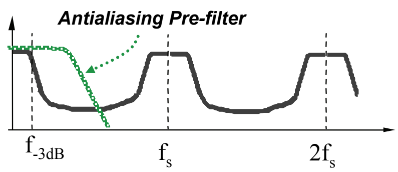
Switched-Capacitor Filters → problem with aliasing



# Sampled-Data Systems (Filters) Anti-aliasing Requirements

- Frequency response repeats at $f_s$, $2f_s$, $3f_s$.....
- High frequency signals close to $f_s$, $2f_s$,….folds back into passband (aliasing)
- Most cases must pre-filter input to a sampled-data systems (filter) to attenuate signal at:
  $f > f_s/2$ (nyquist → $f_{max} < f_s/2$ )
- Usually, anti-aliasing filter → included on-chip as continuous-time filter with relaxed specs. (no tuning)

## Example : Anti-Aliasing Filter Requirements

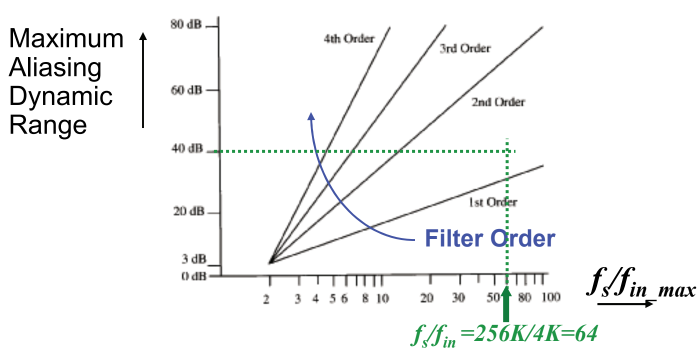



- Voice-band CODEC S.C. filter $f_{-3dB}=4kHz$ & $f_s=256kHz$
- Anti-aliasing filter requirements:
  - Need at least 40dB attenuation of all out-of-band signals which can alias inband
  - Incur no phase-error from 0 to 4kHz
  - Gain error due to anti-aliasing filter → 0 to 4kHz < 0.05dB
  - Allow +-30% variation for anti-aliasing filter corner frequency (no tuning)

**Need to find minimum required filter order**



## Oversampling Ratio versus Anti-Aliasing Filter Order



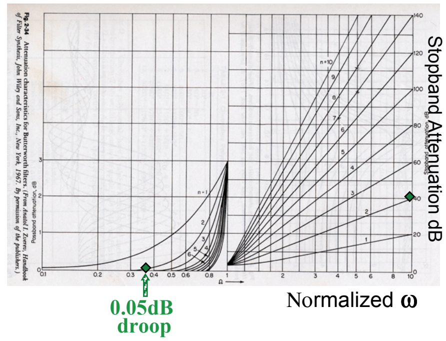

* *Assumption → anti-aliasing filter is Butterworth type*

→ $2^{nd}$ order Butterworth

→ Need to find minimum corner frequency for mag. droop < 0.05dB

## Example : Anti-Aliasing Filter Specifications

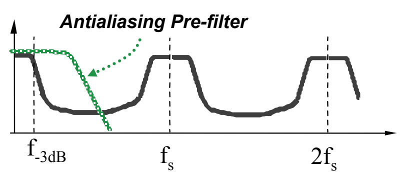
- Normalized frequency for 0.05dB droop: need perform passband simulation→ 0.34→ 4kHz/0.34=12kHz
- Set anti-aliasing filter corner frequency for minimum corner frequency 12kHz → Find nominal corner frequency: 12kHz/0.7=17.1kHz
- Check if attenuation requirement is satisfied for widest filter bandwidth → 17.1x1.3=22.28kHz
- Find $(f_s\text{-}f_{sig})/f_{-3dB}^{\ max}$ →252/22.2=11.35→ make sure enough attenuation
- Check phase-error within 4kHz bandwidth for min. filter bandwidth via simulation




From: Williams and Taylor, p. 2-37



## Example : Anti-Aliasing Filter




- Voice-band S.C. filter $f_{-3dB}=4kHz$ & $f_s=256kHz$
- Anti-aliasing filter requirements:
  - Need 40dB attenuation at clock freq.
  - Incur no phase-error from 0 to 4kHz
  - Gain error 0 to 4kHz < 0.05dB
  - Allow +-30% variation for anti-aliasing corner frequency (no tuning)

→2-pole Butterworth LPF with nominal corner freq. of 17kHz & no tuning (min.=12kHz & max.=22kHz corner frequency )

# Summary

- Sampling theorem → $f_s > 2f_{max\_Signal}$
- Signals at frequencies $nf_S \pm f_{sig}$ fold back down to desired signal band, $f_{sig}$
  - → This is called <u>aliasing</u> & usually mandates use of anti-aliasing pre-filters
- Oversampling helps reduce required order for anti-aliasing filter
- S/H function shapes the frequency response with <u>$sinx/x$</u> shape
  - → Need to pay attention to droop in passband due to <u>$sinx/x$</u>
- If the above requirements are not met, CT signals can NOT be recovered from sampled-data networks without loss of information

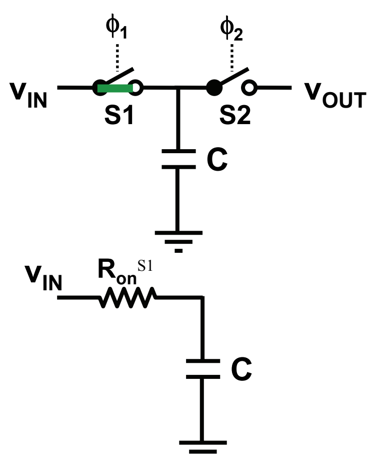


# Switched-Capacitor Network Noise

- During $\phi_1$ high: Resistance of switch S1 ($R_{on}^{S1}$) produces a noise voltage on C with variance $kT/C$ (lecture 1- first order filter noise)
- The corresponding noise charge is:

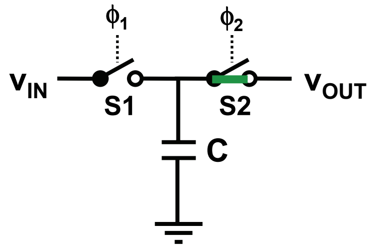
$$Q^2 = C^2V^2 = C^2 . kT/C = kTC$$

- $\phi_1$ low: S1 open →This charge is sampled

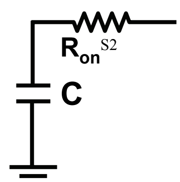


# Switched-Capacitor Noise

- During $\phi_2$ high: Resistance of switch S2 contributes to an uncorrelated noise charge on C at the end of $\phi_2$ : with variance $kT/C$




- Mean-squared noise charge transferred from $v_{IN}$ to $v_{OUT}$ per sample period is:




$$Q^2 = 2kTC$$



# Switched-Capacitor Noise

- The mean-squared noise current due to S1 and S2's $kT/C$ noise is :

$$\text{Since } i = Q/t \quad \text{then} \rightarrow \overline{i^2} = (Qf_s)^2 = 2k_B T C f_s^2$$

- This noise is approximately white and distributed between 0 and $f_s/2$ (noise spectra → single sided by convention)
  The spectral density of the noise is found:

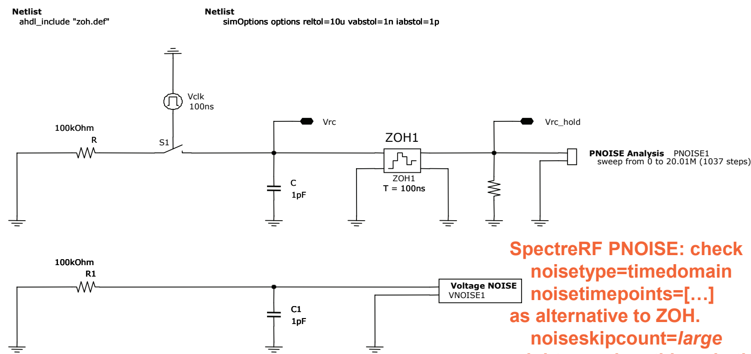
$$\frac{\overline{i^2}}{\Delta f} = \frac{2k_B T C f_s^2}{f_s/2} = 4k_B T C f_s$$

$$\text{Since } R_{EQ} = \frac{1}{f_s C} \quad \text{then:} \quad \boxed{\frac{\overline{i^2}}{\Delta f} = \frac{4k_B T}{R_{EQ}}}$$

**→S.C. resistor noise = a physical resistor noise with same value!**

# Periodic Noise Analysis SpectreRF

### Sampling Noise from SC S/H

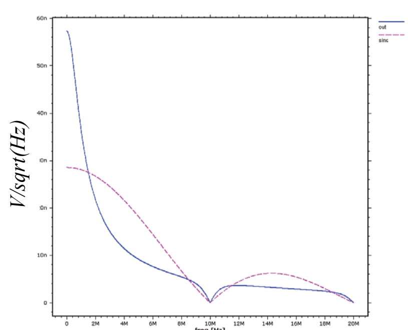



**SpectreRF PNOISE: check**
**noisetype=timedomain**
**noisetimepoints=[…]**
**as alternative to ZOH.**
**noiseskipcount=*large***
**might speed up things in this case.**

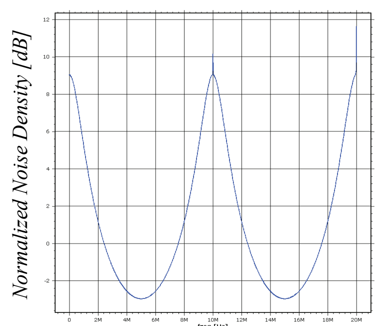
```
PSS pss period=100n maxacfreq=1.5G errpreset=conservative
PNOISE ( Vrc_hold 0 ) pnoise start=0 stop=20M lin=500 maxsideband=10
```



# Sampled Noise Spectrum

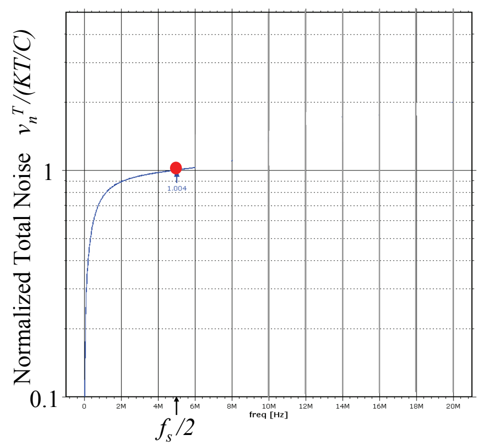



Spectral density of sampled noise including sinx/x effect




Noise spectral density with sinx/x effect taken out

# Total Noise

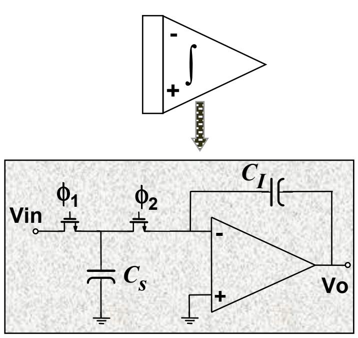


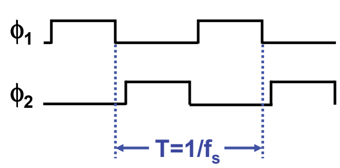

Sampled simulated noise in 0 … $f_s/2$: 62.2μV rms

(expect 64μV for 1pF)



# Switched-Capacitor Integrator



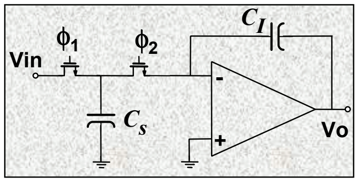

$$for \quad f_{signal} << f_{sampling}$$

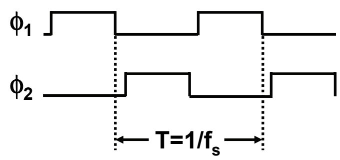
$$\rightarrow V_0 = \frac{f_s \times C_s}{C_I} \int V_{in}\, dt$$

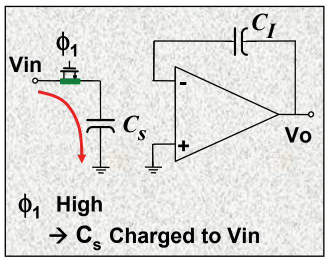
$$\omega_0 = f_s \times \frac{C_s}{C_I}$$

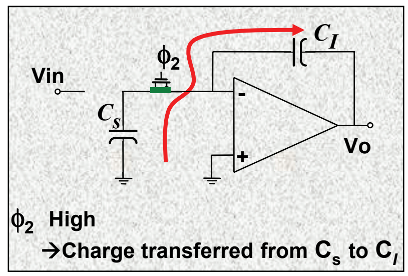
Main advantage: No tuning needed
→ Critical frequency function of ratio of capacitors & clock freq.

# Switched-Capacitor Integrator






# Continuous-Time versus Discrete Time Analysis Approach

**Continuous-Time**

- Write differential equation
- Laplace transform ($F(s)$)
- Let $s=j\omega$ → $F(j\omega)$
- Plot $|F(j\omega)|$, $phase(F(j\omega))$

**Discrete-Time**

- Write difference equation → relates output sequence to input sequence
- Use delay operator $z^{-1}$ to transform the recursive realization to algebraic equation in z domain

$$V_o(nT) = V_i[(n-1)T] \ldots\ldots$$

- Set $z = e^{j\omega T}$
- Plot mag./phase versus frequency

$$V_o(z) = z^{-1} V_i(z) \ldots\ldots$$

# Discrete Time Design Flow

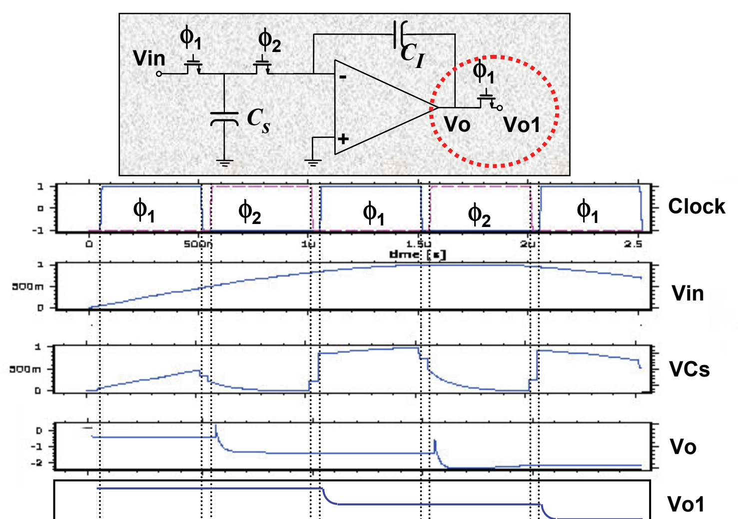
- Transforming the recursive realization to algebraic equation in $z$ domain:
  - Use delay operator $z$ :

$$nT_S \ldots\ldots\ldots\ldots\ldots\ldots\ldots \rightarrow 1$$
$$\left[(n-1)T_S\right] \ldots\ldots\ldots\ldots \rightarrow z^{-1}$$
$$\left[(n-1/2)T_S\right] \ldots\ldots\ldots \rightarrow z^{-1/2}$$
$$\left[(n+1)T_S\right] \ldots\ldots\ldots\ldots \rightarrow z^{+1}$$
$$\left[(n+1/2)T_S\right] \ldots\ldots\ldots \rightarrow z^{+1/2}$$

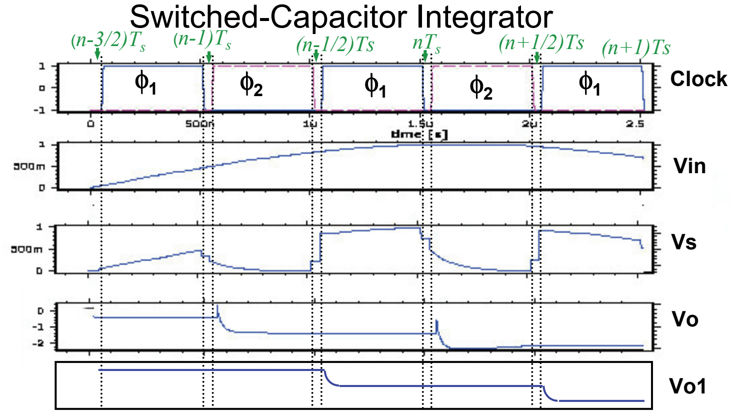
* Note: $z = e^{j\omega T_s} = cos(\omega T_s) + j\, sin(\omega T_s)$



# Switched-Capacitor Integrator Output Sampled on $\phi$1



## Switched-Capacitor Integrator




$\Phi_1 \rightarrow$ $Q_s[(n-1)T_s] = C_s V_i[(n-1)T_s]$ , $Q_I[(n-1)T_s] = Q_I[(n-3/2)T_s]$

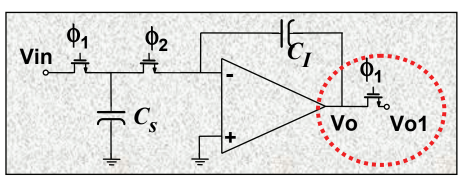
$\Phi_2 \rightarrow$ $Q_s[(n-1/2)T_s] = 0$ , $Q_I[(n-1/2)T_s] = Q_I[(n-1)T_s] + Q_s[(n-1)T_s]$

$\Phi_1 \rightarrow$ $Q_s[nT_s] = C_s V_i[nT_s]$ , $Q_I[nT_s] = Q_I[(n-1)T_s] + Q_s[(n-1)T_s]$

Since $V_{o1} = -Q_I/C_I$ & $V_i = Q_s/C_s$ $\rightarrow C_I V_{o1}(nT_s) = C_I V_{o1}[(n-1)T_s] - C_s V_i[(n-1)T_s]$



## Switched-Capacitor Integrator Output Sampled on $\phi_1$




$$C_I V_o(nT_s) = C_I V_o[(n-1)T_s] - C_s V_{in}[(n-1)T_s]$$

$$V_o(nT_s) = V_o[(n-1)T_s] - \frac{C_s}{C_I} V_{in}[(n-1)T_s]$$

$$V_o(Z) = Z^{-1} V_o(Z) - Z^{-1}\frac{C_s}{C_I} V_{in}(Z)$$

$$\frac{V_o}{V_{in}}(Z) = -\frac{C_s}{C_I} \times \frac{Z^{-1}}{1-Z^{-1}}$$

DDI (Direct-Transform Discrete Integrator)