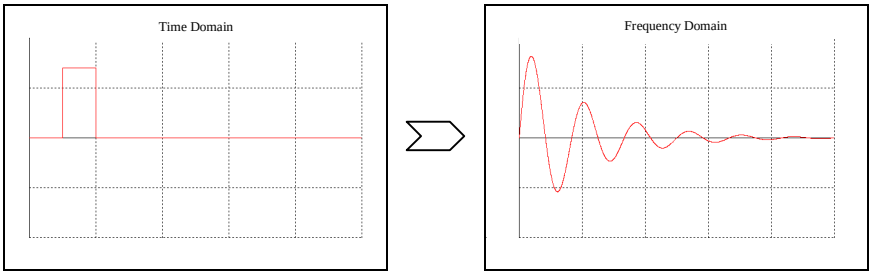
David Goldman

EE 515 – CMOS Mixed-Signal IC Design

Problem 30.1

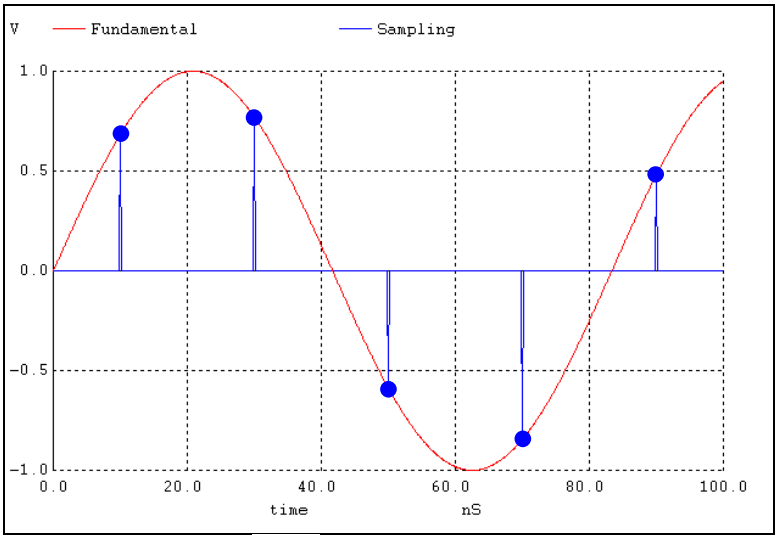
Question: Qualitatively, using figures, show how impulse sampling a sinewave can result in an alias of the sampled sinewave at a different frequency.

Answer: Impulse sampling is composed of a series of finite pulses, in other words, time-limited. Since these pulses are time-limited, they cannot be simultaneously bandwidth-limited. So, all real pulses have infinite spectra. This is shown below with a qualitative plot of the spectrum of a rectangular pulse.

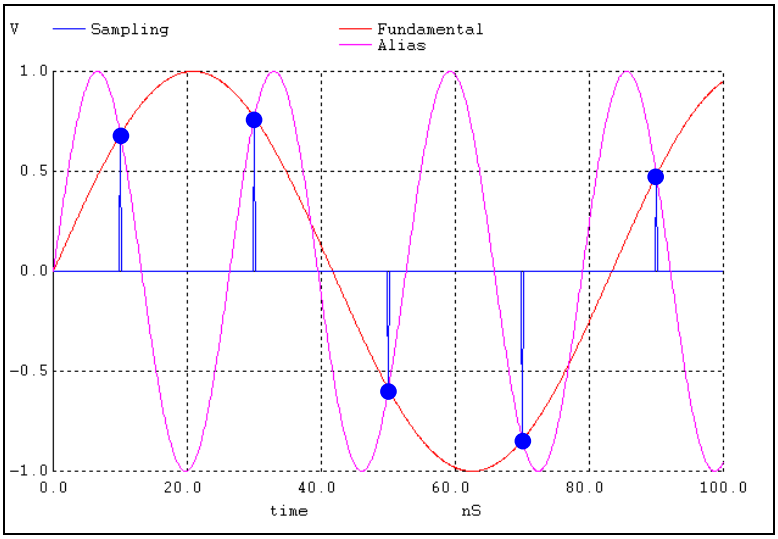


As a result, the spectra of real impulses have infinite "tails" that always extend beyond the Nyquist frequency and overlap other spectra with their own infinite "tails." An alias is created (unless properly filtered) at regions in the frequency domain that contain spectral overlap. The center of this overlap is referred to as the folding frequency, since the spectral content is folded back at these frequencies. This intuitive understanding of aliasing in the frequency domain can be reinforced with an example in the time domain.

Consider the time domain of a fundamental signal and the associated sample pulses shown below (in this example, the signal is 12 MHz and the sample rate is 50 MHz.)



The sampling points that coincide with the fundamental signal, of course, fit the fundamental frequency. But, these points also fit an infinite number of other frequencies. These spurious sinewaves are aliases of the fundamental frequency. One example alias signal that fits these sample pulses is shown below.



In this particular example, the alias signal is 38 MHz. Converting this plot to the frequency domain would produce spectral spikes on the fundamental frequency and on the alias frequency.

Gexin
Ghuang@uidaho.edu

Problem 30.2:
What does linear phase indicate?

Solution for problem 30.2:
For a sine wave signal at frequency f (unit: HZ), its phase delay $\theta(f)$ (unit: radian) through a filter is related to its time delay t_d (unit: second) by

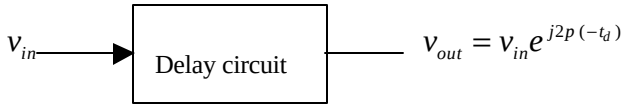
$$\theta(f) = t_d \cdot f \cdot 2\pi$$

If the filter has linear phase response, that is, in the phase response plot, $\frac{\theta(f)}{f}$ is constant. Thus the time delay $t_d = \frac{\theta(f)}{f \cdot 2\pi}$ will be constant independent of signal frequency. Thus linear phase indicates constant time delay.
In many applications, it is desirable that a filter has a linear-phase characteristic. In particular it is important for speech and music processing and for data transmission wherein nonlinear phase would give unacceptable signal distortion.

Brandon Roth
brandonroth@micron.com

30.3 What does multiplying a signal by $e^{j2\pi f(-t_d)}$ indicate? How does the magnitude of the resulting signal change? How does the phase change?

Multiplying a signal by $e^{j2\pi f(-t_d)}$ shifts the signal in time. The multiplied result is delayed by the value of t_d.



The phase shift is related by

$$phaseshift = \theta = \frac{t_d}{T} \bullet 360^o = 2\pi f t_d$$

Where t_d is the delay, and T is the period of the input signal. Dividing t_d by T gives the percentage of delay in one cycle, and then multiplying by 360 degrees gives the phase shift.

Multiplying by $e^{j2\pi f(-t_d)}$ does not change the magnitude of the resulting signal.

From euler's theorem:

$$e^{\pm j\theta} = \cos\theta \pm j\sin\theta$$

let $\theta = j2\pi f(t_d)$ then

$$v_{out} = v_{in}(\cos\theta - j\sin\theta)$$

Taking the magnitude of v_{out}:

$$|v_{out}| = \sqrt{v^2{}_{in}(\cos^2 - (-1)\sin^2)} = \sqrt{v^2{}_{in}(\cos^2\theta + \sin^2\theta)}$$

From trigonometry

$$\cos^2\theta + \sin^2\theta = 1$$

then

$$|v_{out}| = |v_{in}|$$

Thus the magnitude of the input equals the magnitude of the output or in other words multiplying by $e^{j2\pi f(-t_d)}$ does not change the magnitude of the signal.

Multiplying by $e^{j2\pi f(-t_d)}$ does change the phase. The change in phase associated with multiplying by $e^{j2\pi f(-t_d)}$ is found by using Euler's theorem and then finding the phase angle of the resulting vector.

From above

$$e^{\pm jq} = \cos q \pm j \sin q$$

phase angle = $f = \tan^{-1} \dfrac{\{\text{Im}\}}{\{\text{Re}\}}$ or $f = \tan^{-1} \dfrac{\sin q}{\cos q}$ where $\theta = j2\pi f(-t_d)$.

then

$$f = \tan^{-1} \tan q = q$$

The phase resulting from multiply by $e^{j2\pi f(-t_d)}$ is simply the delay $2\pi f(-t_d)$.

EE515: CMOS Mixed-Signal IC Design

Problem 30.4
Jim Slupe

30.4 Show, in the time-domain, the input, output of the transmission line, and output of the comb filter in Fig. 30.11 if the input signal is a sine wave with a peak amplitude of 1 V and a frequency of 100 MHz. Show the two 500 resistors average the input signal and the output signal of the delay line (transmission line).

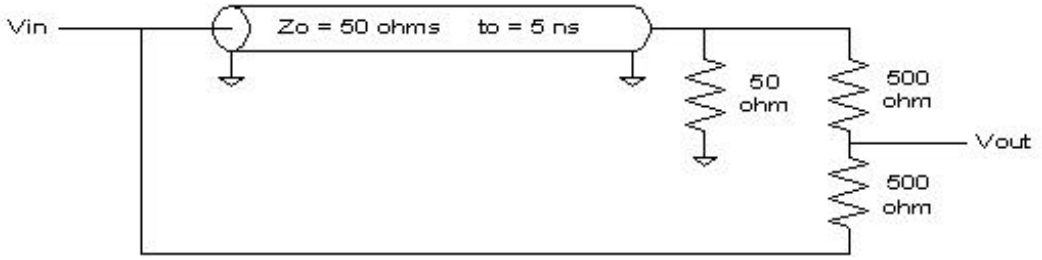


Figure 1

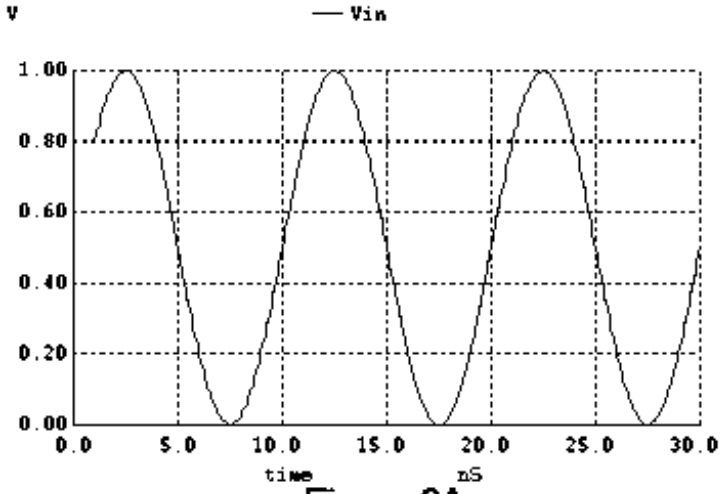


Figure 2A

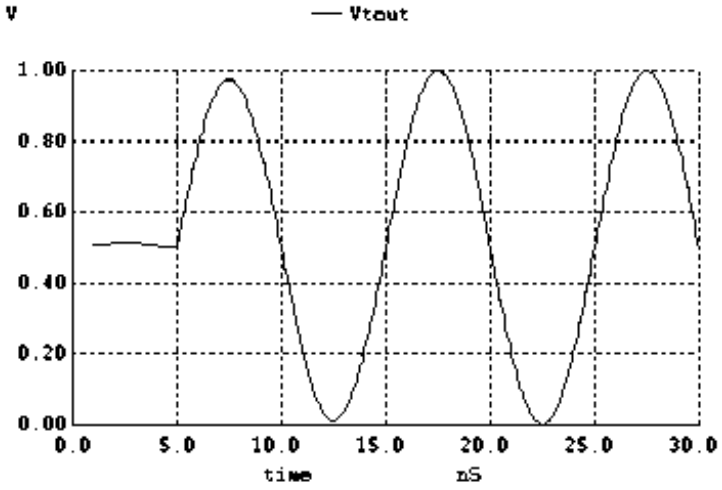


Figure 2B

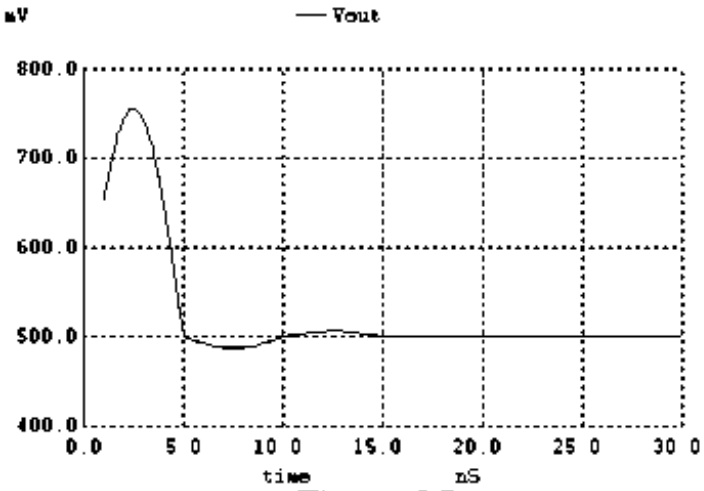


Figure 2C

SPICE listing

```
* Modified Figure 30.12 CMOS: Mixed-Signal Circuit Design *

*WinSPICE command scripts
*#destroy all
*#run
*#plot Vin Vtout Vout

Vin Vin 0 DC 0 SIN(0.5 0.5 100MEG 0 0)

Rtout Vtout 0     50
Rt1   Vtout Vout  500
Rt2   Vin   Vout  500
T1    Vin   0     Vtout 0     ZO=50 TD=5n
.tran 0.1NS 30NS 1NS
.end
```

Figure 1 is a reproduction of the comb filter in Fig. 30.11. The SPICE listing shown above was used to generate the plots shown in figures 2A through 2C.

Figure 2A (Vin) is a 100 MHz sine wave driven into the comb filter.

Figure 2B, the output of the delay line, can be seen as a signal 180 out-of-phase from the input signal as one would expect. The length of the transmission line is 1/2 a wave length of a 100 MHz signal.

Figure 2C, the output of the comb filter, is shown as the decaying voltage that settles to the average of the two signals Vin and Vtout. This is as expected since the delay line creates a situation where two sine waves centered around 0.5 Volts, of equal magnitude, 180 out-of-phase, are summed together at the junction of the two 500 ohm resistors.

If one were to change the sine wave to any of the frequencies (100, 300, 500, etc.) where the frequency domain plot of Figure 30.12 goes to zero, one would find a similar decaying output centered about 0.5 Volts. At each of these frequencies, the length of the transmission line is such that it creates a phase shifted image of the input signal that is a multiple of 180 that, when combined by the 500 ohm resistors, leaves only the 0.5 Volt DC component.

One other point of interest: In the SPICE output of Figure 2C, one can see the behavior before steady state is reached. Vout between 0 and 5 nsec is composed entirely of the DC component and that part of Vin that does not

pass through the transmission line. Shown below in Figure 3 is the value of Vout at 2.5 nsec.

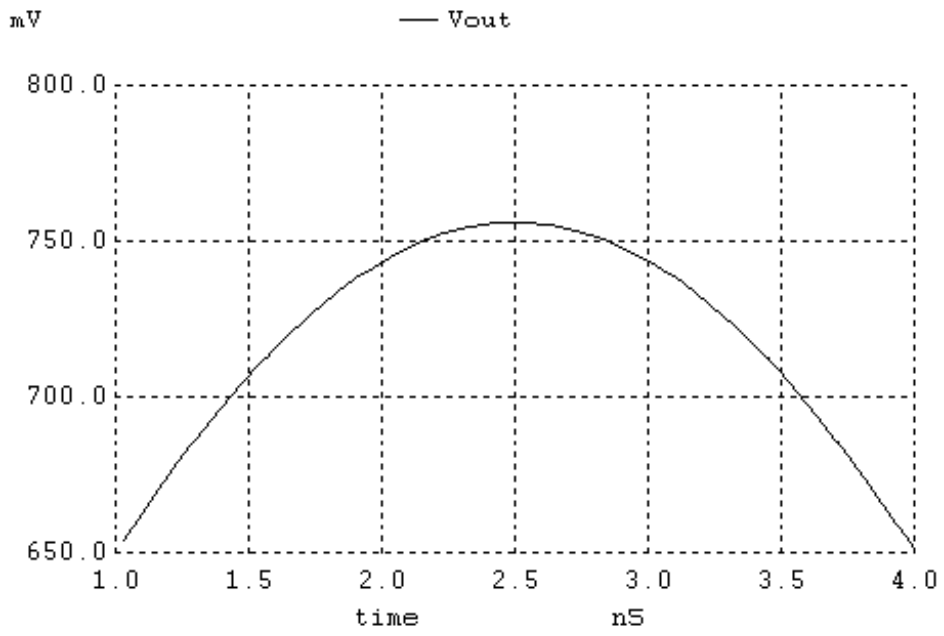


Figure 3

At the instant in time shown in Figure 3 the circuit looks like that shown in Figure 4 below:

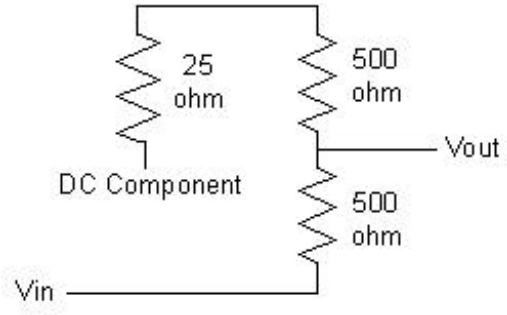
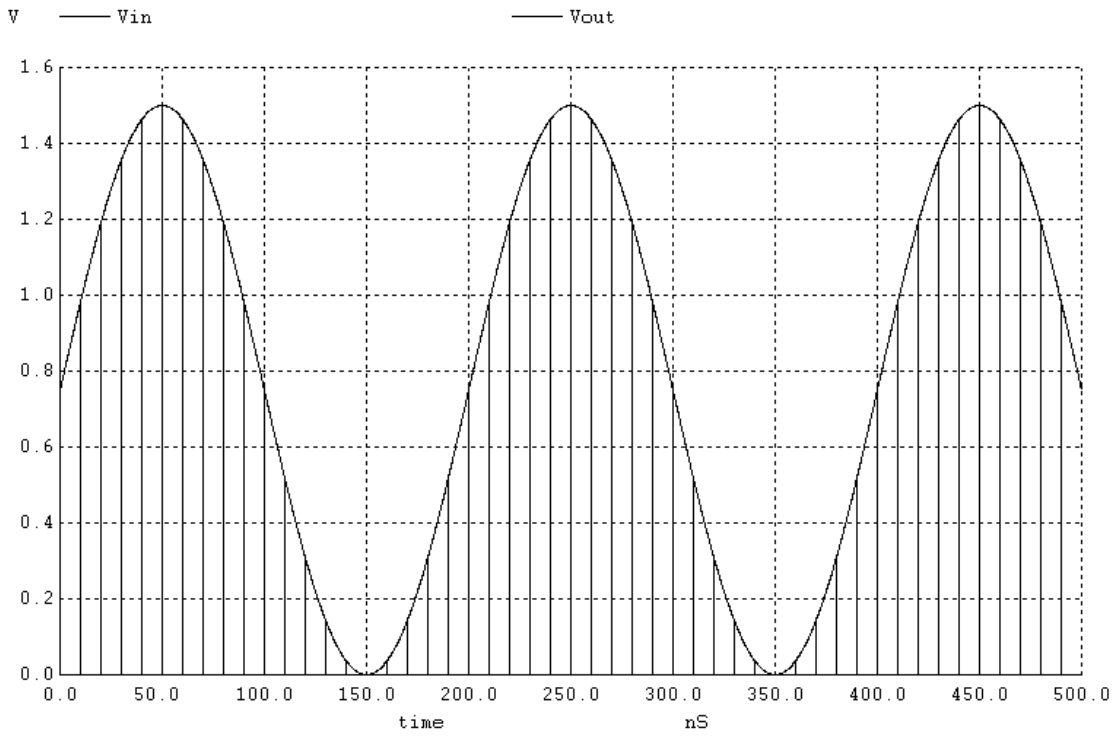


Figure 4

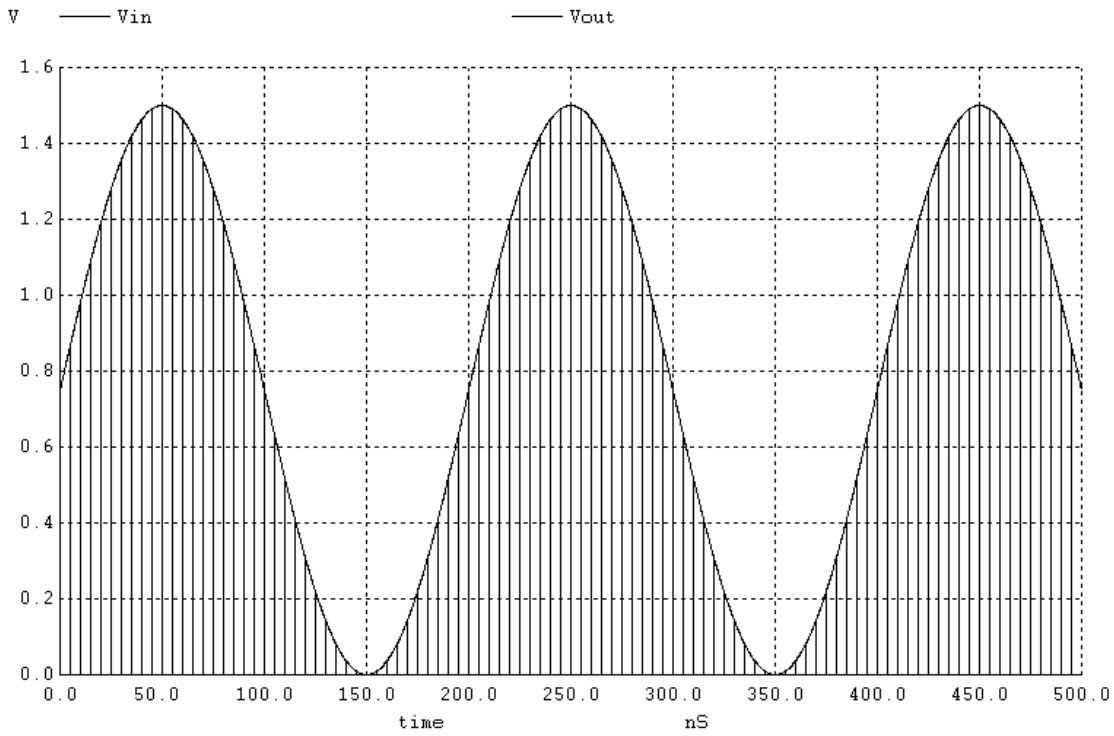
Where the 50 ohm resistor in parallel with the 50 ohm transmission line is replaced by a 25 ohm resistor.

Problem 30.5 Regenerate Fig. 30.19 if the switches are closed for 5ns instead of 100ps.



Solution: Replace the pulse width parameter for Vclock of 100p with 5n.

```
* Problem 30.5 CMOS: Mixed-Signal Circuit Design *

.tran .1n 500n 0 .1n UIC

*WinSPICE command scripts
*#destroy all
*#run
**#plot Clock
*#plot Vin Vout

Vin    Vin   0 DC 0  Sin 0.75 0.75 5MEG
Vclock Clock 0 DC 0  Pulse 0 1.5 0 0 0  5n 10n
Vtrip  Vtrip 0 DC .75

Ebufin      Vinb  0     Vin   Vinb  100MEG
S1    Vinb        Vins  CLOCK       VTRIP switmod
Rout  Vins  0     10k
Ebufout     Vout  0     Vins  Vout  100MEG
.model switmod SW

.end
```



This figure appears to have twice the frequency of the previous version. However, close Inspection of the plot reveals that the pulse in the previous version rises and falls on the 10ns period, whereas the second version rises every 10 ns and falls 5 ns later.

QED

EE515: CMOS Mixed-Signal IC Design
Brian Bergeson
Problem 30.6
bbergeso@poci.amis.com

Problem 30.6:

What sets the minimum resolution of an FFT in a SPICE spectral analysis?

Answer:

An FFT is calculated in SPICE using the "spec" (spectral analysis) command. The minimum resolution allowed when using the spec command is set by the transient simulation time.

On page 16 of CMOS: Mixed-Signal Circuit Design, equation 30.6 defines the minimum resolution for the SPICE "spec" command as:

FFT_resolution >= 1 / simulation_time

For example, if a circuit is simulated for 500ns, then the minimum resolution allowed for the SPICE spec command would be 1 / 500E-9 = 2E6 or 2MHz.

30.7

Explain why the sinewave in Fig. 30.19 is "double sampled".

The sinewave in Fig. 30.19 is a sampled sinewave produced by using discrete time steps of 100 ps in the SPICE simulation of the impulse sampler. Evidence of this sampling is shown by the small stair-step pattern of the input sinewave. The sinewave is again sampled at 100 MHz to produce the output of the impulse sampler.

EE515: CMOS Mixed-Signal IC Design

Richard Friel
Problem 30.8
Rich_Friel@AMIS.COM

Problem #30.8 Explain why ***z*** is used in signal processing. What does multiplying a digital signal by z^{-1} do?

Why is ***z*** is used in signal processing?

The counterpart of the Fourier transform for continuous time signal analysis is the ***z***-transform, for discrete-time signal analysis. The ***z***-transform simplifies linear time-invariant discrete-time (LTID) system analysis and offers insight into system behaviors in terms of frequency-domain concepts such as frequency response and filtering. The Fourier transform converts integro-differential equations into algebraic equations. In a similar manner, the ***z***-transform changes difference equations into algebraic equations, simplifying the analysis of discrete-time systems. Similar to the difference equation, (Eq. 30.10), being transformed into (Eq. 30.14) using the ***z***-transform.

The ***z***-transform is a tool by which a signal ***x[k]*** is expressed as a sum of it's spectral components of the form z^k. If $H[z]z^k$ is the system response to input z^k, we add the system response to all exponential components of ***x[k]*** to obtain the system response, ***y[k],*** using convolution in time domain or multiplication and the inverse ***z***-transform in the frequency domain.[1]

What does multiplying a digital signal by z^{-1} do?

The Fourier transform of the continuous signal ***x(t)*** that is time sampled to create ***y(t)*** in (Eq. 30.10), results in the frequency domain response ***Y(f).*** This shows the individual frequency components that make up the sampled time domain signal ***y(t)*** and the exponential terms that are summed, which correspond to each sample period. The exponential terms represent a phase shift in frequency domain or time delay in the temporal domain. ***z*** is defined to be a phase shift or delay that corresponds to the sampling period T_s as shown in (Eq. 30.12).

When ***x(k)*** in (Eq. 30.14) is multiplied by z^{-1}, this delays every term of the summation, in the output, ***Y[z],*** by one sampling period, T_s, which results in the following equation,

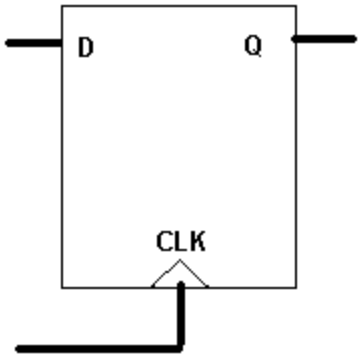
$$Y[z] = \sum_{k=0}^{\infty} x(k) z^{-(k+1)}$$

A good example is given in Example 30.6 by studying the effect of z^{-1} on the Ideal impulse sampler, which delays the output of the ideal sampler by one sampling period, T_s.

[1] B.P. Lathi, "Linear Signals and Systems," Berkeley/Cambridge Press, pp.377, 1992

Problem 30.9 Sketch the implementation of a circuit that will multiply a digital signal by z^{-1}.

Solution: Multiplying a signal by z^{-1} is the same as delaying the signal. A rudimentary delay can be created by a D flipflop.



Bob Moehrke EE515 (bmoehrke@amis.com)

Question 30.10

Sketch the time-domain representation of the five signals shown in Fig. 30.29 on different plots. Regenerate Fig. 30.29 if the input signal is a 1 V peak sinewave at 5 MHz and zero offset. Explain the resulting plot.

Figure 30.29 shows a frequency plot of the output of a S/H of a 0.75 V (peak), 3 MHz sinewave centered at 0.75 V (-2.5 dB) and sampled (and held) at 100 MHz. Due to the 0.75 V offset there is a signal at DC (0 Hz); and due to aliasing there is signal content not only at 3 MHz, but also $f_S \pm 3$ MHz (97 and 103 MHz) and $2F_S - 3$ MHz (197 MHz). Assuming the effects of double sampling are negligible, the magnitudes of each signal content can be determined by taking into account the magnitude of the input waveform and the attenuation of the $\text{sinc}(\pi f/f_S)$ function resulting from the sample and hold operation:

M(0) = 0.75 * sinc(π0/100) = 0.75 = -2.499 dB
M(3) = 0.75 * sinc(π3/100) = 0.7489 = -2.512 dB
M(97) = 0.75 * sinc(π97/100) = 0.02316 = -32.70 dB
M(103) = 0.75 * sinc(π103/100) = 0.02181 = -33.23 dB
M(197) = 0.75 * sinc(π197/100) = 0.01140 = -38.86 dB

These values can be seen (respectively) in the following plots, Figures 1-5:

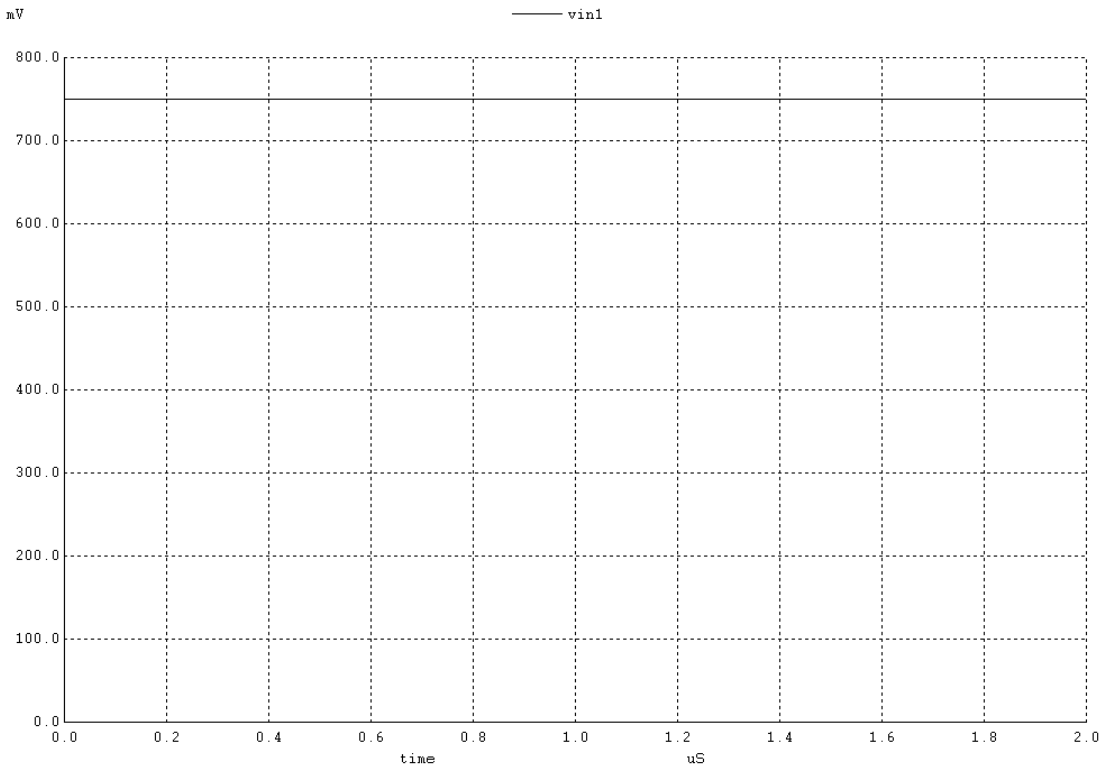


Figure 1

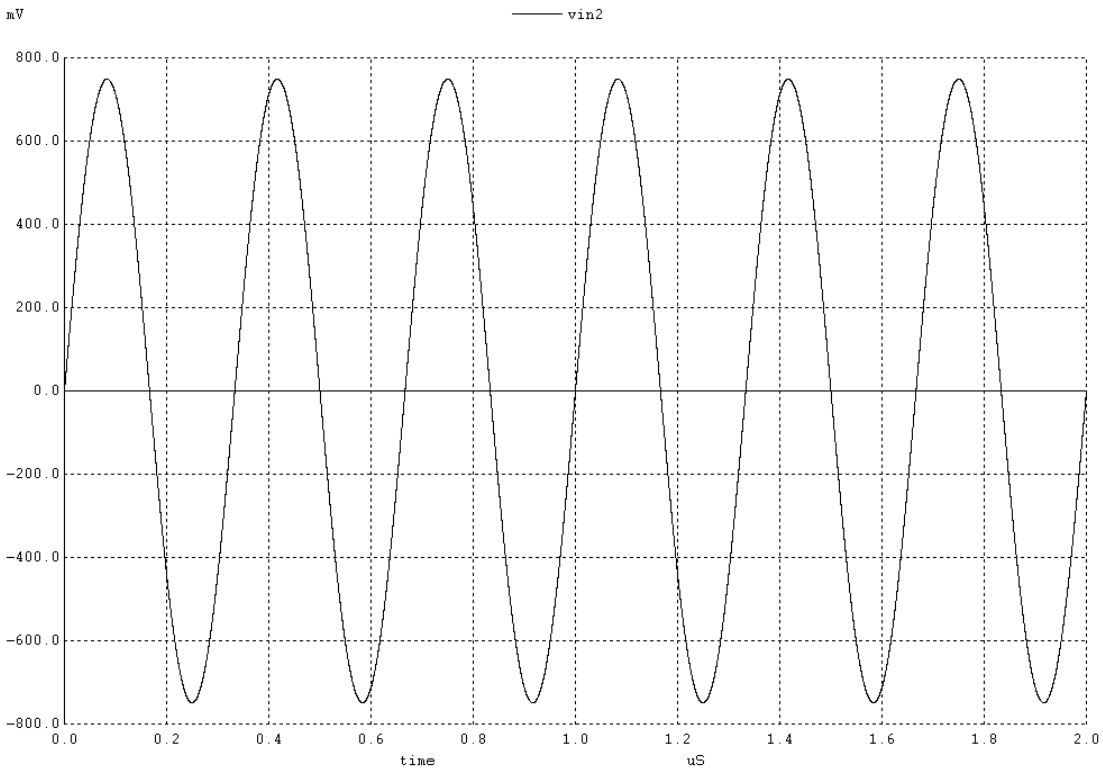


Figure 2

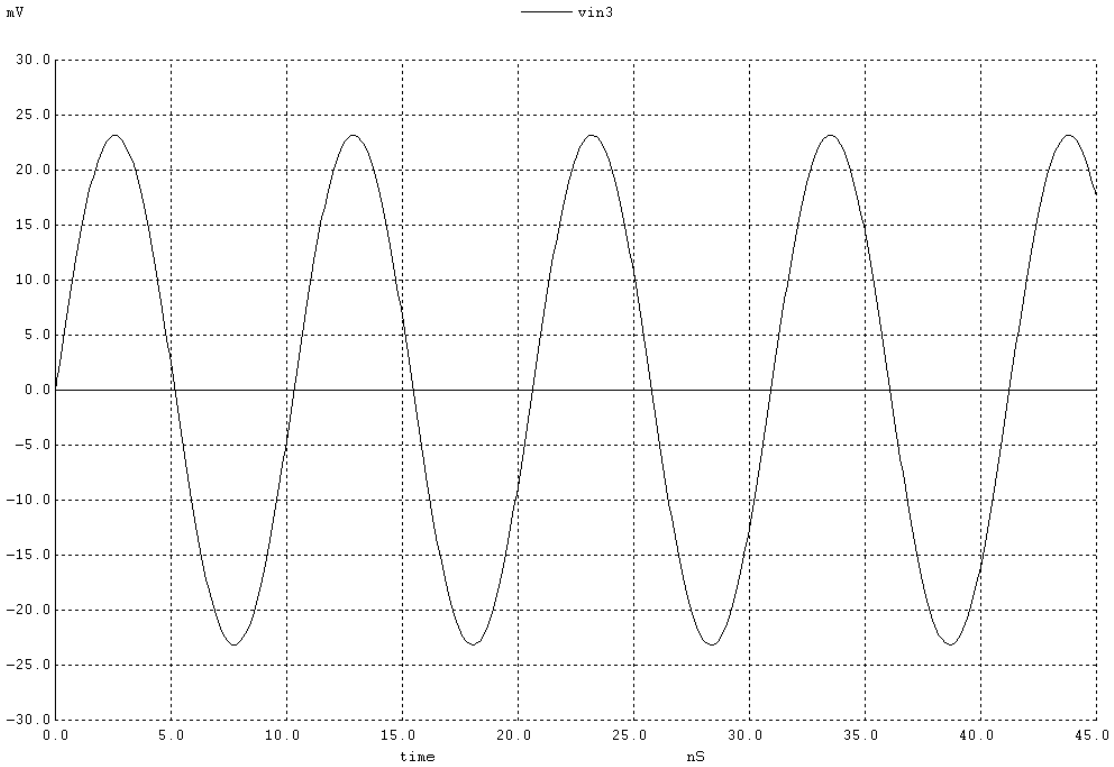


Figure 3

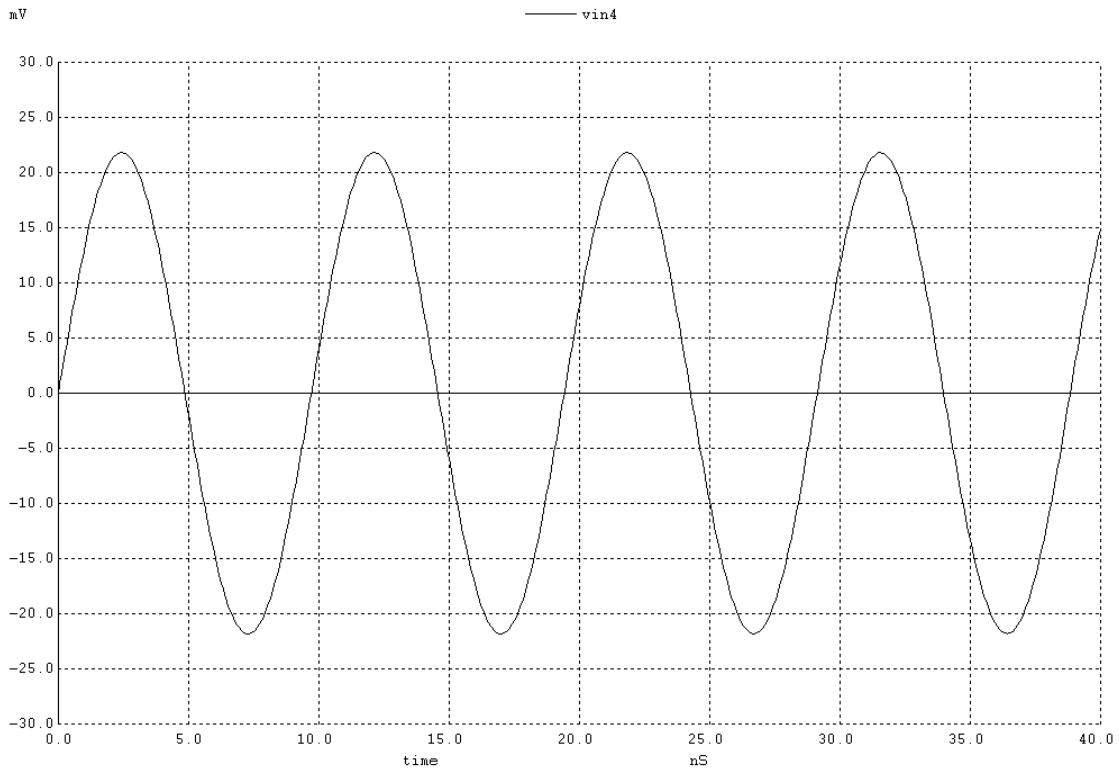


Figure 4

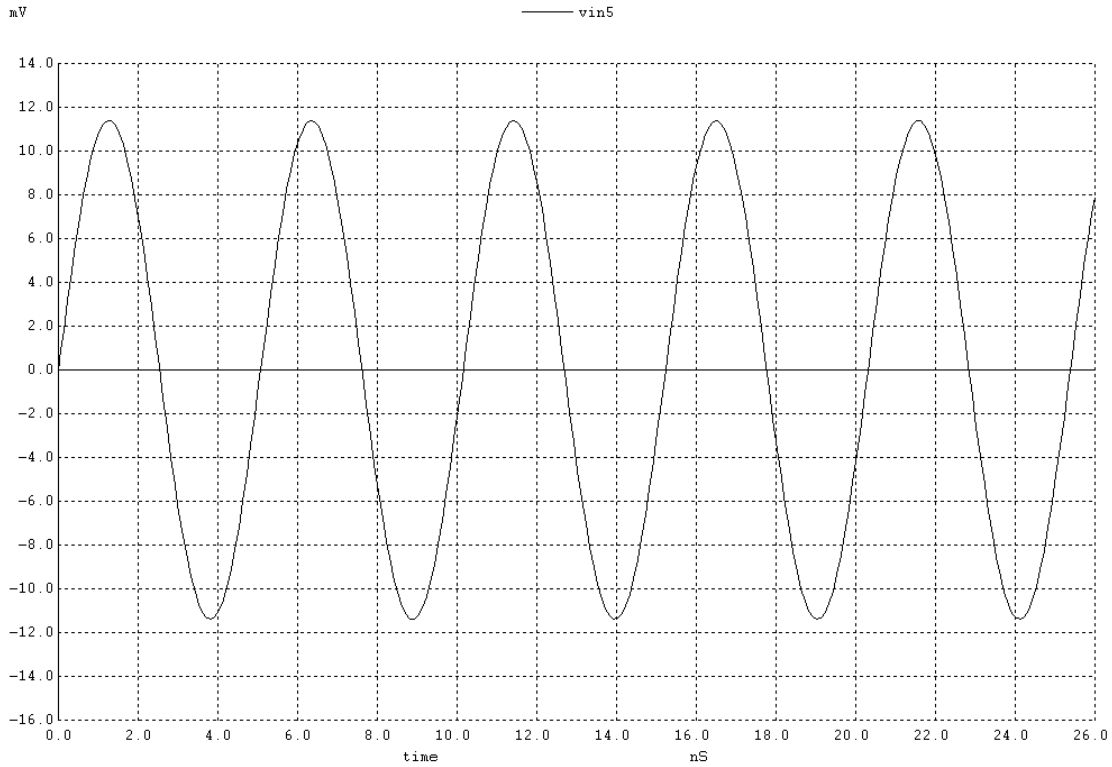


Figure 5

There is no offset in the input waveform used to regenerate Fig. 30.29, so the signal at DC (0 Hz) is not present. As before, due to aliasing there is signal content not only at 5 MHz, but also $f_S \pm 5$ MHz (95 and 105 MHz) and $2F_S - 5$ MHz (195 MHz). Again, the magnitudes of each signal content can be determined by taking into account the magnitude of the input waveform and the attenuation of the $\text{sinc}(\pi f/f_S)$ function resulting from the sample and hold operation:

M(5) = 1 * sinc(π5/100) = 0.9959 = -0.03575 dB
M(95) = 1 * sinc(π95/100) = 0.05242 = -25.61 dB
M(105) = 1 * sinc(π105/100) = 0.04742 = -26.48 dB
M(195) = 1 * sinc(π195/100) = 0.02554 = -31.86 dB

After looking at the plot in Figure 6, it can be seen that the magnitude and location of each signal matches the calculated values.

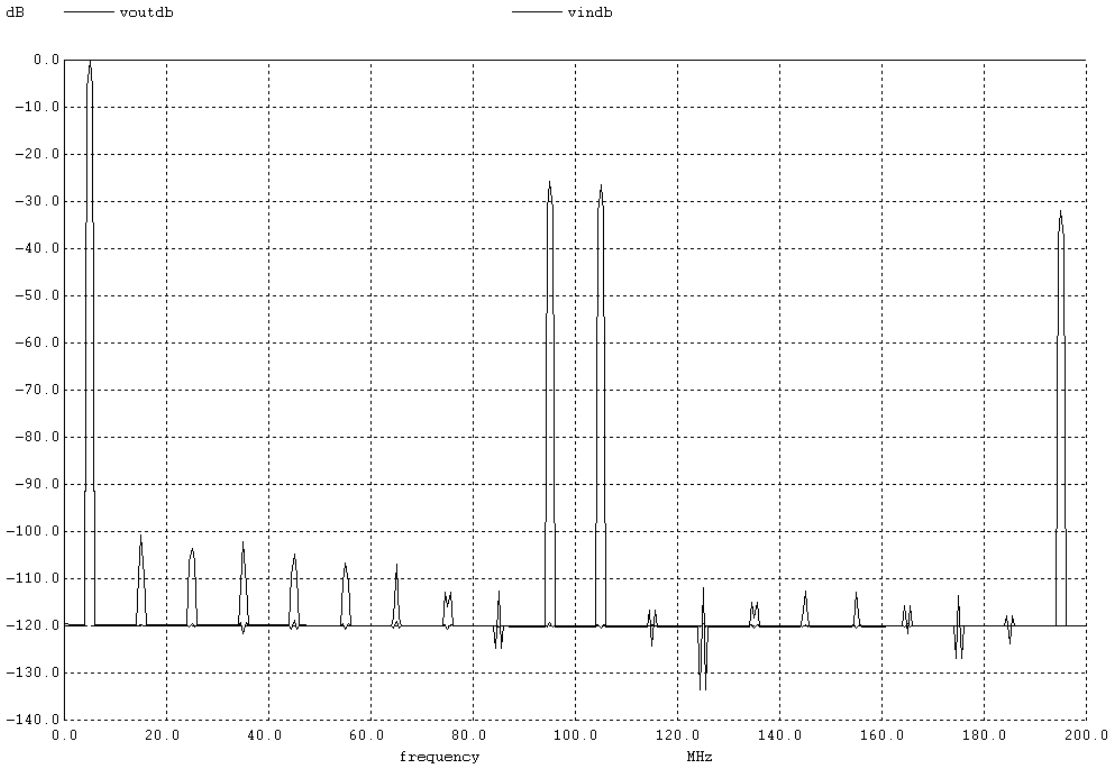


Figure 6

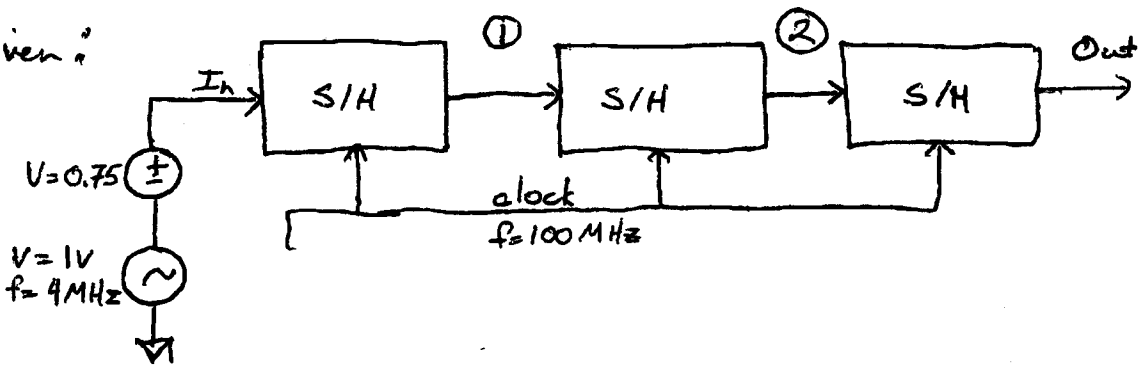
SPICE NETLIST for regenerating Figure 30.29:

```
* Regenerated Figure 30.29 CMOS: Mixed-Signal Circuit Design *

*Since using ideal components and no MOSFETs increase step size to 1ns
.tran .1n 2000n 0 .1n UIC
.options reltol=1u
.save Vout Vin

*WinSPICE command scripts
*#destroy all
*#run
**#plot Clock
```

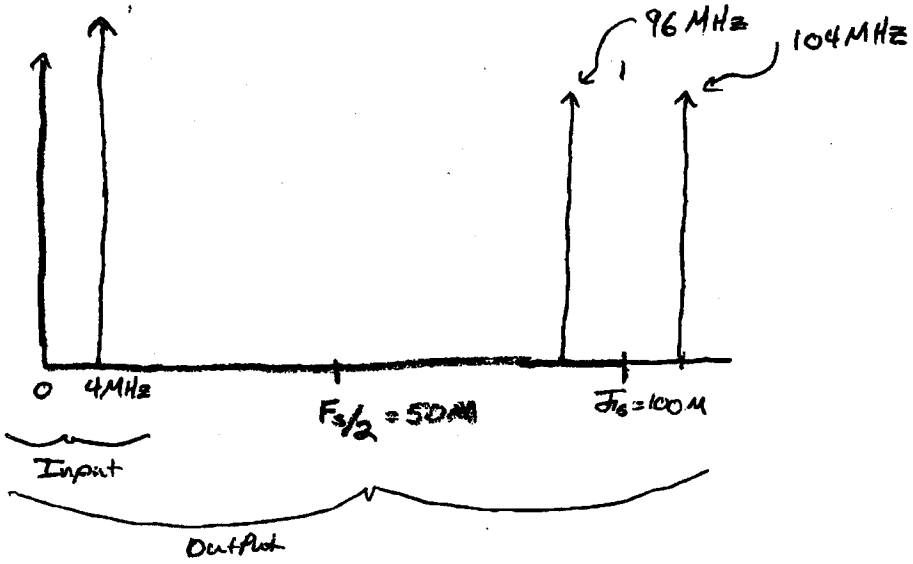
```
**#plot Vin Vout
*#linearize Vout Vin
*#spec 0 200MEG 500k Vout Vin
* Set noise floor at -120 dB (1uV)
*#let voutdb=db(Vout+1e-6)
*#let vindb=db(Vin+1e-6)
*#plot voutdb vindb

Vin    Vin   0 DC 0  Sin 0 1 5MEG
Vclock Clock 0 DC 0  Pulse 0 1.5 0 0 0 4.9n 10n
Vtrip  Vtrip 0 DC .75
VDD    VDD   0 DC 1.5

Ein   Vinbuf      0     Vin   Vinbuf      100MEG

S1    VDD   CLKB  VTRIP CLOCK switmod
S2    0     CLKB        CLOCK       VTRIP switmod

S3    Vinbuf  VinS      CLOCK VTRIP switmod
Cs1   VinS  0     100p

S4    VinS  Vout1       CLKB  VTRIP switmod
Cout1       Vout1       0     10f

Eout  Vout  0     Vout1       Vout  100MEG

.model switmod SW (RON=10m)

.end
```

30.11 [redacted] | Jeremy Rice

Given:



Find: Spectrum at the input and output nodes



Explanation:

The input spectrum will be the 4MHz spike, and DC spike as shown.
The output spectrum will also include the spurs at 96 and 104 MHz as a result of the sampling. These extra tones will be present @ ±4MHz of multiples of the clock frequency, and will be weighted by the sinc function.

Having three ideal S/H circuits in series does not add any additional Amplitude reduction or introduce any additional tones, since all 3 will sample the same signal.

Problem 30.12 Using the models developed in this chapter to design a SPICE model for S/H of Fig. 30.31. Use the model to generate Fig. 30.29.

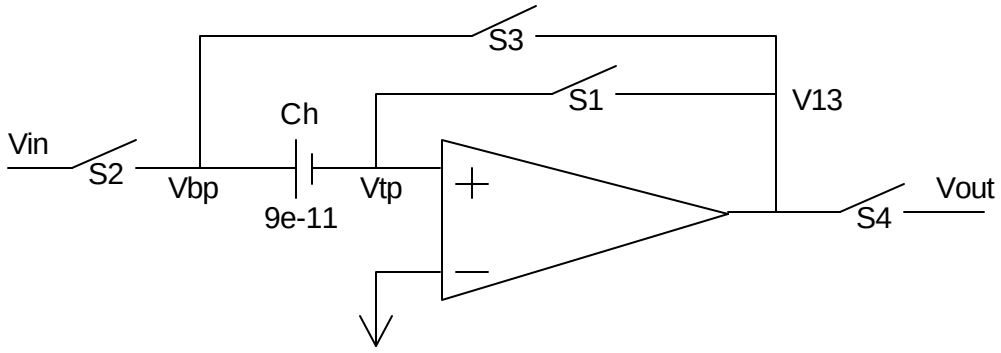


Fig. 30.12a Single-ended S/H circuit used to re-generate Fig.30.29.

Fig. 30.12b is the SPICE output of Fig. 30.12a. The SPICE-model file is called prob_30_12a.cir. Note that S3 and S4 above, ideally, switch at the same time. In the SPICE file, sampling instant (off time of the clock) is very narrow to keep the op-amp from drifting.

Initially, S1 and S2 are closed while S3 and S4 are open; shortly after that, S1 is open, then S2 is open. Holding operation takes place when S3 and S4 are closed while S1 and S2 are open. Sampling operation takes place from the time that S1 starts to open until the time that S3 and S4 start to close.

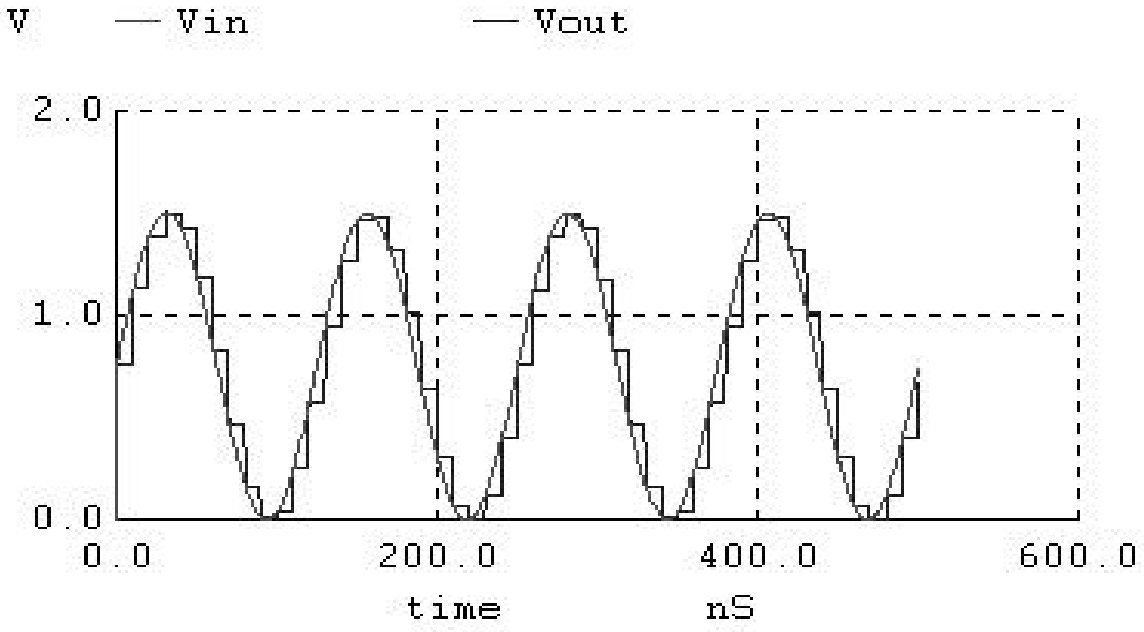


Fig. 30.12b Output of single-ended S/H in Figure 30.31.

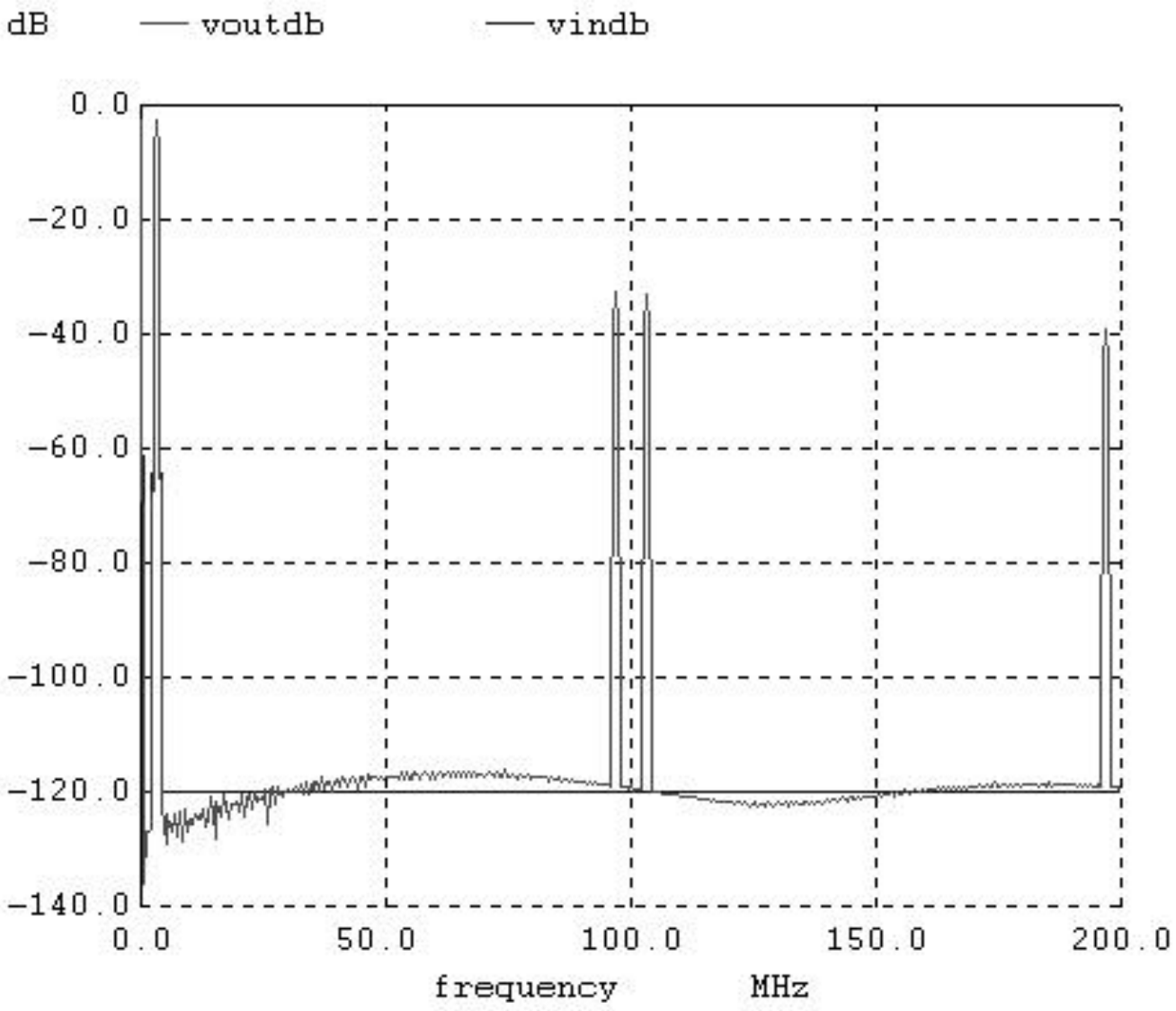


Fig. 30.12c Output of the single-ended S/H after sampling a 3MHZ sinewave at 100 Mhz, re-creating Figure 30.29.

The SPICE file to generate Fig. 30.12c above is called prob_30_12b.cir. I had to set noise floor RELTOL= 1nV to keep the noise down. All outputs amplitudes are maintained the same as those shown in Fig. 30.29 in the text.

Tyler J. Gomm
tjgomm@micron.com

30.13 If V_{REF+} = 1.5 V and V_{REF-} = 0 regenerate Fig. 30.33 using SPICE. (Design a 3-bit ideal DAC model in SPICE.) The y-axis will be voltages in decimal form.

The output voltage of a DAC can be represented using Eq. (30.25) as follows:

$$V_{OUT} = (V_{REF+} - V_{REF-}) \cdot \frac{1}{2^N} \cdot (b_{N-1} 2^{N-1} + b_{N-2} \cdot 2^{N-2} + \ldots + b_1 2^1 + b_0) + V_{REF-}$$

This is implemented in SPICE using a non-linear dependent source (a B source). As stated in the text, for a 3-bit, ideal DAC, the statement that implements this equation looks like:

```
*Non-linear dependent source, B, for generating the DAC output
Bout Vout 0 V=((v(vrefp)-v(vrefm))/8)*(v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)
```

The terms BxL are logic signals that have values of 0 V or 1V. These are created using two ideal switched in SPICE as shown in Figure 1.

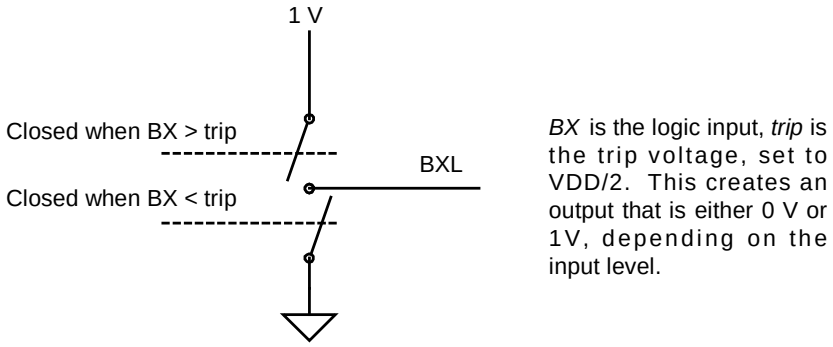


Figure 1 Generating logic levels using two voltage-controlled switches.

The VDD/2 trip voltage can be created as follows:

```
*Generate Logic switching point, or trip, voltage
R1 VDD trip 100MEG
R2 trip 0   100MEG
```

The switch configuration is implemented using the following:

```
.subckt Bitlogic trip BX BXL
Vone    one     0       DC      1
SH      one     BXL     BX      trip    Switmod
SL      0       BXL     trip    BX      Switmod
.model switmod SW
```

This forces the inputs to well-defined logic levels. Each bit in the DAC is instantiated (prior to the statement for the non-linear dependent source) as follows:

```
X2 trip B2 B2L Bitlogic
X1 trip B1 B1L Bitlogic
X0 trip B0 B0L Bitlogic
```

The digital inputs can be created with simple pulse sources. The entire netlist, as well as the WinSpice control statements is shown in Figure 2.

```
* Problem 30.13 CMOS: Mixed-Signal Circuit Design *

.tran 1n 80n 0 1n

*#destroy all
*#run
*#let bin2=b2+4
*#let bin1=b1+2
*#let bin0=b0+0
*#plot bin2 bin1 bin0
*#plot Vout
*#plot (Vout-VREFM)/(VREFP-VREFM)

VDD VDD 0 DC 1.5
VREFP VREFP 0 DC 1.5
VREFM VREFM 0 DC 0.0

VB2 B2 0 DC 0 pulse 1.5 0 0 200p 200p 39.8n   80n
VB1 B1 0 DC 0 pulse 1.5 0 0 200p 200p 19.8n   40n
VB0 B0 0 DC 0 pulse 1.5 0 0 200p 200p 9.8n    20n

X1 VDD VREFP VREFM Vout B2 B1 B0 DAC3bit

*** Start Ideal DAC Subcircuit ******************************

.subckt DAC3bit VDD VREFP VREFM Vout B2 B1 B0

*Generate Logic switching point, or trip, voltage
R1 VDD trip 100MEG
R2 trip 0   100MEG

*Change input logic signals into logic 0s or 1s
X2 trip B2 B2L Bitlogic
X1 trip B1 B1L Bitlogic
X0 trip B0 B0L Bitlogic

*Non-linear dependent source, B, for generating the DAC output
Bout Vout 0 V=((v(vrefp)-v(vrefm))/8)*(v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)

.ends

.subckt Bitlogic trip BX BXL
Vone    one     0       DC      1
SH      one     BXL     BX      trip    Switmod
SL      0       BXL     trip    BX      Switmod
.model switmod SW
.ends

*** END DAC Subcircuit ***************************************
.end
```

Figure 2 Full SPICE netlist.

In order to recreate the same scale as Fig. 30.33, the y-axis has to be normalized to the full DAC range. This makes the 'steps' in the graph equal to 1 LSB.

For the 3-bit ideal DAC, one *normalized* LSB = $1/2^3 = 0.125$.

To prevent a divide-by-zero error when normalizing, the transient analysis may be run without using initial conditions (UIC).

The non-normalized LSB for this particular DAC is:

$$(V_{REF+} - V_{REF-})/ 2^N = (1.5\ V - 0\ V)/8 = 0.1875\ V.$$

Figure 3 shows the DAC inputs and the non-normalized DAC output. Note that the full DAC swing ranges from 0 V to V_{REF+} - 1 LSB, or from 0 V to 1.3125 V, with exactly eight possible output levels.

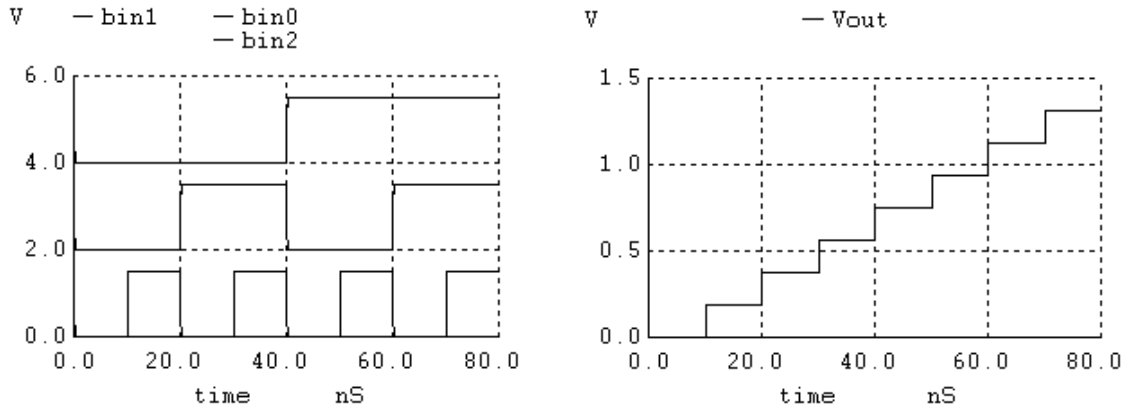


Figure 3 Ideal 3-bit DAC input (left) and non-normalized DAC output(right).

Figure 4 shows the SPICE replication of Figure 30.33 in the text.

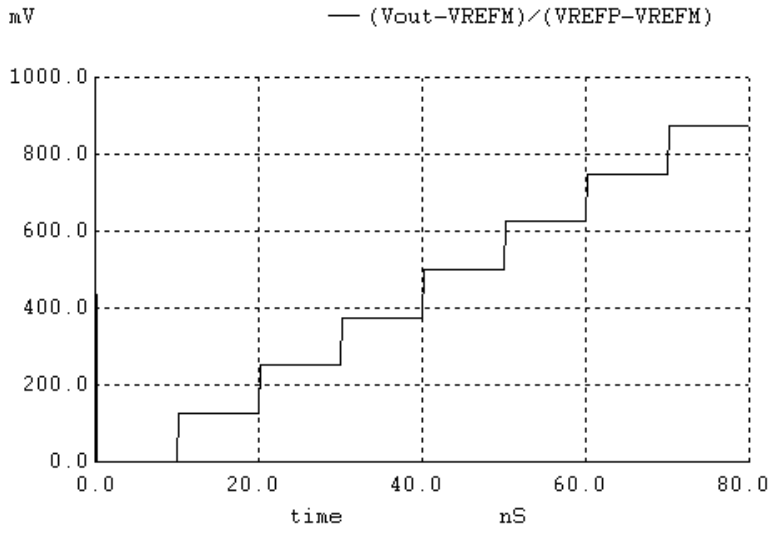


Figure 4 Normalized ideal 3-bit DAC output.

Tyler J. Gomm
tjgomm@micron.com

30.14 If, again, V_{REF+} = 1.5 V and V_{REF-} = 0 sketch Fig. 30.33 for a one-bit DAC. Note that the digital input code will either be a 0 or a 1 and the analog voltage out of the DAC will be either 0 or 1.5 V. Using Eq. (30.23) what is the voltage value of 1 LSB? How does this compare to the value of 1 LSB we get from the sketch? Is Eq. (30.23) valid for a one-bit DAC? Why? The 1-bit DAC will be a ubiquitous component in our noise-shaping modulators in Ch. 32 (see Fig. 32.28).

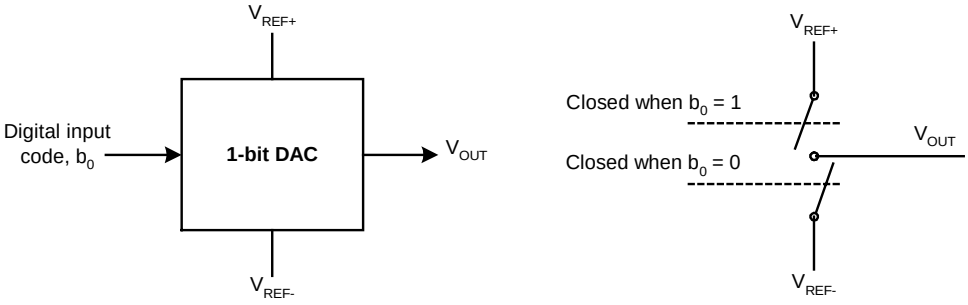


Figure 1 1-bit DAC block diagram and an ideal implementation.

A one-bit DAC (as defined in the problem) can be ideally implemented as two switches connecting the output to V_{REF+} or V_{REF-} (see Fig. 1). There is only one control bit for switching between states, so the simplest method is to just toggle between the rails. This will result in a wide-swing, full-range DAC.

This can be implemented using a comparator that switches to the rails at the midpoint of the full range (see Fig. 2).

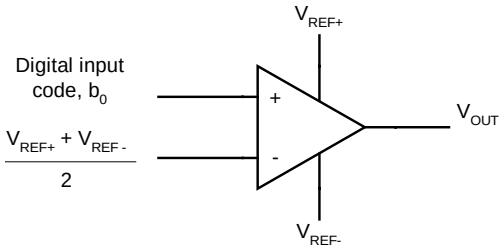


Figure 2 Implementation of a 1-bit DAC using a comparator.

As shown in Fig. 3, this DAC, *by definition*, will simply switch between V_{REF+} and V_{REF-}, based on the single-bit input. This can be compared to Fig. 30.33.

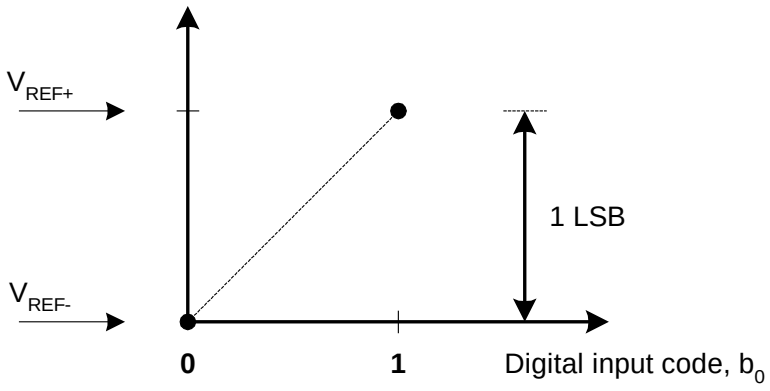


Figure 3 Transfer curve for an ideal 1-bit DAC.

Using Eq. (30.23) the voltage value of 1 LSB is as follows:

$$1\,LSB = \frac{V_{REF+} - V_{REF-}}{2^N} = V_{LSB}$$

$$1\,LSB = \frac{1.5\,V - 0\,V}{2^1} = 0.75\,V$$

The value of 1 LSB obtained from the sketch is 1.5 V, or twice the value obtained from Eq. (30.23). Therefore, Eq. (30.23) does *not* apply to the one-bit DAC. This is due to the way the wide-swing one-bit DAC is defined; that it switches across the *full range* of input voltages.

A one-bit DAC *could* be implemented which would switch between V_{REF+} / 2 and V_{REF-}, forcing Eq. (30.23) to be valid, but the output range would be cut in half!

The correct equation for 1 LSB for the full-range DAC *as defined* is as follows:

$$1\,LSB = \frac{V_{REF+} - V_{REF-}}{2^N - 1} = V_{LSB}$$

For the DAC in this problem this equation yields:

$$1\,LSB = \frac{1.5\,V - 0\,V}{2^1 - 1} = 1.5\,V$$

This definition of a full-range DAC could be reflected back to a general N-bit DAC, resulting in transfer curves similar to that shown in Fig. 4. Although this DAC has the full range of output values, the LSB value is no longer a fraction based on powers of two. (i.e.: 3-bit DAC has a normalized LSB = 1/7, rather than 1/8.)

When working with DACs with more than a few bits of resolution, the loss of 1 LSB across the full range is miniscule, therefore it is mathematically easier to use an LSB that is a fraction based on powers of two.

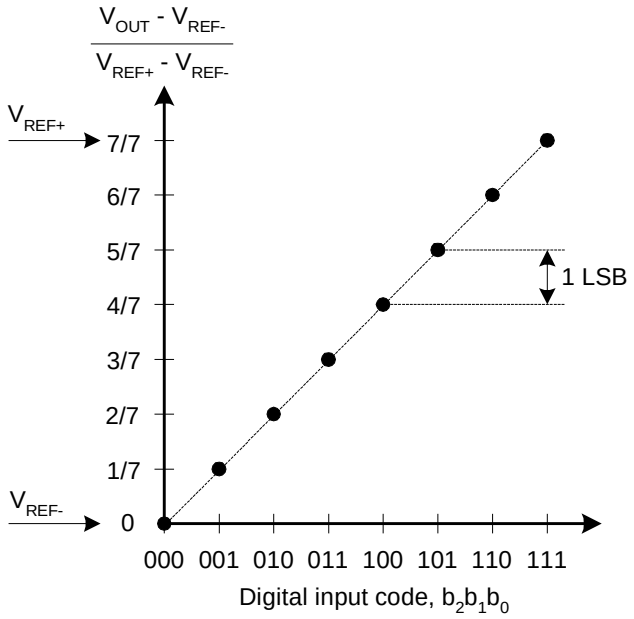


Figure 4 Transfer curve for an ideal *full-range* 3-bit DAC.

As stated in the problem, the 1-bit DAC will be a ubiquitous component in noise-shaping modulators in Ch.32 (see Fig. 32.28). This figure shows the ideal (as well as non-ideal) transfer curve for a 1-bit DAC.

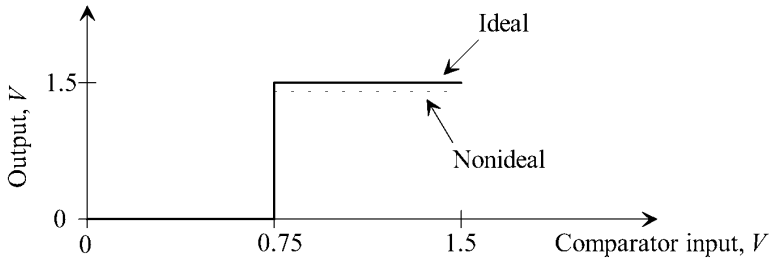


Figure 32.28 Ideal and nonideal transfer curves for the one-bit DAC and comparator cascade.

Problem 30.15 Using SPICE to implement an ideal 4-bit DAC and regenerate Fig. 30.36.

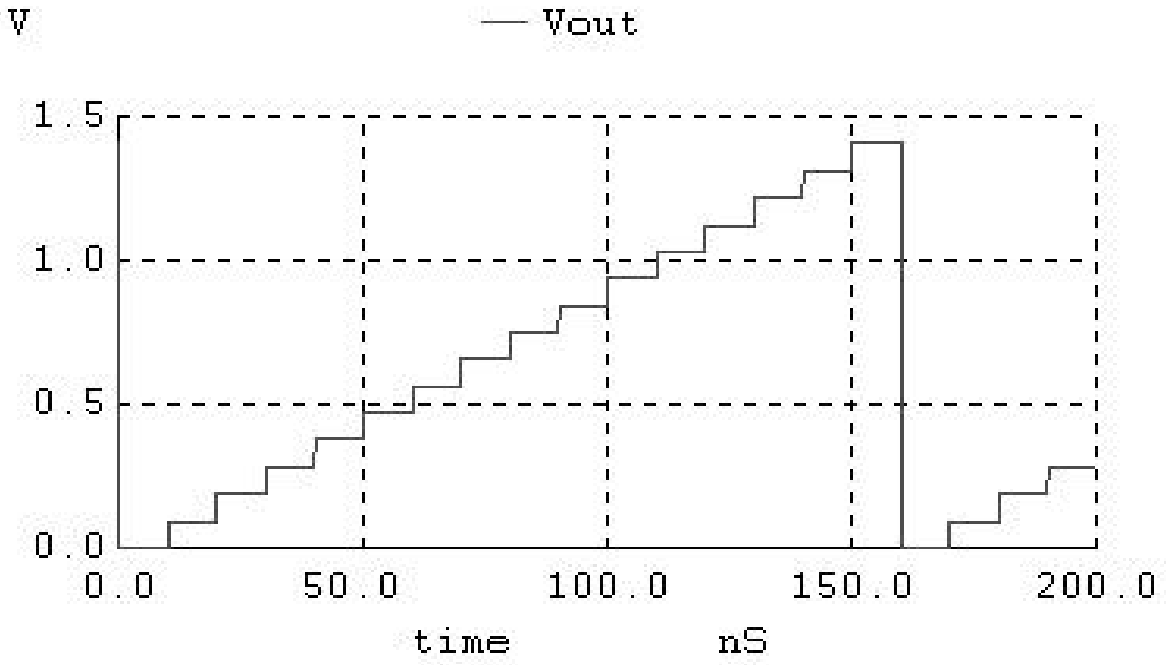


Fig. 30.15a Output of 4-bit DAC

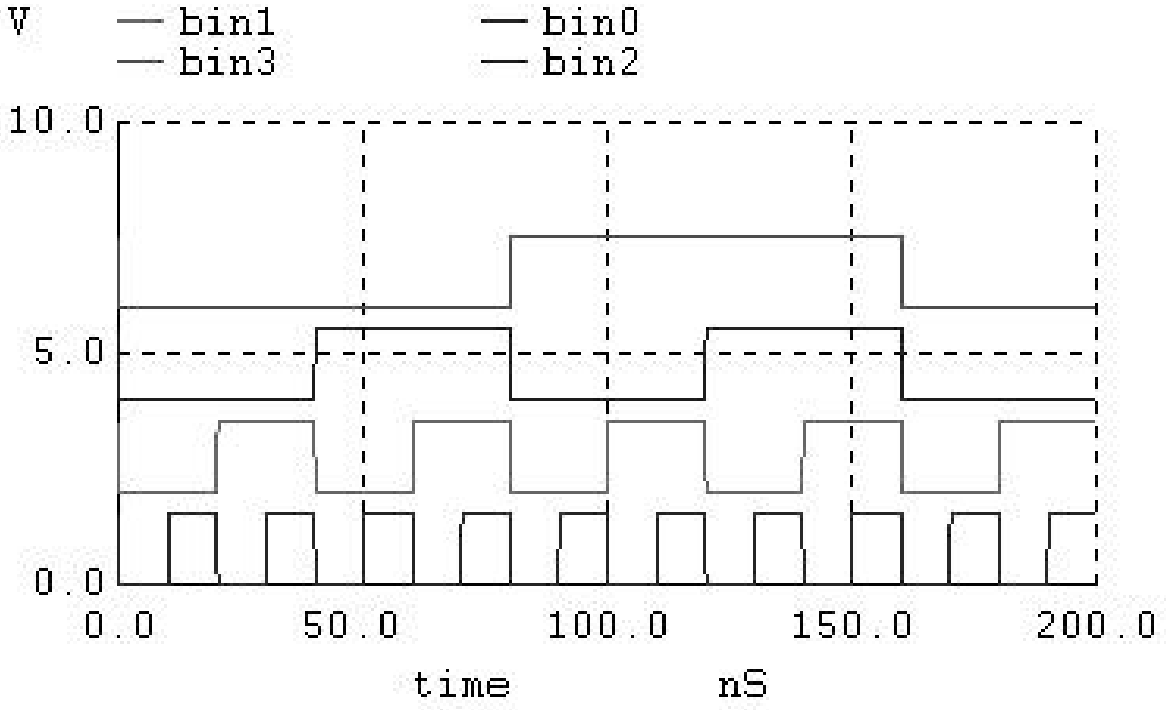


Fig. 30.15b Inputs to 4-bit DAC

The SPICE listing of this 4-bit DAC is called prob_30_15.cir. Each step in Fig. 30.15a is 1 LSB= 1.5V/ 16= 93.75 mV. When input is 1111, the output is 1.5V – 1 LSB= 1.40625V.

```
* For problem 30.15, ideal 4-bit DAC *

.tran 1n 200n 0 1n UIC

*#destroy all
*#run
*#let bin3=b3+6
*#let bin2=b2+4
*#let bin1=b1+2
*#let bin0=b0+0
*#plot bin3 bin2 bin1 bin0
*#plot Vout

VDD VDD 0 DC 1.5
VREFP VREFP 0 DC 1.5
VREFM VREFM 0 DC 0.0

* Width of each bit
VB3 B3 0 DC 0 pulse 1.5 0 0 200p 200p 79.8n 160n
VB2 B2 0 DC 0 pulse 1.5 0 0 200p 200p 39.8n 80n
VB1 B1 0 DC 0 pulse 1.5 0 0 200p 200p 19.8n 40n
VB0 B0 0 DC 0 pulse 1.5 0 0 200p 200p 9.8n 20n


X1 VDD VREFP VREFM Vout B3 B2 B1 B0 DAC4bit

*** Start Ideal DAC Subcircuit ******************************

.subckt DAC4bit VDD VREFP VREFM Vout B3 B2 B1 B0

*Generate Logic switching point, or trip, voltage
R1 VDD trip 100MEG
R2 trip 0 100MEG


*Change input logic signals into logic 0s or 1s
X3 trip B3 B3L  Bitlogic
X2 trip B2 B2L  Bitlogic
X1 trip B1 B1L  Bitlogic
X0 trip B0 B0L  Bitlogic


*Non-linear dependent source, B, for generating the DAC output
Bout Vout 0 V=((v(vrefp)-v(vrefm))/16)*(v(B3L)*8+v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)

.ends

.subckt Bitlogic trip BX BXL
Vone one 0 DC 1
SH one BXL BX trip Switmod
```

```
SL 0 BXL trip BX Switmod
.model switmod SW
.ends

*** END DAC Subcircuit ************************************
.end
```

The tranfer curve of 30.37 shows a shift of ½ LSB to the left due to the way it was implemented in spice. This shifting of the tranfer curve reduces the peak quanization error from 1 LSB to ½ LSB, assuming the maximum input is reduced to ½ LSB below Vref.

To implement this effect in spice, all that needs to be done is to level shift the input by ½ LSB relative to Vref. This essentialy will move the transition points by the ½ LSB shown in Fig 30.37.

Question 30.17

Repeat question 30.15 for an ADC.

30.15 Using SPICE implement an ideal 4-bit DAC and regenerate Fig. 30.36.

Using the SPICE example given in Figure 30.38 for an ideal 8-bit ADC and the SPICE netlist used to generate Figure 30.39, the following SPICE netlist was created for an ideal 4-bit ADC.

SPICE NETLIST for regenerating Figure 30.39:

```
* Regenerated Figure 30.39 CMOS: Mixed-Signal Circuit Design *

.tran 1n 3000n 0 1n UIC

*WinSPICE command scripts
*#destroy all
*#run
**#let bin7=b7+16
**#let bin6=b6+14
**#let bin5=b5+12
**#let bin4=b4+10
*#let bin3=b3+8
*#let bin2=b2+6
*#let bin1=b1+4
*#let bin0=b0+2
**#plot clock bin0 bin1 bin2 bin3 bin4 bin5 bin6 bin7
*#plot clock bin0 bin1 bin2 bin3
*#plot Vin x1:outsh

VDD VDD 0 DC 1.5
VREFP VREFP 0 DC 1.5
VREFM VREFM 0 DC 0

Vin    Vin   0 DC 0  Pulse 0 1.5   0   3000n
Vclock clock 0 DC 0  Pulse 0 1.5   0   200p  200p  4.8n 10n

*X1 VDD VREFP VREFM Vin B7 B6 B5 B4 B3 B2 B1 B0 clock ADC8bit
X1 VDD VREFP VREFM Vin B3 B2 B1 B0 clock ADC4bit

*** START ADC Subcircuit **********************************

*.subckt ADC8bit VDD VREFP VREFM Vin B7 B6 B5 B4 B3 B2 B1 B0 CLOCK
.subckt ADC4bit VDD VREFP VREFM Vin B3 B2 B1 B0 CLOCK

* Set up common mode voltage
BCM   VCM   0     V=(V(VREFP)-V(VREFM))/2

* Set up logic switching point
R3    VDD         VTRIP       100MEG
R4    VTRIP       0           100MEG

* Ideal input sample and hold
XSH   VDD   VTRIP       VIN   OUTSH       CLOCK       SAMPHOLD
```

```
* Level shift by VREFM and 1/2LSB
*BPIP PIPIN       0     V=V(OUTSH)-V(VREFM)+((V(VREFP)-V(VREFM))/2^9)
BPIP  PIPIN       0     V=V(OUTSH)-V(VREFM)+((V(VREFP)-V(VREFM))/2^5)

* 8-bit pipeline ADC
*X7   VDD   VTRIP       VCM   PIPIN       B7    VOUT7       ADCBIT
*X6   VDD   VTRIP       VCM   VOUT7       B6    VOUT6       ADCBIT
*X5   VDD   VTRIP       VCM   VOUT6       B5    VOUT5       ADCBIT
*X4   VDD   VTRIP       VCM   VOUT5       B4    VOUT4       ADCBIT
*X3   VDD   VTRIP       VCM   VOUT4       B3    VOUT3       ADCBIT
* 4-bit pipeline ADC
X3    VDD   VTRIP       VCM   PIPIN       B3    VOUT3       ADCBIT
X2    VDD   VTRIP       VCM   VOUT3       B2    VOUT2       ADCBIT
X1    VDD   VTRIP       VCM   VOUT2       B1    VOUT1       ADCBIT
X0    VDD   VTRIP       VCM   VOUT1       B0    VOUT0       ADCBIT
.ends

* Ideal Sample and Hold subcircuit
.SUBCKT           SAMPHOLD    VDD   VTRIP Vin   Vout  CLOCK
Ein   Vinbuf      0     Vin   Vinbuf      100MEG
S1    Vinbuf  VinS      VTRIP CLOCK switmod
Cs1   VinS  0     1e-10
S2    VinS  Vout1       CLOCK VTRIP switmod
Cout1       Vout1       0     1e-16
Eout  Vout  0     Vout1       0     1
.model switmod SW
.ends

* Pipeline stage
.SUBCKT     ADCBIT      VDD   VTRIP       VCM   VIN   BITOUT      VOUT
S1    VDD   BITOUT      VIN   VCM   switmod
S2    0     BITOUT      VCM     VIN       switmod
Eouth       Vinh  0     VIN   VCM   2
Eoutl       Vinl  0     VIN   0     2
S3    Vinh  VOUT  BITOUT      VTRIP switmod
S4    Vinl  VOUT        VTRIP BITOUT      switmod
.model switmod SW
.ends
*** END ADC Subcircuit ************************************

.end
```

After looking at the plots in Figures 1, 2, and 3, it can be seen that the ADC behaves just as the 8-bit ADC did in the original Figure 30.39 except it uses only 4 bits instead of 8.

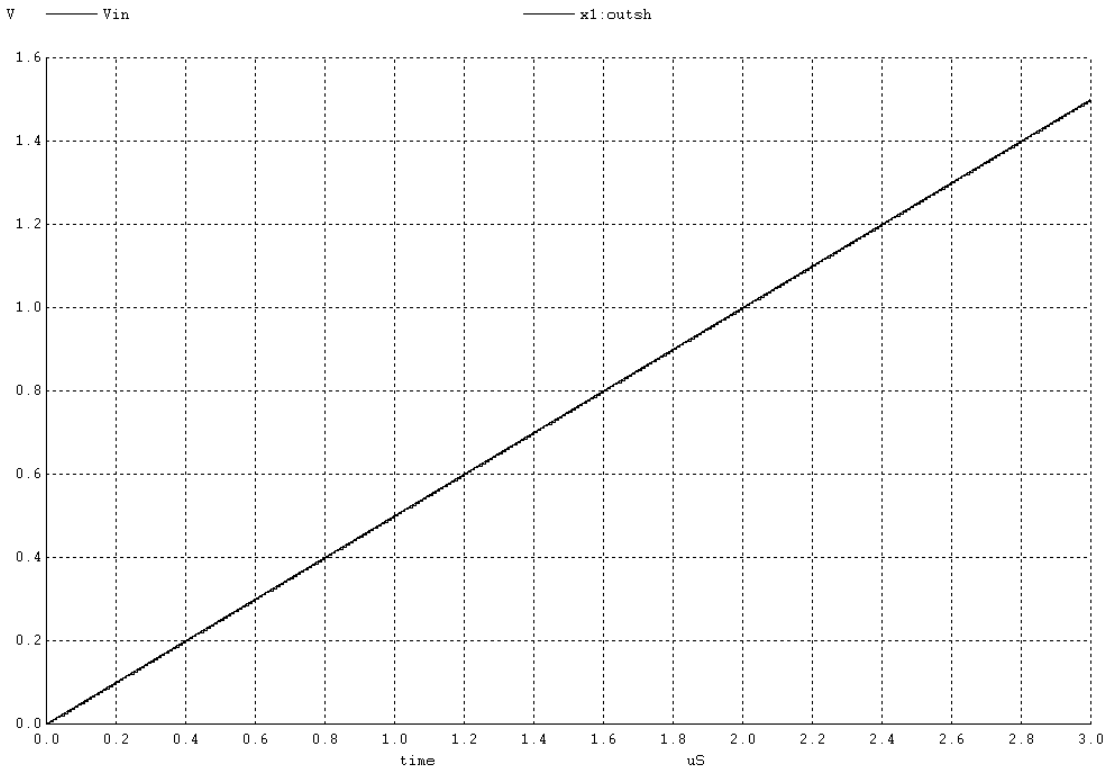


Figure 1

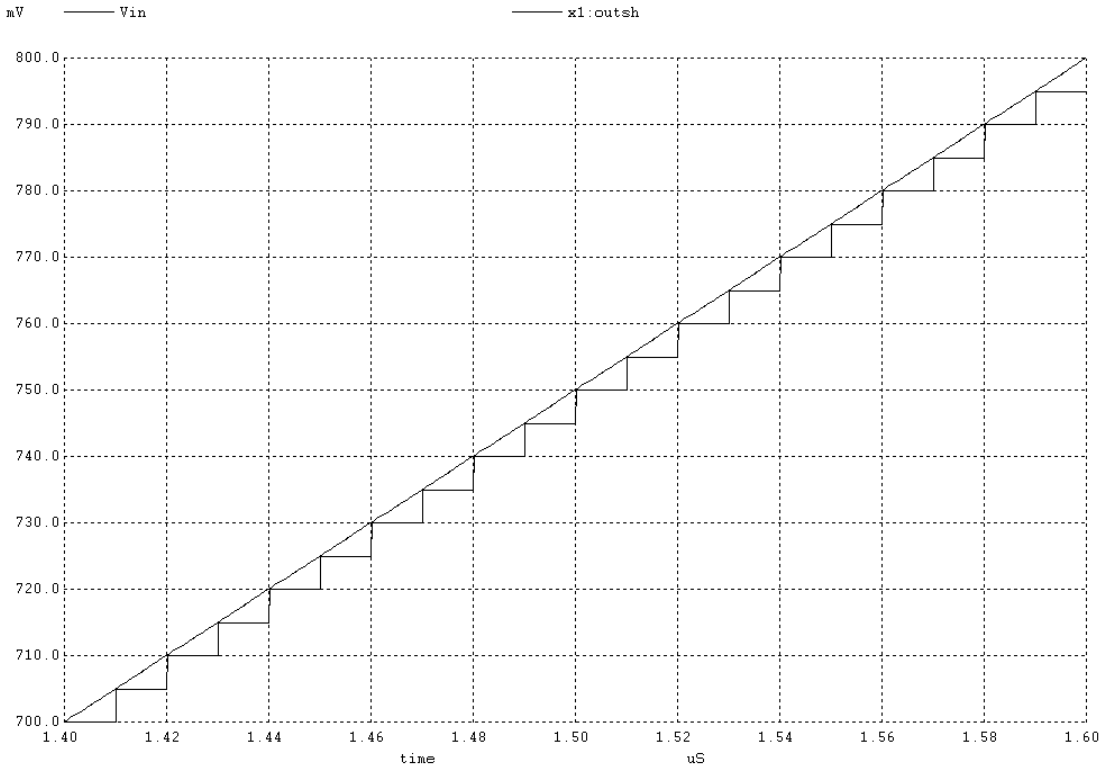


Figure 2

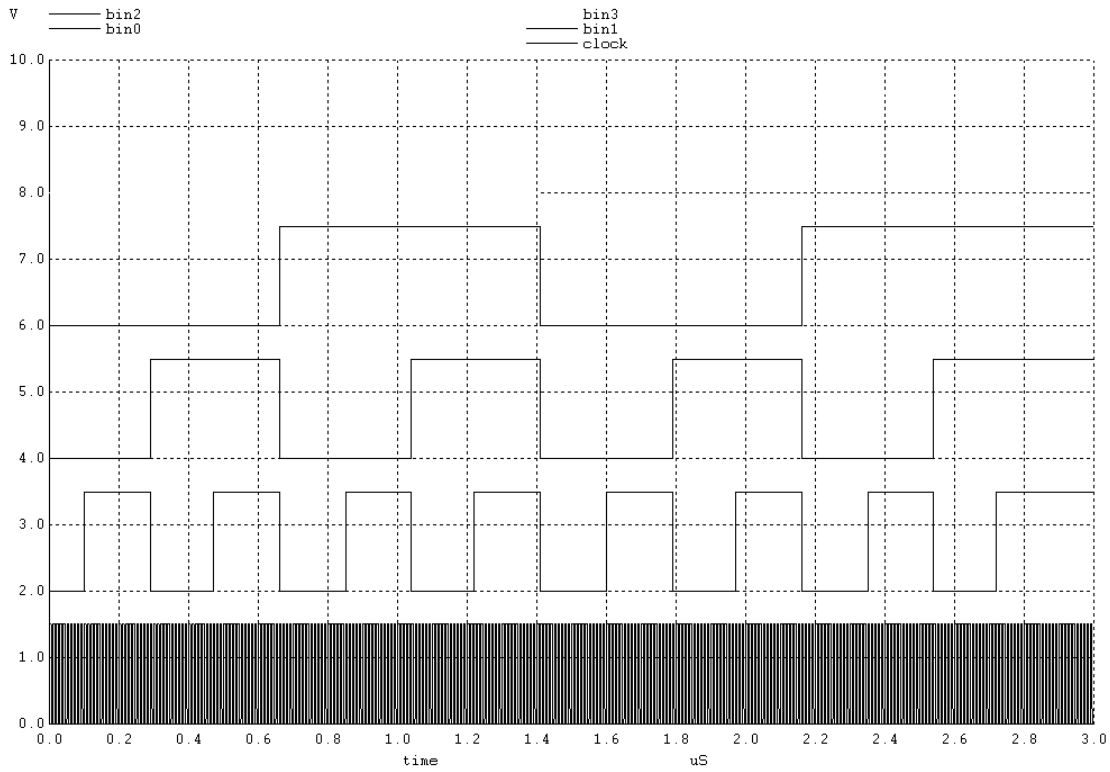

V
bin2
bin0
bin3
bin1
clock
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
time
uS

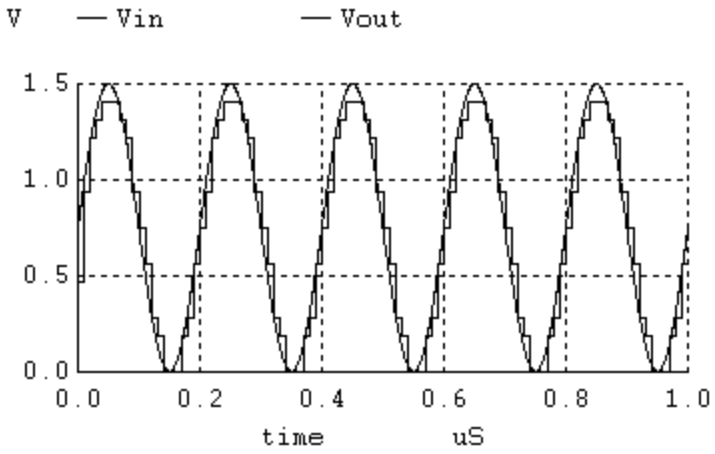
Figure 3

Problem 30.18 Using the models developed in problems 30.15 and 30.17 with a clock frequency of 100MHz apply an input sinewave having an amplitude of 750 mV peak centered around 750 mV DC and a frequency of 5MHz to the input of the 4-bit ADC. If the ideal 4-bit DAC is connected to the digital outputs of the ADC also show the DAC's analog output.

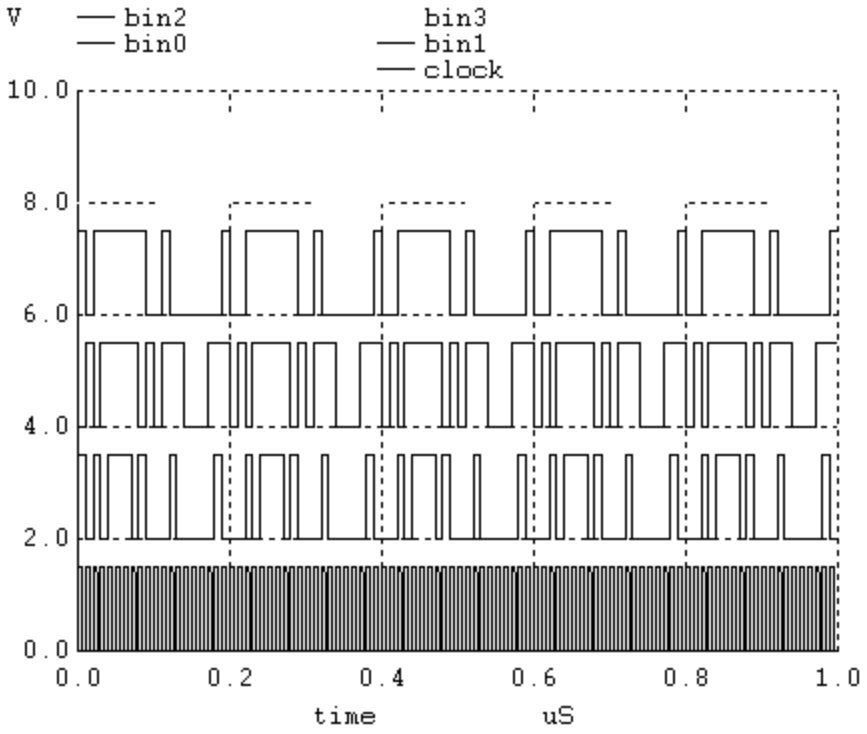
Solution:

```
* Problem 30_18 CMOS: Mixed-Signal Circuit Design *

.tran 1n 1000n 0 1n UIC

*WinSPICE command scripts
*#destroy all
*#run
*#let bin3=b3+8
*#let bin2=b2+6
*#let bin1=b1+4
*#let bin0=b0+2
*#plot clock bin0 bin1 bin2 bin3
*#plot Vin Vout

VDD VDD 0 DC 1.5
VREFP VREFP 0 DC 1.5
VREFM VREFM 0 DC 0

Vin    Vin   0 DC 0  Sin 0.75 0.75 5MEG
Vclock clock 0 DC 0  Pulse 0 1.5   0   200p  200p  4.8n 10n


Xadc VDD VREFP VREFM Vin B3 B2 B1 B0 clock ADC4bit
Xdac VDD VREFP VREFM Vout B3 B2 B1 B0 DAC4bit

*** START ADC Subcircuit **********************************

.subckt ADC4bit VDD VREFP VREFM Vin B3 B2 B1 B0 CLOCK

* Set up common mode voltage
BCM   VCM   0     V=(V(VREFP)-V(VREFM))/2

* Set up logic switching point
R3    VDD         VTRIP       100MEG
R4    VTRIP       0           100MEG

* Ideal input sample and hold
XSH   VDD   VTRIP       VIN   OUTSH       CLOCK       SAMPHOLD

* Level shift by VREFM and 1/2LSB
BPIP  PIPIN       0     V=V(OUTSH)-V(VREFM)+((V(VREFP)-V(VREFM))/2^5)

* 8-bit pipeline ADC
X3    VDD   VTRIP       VCM   PIPIN       B3    VOUT3       ADCBIT
X2    VDD   VTRIP       VCM   VOUT3       B2    VOUT2       ADCBIT
X1    VDD   VTRIP       VCM   VOUT2       B1    VOUT1       ADCBIT
X0    VDD   VTRIP       VCM   VOUT1       B0    VOUT0       ADCBIT
.ends
```

```
* Ideal Sample and Hold subcircuit
.SUBCKT           SAMPHOLD    VDD   VTRIP Vin   Vout  CLOCK
Ein   Vinbuf      0     Vin   Vinbuf      100MEG
S1    Vinbuf  VinS      VTRIP CLOCK switmod
Cs1   VinS  0     1e-10
S2    VinS  Vout1       CLOCK VTRIP switmod
Cout1       Vout1       0     1e-16
Eout  Vout  0     Vout1       0     1
.model switmod SW
.ends

* Pipeline stage
.SUBCKT     ADCBIT      VDD   VTRIP       VCM   VIN   BITOUT      VOUT
S1    VDD   BITOUT      VIN   VCM   switmod
S2    0     BITOUT      VCM     VIN       switmod
Eouth       Vinh  0     VIN   VCM   2
Eoutl       Vinl  0     VIN   0     2
S3    Vinh  VOUT  BITOUT      VTRIP switmod
S4    Vinl  VOUT        VTRIP BITOUT      switmod
.model switmod SW
.ends
*** END ADC Subcircuit ***********************************
*** Start Ideal DAC Subcircuit ******************************

.subckt DAC4bit VDD VREFP VREFM Vout B3 B2 B1 B0

*Generate Logic switching point, or trip, voltage
R1 VDD trip 100MEG
R2 trip 0   100MEG

*Change input logic signals into logic 0s or 1s
X3 trip B3 B3L Bitlogic
X2 trip B2 B2L Bitlogic
X1 trip B1 B1L Bitlogic
X0 trip B0 B0L Bitlogic

*Non-linear dependent source, B, for generating the DAC output
Bout Vout 0 V=((v(vrefp)-
v(vrefm))/16)*(v(B3L)*8+v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)

.ends

.subckt Bitlogic trip BX BXL
Vone  one   0     DC    1
SH    one   BXL   BX    trip  Switmod
SL    0     BXL   trip  BX    Switmod
.model switmod SW
.ends

*** END DAC Subcircuit ***********************************

.end
```


V
Vin
Vout
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
time
uS

—


V
bin2
bin0
bin3
bin1
clock
10.0
8.0
6.0
4.0
2.0
0.0
0.0
0.2
0.4
0.6
0.8
1.0
time
uS

QED

EE515: CMOS Mixed-Signal IC Design

Problem 30.19
Richard Friel
Rich_Friel@AMIS.COM

Problem #30.19 Using Spice, generate the frequency spectrums of the input and output signals in Problem 30.18 using Figure 1 as a reference.

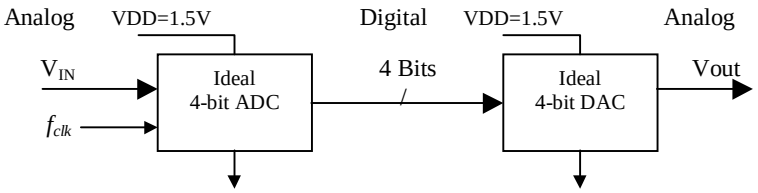


Figure 1.

Ideal 4-bit ADC

One implementation of the ideal ADC is shown in Chapter 28, Figure 29.30. The spice model to be developed, will use an algorithm based on the pipeline ADC. It consists of passing the output of an ideal Sample and Hold (S/H), through the algorithm to generate the output bits.

In order to make the spice model as useful as possible, we want to complete three steps before implementing the pipeline algorithm:

1. Level shift the input signal and reference it to 0V,
2. Find the common mode voltage, Vcm, and reference it to 0V,
3. Shift the ADC transfer curves to the left by ½ LSB as shown in Figure 30.37.

The following non-linear dependent source spice statement will accomplish the three steps,

BPIP PIPIN 0 V=V(OUTSH)-V(REFM)+((V(VREFP-V(REFM))/2^5)

where V(OUTSH) is the output voltage of the ideal S/H [input to pipeline algorithm].

The last term in the spice statement determines ½ LSB for the ADC and is calculated by using Equation 30.29 in the text.

The spice subcircuit model of the pipeline ADC algorithm is listed in Figure 2, under the heading:

*** START IDEAL 4-BIT ADC Subcircuit***

Ideal 4-bit DAC

The output of an Ideal 4-bit DAC can be expressed in the following terms:

$$V_{out} = (V_{ref+} - V_{ref-}) * 1/2^N * (b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + b_{N-3}2^{N-3} + b_0) + V_{ref-}$$

This equation can be implemented in a SPICE model using a non-linear dependant source (a B source). For a 4-bit, Ideal DAC, the SPICE statement would look like:

```
*Non-linear dependant source, B, for generating DAC output
Bout Vout 0 V=((v(vrefp)-v(refm))/16)*(v(B3L)*8+v(B2L)*4+v(B1L)*2+v(B0L))+v(vrefm)
```

The terms BXL correspond to logic signals that have a value of either 1V or 0V.

To implement the VDD/2 switch point, or trip voltage, to create the digital logice levels of 0V or 1V, use the following SPICE statements,

```
*Generate Logic switching point, or trip voltage
R1  VDD trip 100MEG
R2  trip   0     100MEG
```

Switch implementation is shown in Figure 30.34 and in Figure 2 SPICE code, under the heading,

.subckt Bitlogic trip BX BXL

To implement the spectral plotting statement in spice use the following Winspice command script:

```
spec startf stopf stepf vector
```

The above statement calculates a new vector containing the Fourier transform of the input vector. This vector should be the output of a transient analysis.

Note that the time axis of the input vector should be linearized first by using the 'linearize' , because **WinSpice3** does not produce a linear time axis for transient analyses. After using the 'spec' command, the spectrum can be displayed by plotting the magnitude of the resultant vector. For example, after a transient analysis resulting in transient vector v(1), the spectrum can be plotted with the following commands:-

```
linearize v(1)
spec 10 100000 5000 v(1)
plot mag(v(1))
```

The spice subcircuit model of the DAC algorithm is listed in Figure 2, under the heading:

*** START IDEAL 4-BIT DAC Subcircuit***

```
* Figure 2, Input → ADC → DAC → Output,show Input/Output spectrum


.tran .1n 1000n 0 .1n UIC
.save Vout Vin

*WinSPICE command scripts
*#destroy all
*#run
*#linearize Vout Vin
*#spec 0 200MEG 1MEG Vout Vin
*#let voutdb=db(Vout)
*#let vindb=db(Vin)
*#plot voutdb vindb

VDD VDD 0 DC 1.5
VREFP VREFP 0 DC 1.5
VREFM VREFM 0 DC 0
.options reltol=1u

Vin    Vin   0 DC 0  Sin   0.75 0.75  5MEG
Vclock clock 0 DC 0  Pulse 0    1.5   0    200p  200p  4.8n 10n

X1 VDD VREFP VREFM Vin B3 B2 B1 B0 clock ADC4bit
X2 VDD VREFP VREFM Vout B3 B2 B1 B0 DAC4bit

*** Start Ideal DAC Subcircuit ******************************

.subckt DAC4bit VDD VREFP VREFM Vout B3 B2 B1 B0

*Generate Logic switching point, or trip, voltage
R1 VDD trip 100MEG
R2 trip 0   100MEG

*Change input logic signals into logic 0s or 1s
X3 trip B3 B3L Bitlogic
X2 trip B2 B2L Bitlogic
X1 trip B1 B1L Bitlogic
X0 trip B0 B0L Bitlogic

*Non-linear dependent source, B, for generating the DAC output
Bout Vout 0 V=((v(vrefp)-v(vrefm))/16)*(v(B3L)*8+v(B2L)*4+v(B1L)
+ *2+v(B0L))+v(vrefm)

.ends

.subckt Bitlogic trip BX BXL
Vone  one   0     DC    1
SH    one   BXL   BX    trip  Switmod
SL    0     BXL   trip  BX    Switmod
.model switmod SW
.ends

*** END DAC Subcircuit ***********************************
```

Figure 2, Continued…..

```
*** START ADC Subcircuit ***********************************

.subckt ADC4bit VDD VREFP VREFM Vin B3 B2 B1 B0 CLOCK

* Set up common mode voltage
BCM   VCM   0     V=(V(VREFP)-V(VREFM))/2

* Set up logic switching point
R3    VDD         VTRIP       100MEG
R4    VTRIP       0           100MEG

* Ideal input sample and hold
XSH   VDD   VTRIP       VIN   OUTSH       CLOCK       SAMPHOLD

* Level shift by VREFM and 1/2LSB
BPIP PIPIN 0 V=V(OUTSH)-V(VREFM)+((V(VREFP)-V(VREFM))/2^9)

* 4-bit pipeline ADC
X3    VDD   VTRIP       VCM   PIPIN       B3    VOUT3       ADCBIT
X2    VDD   VTRIP       VCM   VOUT3       B2    VOUT2       ADCBIT
X1    VDD   VTRIP       VCM   VOUT2       B1    VOUT1       ADCBIT
X0    VDD   VTRIP       VCM   VOUT1       B0    VOUT0       ADCBIT
.ends

* Ideal Sample and Hold subcircuit
.SUBCKT      SAMPHOLD VDD     VTRIP Vin   Vout  CLOCK
Ein   Vinbuf      0     Vin   Vinbuf      100MEG
S1    Vinbuf  VinS      VTRIP CLOCK switmod
Cs1   VinS  0     1e-10
S2    VinS  Vout1       CLOCK VTRIP switmod
Cout1       Vout1       0     1e-16
Eout  Vout  0     Vout1       0     1
.model switmod SW
.ends

* Pipeline stage
.SUBCKT     ADCBIT      VDD   VTRIP       VCM   VIN   BITOUT
      VOUT
S1    VDD   BITOUT      VIN   VCM   switmod
S2    0     BITOUT      VCM     VIN       switmod
Eouth       Vinh  0     VIN   VCM   2
Eoutl       Vinl  0     VIN   0     2
S3    Vinh  VOUT  BITOUT      VTRIP switmod
S4    Vinl  VOUT        VTRIP BITOUT      switmod
.model switmod SW
.ends

*** END ADC Subcircuit ***********************************

.end
```

The plot in Figure 3, shows the 5MHz input signal to the ADC, Vin, and the resulting DAC output, Vout, reconstructed waveforms as simulated from the SPICE code in Figure 2. The DAC output has not been filtered.

Figure 4 shows the spectral densities of the 5MHz input, Vin, and the reconstructed DAC output, Vout. The noise floor of the input is about –150dB, while the noise floor of the output has been raised to –50dB.

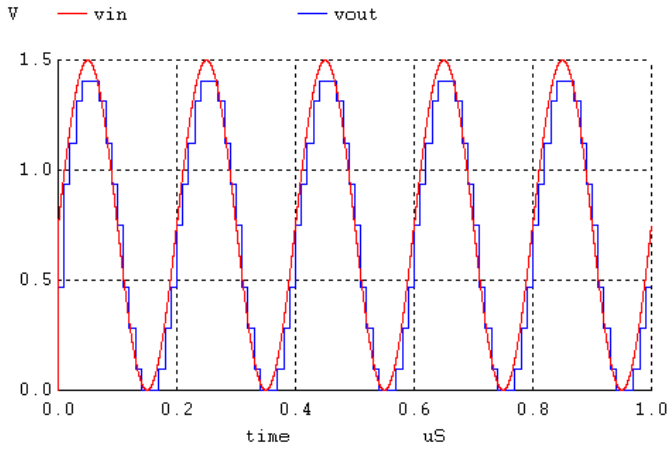


Figure 3.

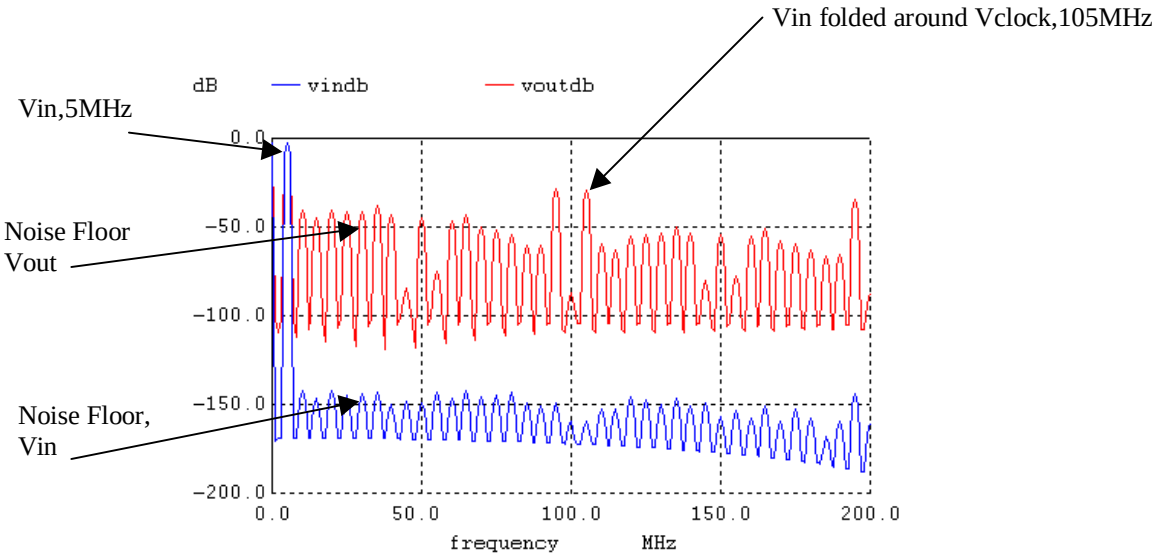


Figure 4.

30.20

Does an ideal S/H introduce amplitude quantization noise into an input waveform? Why or why not?

An ideal S/H does not introduce amplitude quantization noise into an input waveform. The amplitude of the input into the ideal S/H is exactly the amplitude out, at the moment of sampling. The quantization noise comes from the difference between the amplitude of the analog signal and the amplitude of the quantized or digitized level of the output of the ADC. So quantization noise is added to the signal after it is sampled during the conversion from an analog to a digital signal.

EE515: CMOS Mixed-Signal IC Design
Brian Bergeson
Problem 30.21
bbergeso@poci.amis.com

Problem 30.21:

Why are the amplitudes of the mirror images decreasing with an increase in frequency in fig. 30.44b?

Answer:

The reason why the amplitude is decreasing with increasing frequency is because of the sync response filter attenuation in the output signal of the S/H circuit.

Explanation:

Fig. 30.44b shows the output spectrum of the circuit in fig. 30.43. This circuit calculates the difference between the input and the output of an ideal S/H. Fig. 30.44b shows the spectrum of the signal that represents this difference. The output spectrum of an ideal S/H is attenuated as if it was passed through a sync response filter (see fig 30.29). From page 23 of CMOS: Mixed-Signal Circuit Design: "In other words, the S/H can be thought of as an ideal impulse sampler followed by a sync response filter." Since the input is not attenuated, but the output is, the difference between the two will also show the sync response attenuation. This attenuation is why the amplitude decreases with increasing frequency.

30.22

$V_{Qe,RMS} = \sqrt{(1/T \int_0^T (0.5LSB - 1LSB.t/T)^2 \, dt}$

$V_{Qe,RMS} = \sqrt{(1/T \int_0^T (0.25LSB^2 - LSB^2.t/T + LSB^2.t^2/T^2) \, dt}$

$V_{Qe,RMS} = \sqrt{(1/T \, (0.25LSB^2.t - LSB^2.t^2/2T + LSB^2.t^3/3T^2) \, |_0^T}$

$V_{Qe,RMS} = \sqrt{(1/T \, (0.25LSB^2.T - LSB^2.T^2/2T + LSB^2.T^3/3T^2)}$

$V_{Qe,RMS} = \sqrt{(LSB^2/4 - LSB^2/2 + LSB^2/3)}$

$V_{Qe,RMS} = \sqrt{(LSB^2/12)}$

$V_{Qe,RMS} = V_{LSB}/\sqrt{12}$

EE515: CMOS Mixed-Signal IC Design

Problem 30.23
Jim Slupe

30.23 How are voltage spectral density, power spectral density (PSD), average power, and RMS voltage related for a random signal? What are the units for each? Provide answers for both continuous signals and signals that are only defined at discrete frequencies.

Answer:

Discrete signals

RMS voltage is related peak voltage and to power spectral density by equation 30.33 given here:

$V_{Qe,RMS}$ =(1 / sqrt(2))*(sqrt(summation(V^2_{FFT}*(k ·f_{RES}))))
where the summation is from k=0 to k=M and M = #FFT points. In this formula which defines RMS quantization noise voltage for a discrete signal, power spectral density is the term V^2_{FFT}. If one were to increase this $V_{Qe,RMS}$ value by 3 dB (multiply by root 2) one would then have the peak voltage or voltage spectral density.

Continuous signals

The equation 30.33 can be modified to reflect continuous signals by substituting $((V(f)|_{f=n*f_{res}})$ / delta t)^2 for the term V^2_{FFT} (refer to equation 30.40). The relationships would be the same as those for the discrete signals given above.

Units

Voltage Spectral Density: Volts divided by the square root of Hertz.
Power Spectral Density: Volts squared divided Hertz.
RMS Voltage: Volts divided by the square root of Hertz.
Average Power: Volts squared or dB.

Brandon Roth
brandonroth@micron.com

30.24 How would we convert voltage spectral density of Fig. 30.48 into a power spectral density plot? What term in Eq. 30.33 is the PSD? How would we rewrite Eq. (30.33) to give the average power of the quantization error?

In general power is given by $P = V^2/R$. If we normalize the power we assume a resistance of one ohm.

Then $P = V^2/1$.

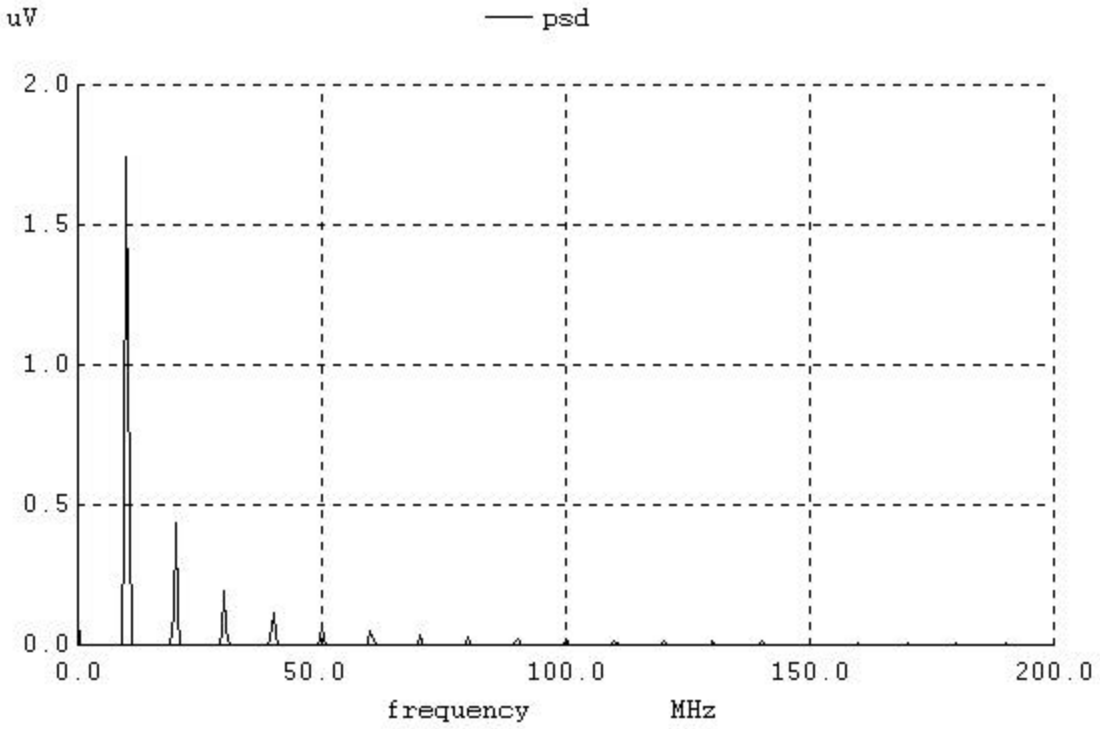
To convert the voltage spectrum to a power spectrum:

1. Calculate the RMS voltage by dividing by the sqrt of 2 or multiplying by 0.707.
2. Then square the RMS quantization voltage.

Adding the following lines to winSpice example 30.48 will plot the power spectral density of the quantization noise.

```
*#let vrms = mag(voutd)*0.707
*#let psd = vrms*vrms
*#plot psd
```

Below is the power spectral density plot from the changes above.



Power Spectral Density of Quantization Noise.

In equation 30.33 the V^2_{FFT} term is the PSD. Note again a resistor value of 1 ohm was assumed.

$$V_{Qe,RMS} = \frac{1}{\sqrt{2}} \sqrt{\sum_{k=0}^{M-1} V^2{}_{FFT}(k \times f_{res})} \qquad (30.33)$$

Use equation 30.33 to solve for total average power. The average power for each spectra is equal to the quantization noise RMS voltage squared. The total average power is the sum of the average power for each spectra in the spectrum.

$$P_{AVG} = V^2{}_{RMS}(k \times f_{res}) \qquad \text{for any spectra k.}$$

$$P_{TotalAVG} = \sum_{k=0}^{M-1} V^2{}_{RMS}(k \times f_{res})$$

$$Note: V_{RMS} = \frac{V_{FFT}}{\sqrt{2}}$$

Where M is the number of FFT samples and f_{res} is the FFT resolution.

Using equation 30.33....

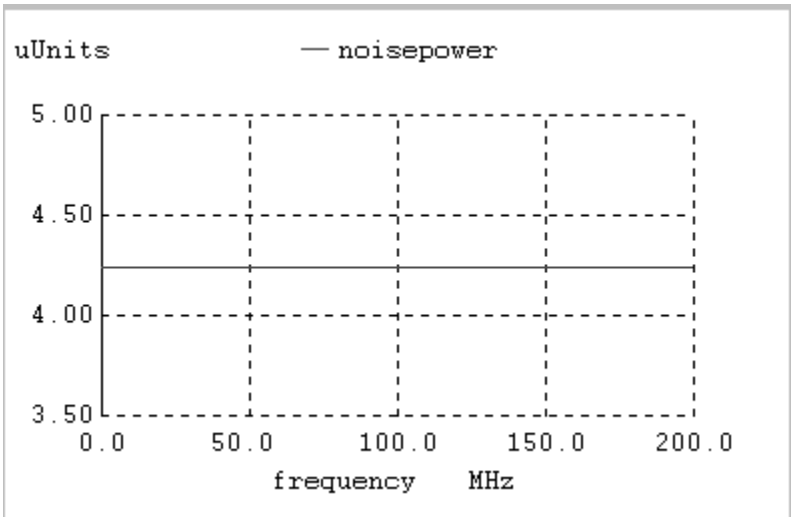
$$P_{TotalAVG} = \frac{1}{\sqrt{2}} \sum_{k=0}^{M-1} V^2{}_{FFT}(k \times f_{res})$$

Gexin
Ghuang@uidaho.edu

Problem 30.25:
Repeat Ex. 30.10 if we want to determine the quantization noise power. Show the simulation results and the commands used to determine this power.

Solution for problem 30.25:

The simulation result is shown below:



From the simulation result, the quantization noise power is 4.242uw.

The command used to generate the quantization noise power is given below:

```
* For Problem 30.25 *

.tran .2n 2000n 0 .2n UIC

*WinSPICE command scripts
*#destroy all
*#run
*#plot Vin Vout xlimit 900n 1500n ylimit 740m 790m
*#plot Voutd xlimit 900n 1500n ylimit -7m 7m
*#linearize Voutd Vin
*#spec 0 200MEG 0.5MEG Voutd Vin
*#let m=mag(Voutd)
*#let qnoise=0.707*sqrt(mean(m*m)*length(m))
*#let noisepower=qnoise*qnoise
*#plot noisepower
```

David Goldman

EE 515 – CMOS Mixed-Signal IC Design

Problem 30.26
Question: Derive Eq. (30.43).

Answer: Eq. (30.42) relates the quantization noise power (derived from quantization noise voltage in Eq. (30.30)) to the sum of the spectral density from DC to the Nyquist frequency.

$$\frac{V^2_{LSB}}{12} = 2\int_0^{f_s/2} V^2_{Qe}(f)\cdot df$$

This equation assumes filtering any alias spectra above the Nyquist frequency. If the voltage spectrum is assumed to be flat, then $V^2_{Qe}(f)$ is a constant and the equation can be evaluated.

$$\frac{V^2_{LSB}}{12} = 2\cdot\left[V^2_{Qe}(f)\cdot f\right]_0^{f_s/2}$$

$$\frac{V^2_{LSB}}{12} = 2V^2_{Qe}(f)\cdot\left(f_s/2-0\right)$$

$$\frac{V^2_{LSB}}{12} = V^2_{Qe}(f)\cdot f_s$$

$$V^2_{Qe}(f) = \frac{V^2_{LSB}}{12 f_s}$$

$$V_{Qe}(f) = \frac{V_{LSB}}{\sqrt{12 f_s}}$$

In order to continue, a new equation that relates the V_{LSB} to the reference voltages and number of bits is needed. This is found in Eq. (30.23).

$$1LSB = \frac{V_{REF+} - V_{REF-}}{2^N} = V_{LSB}$$

Replacing the V_{LSB} from the evaluated Eq. (30.42) with Eq. (30.23), the final form of Eq. (30.43) can be written.

$$V_{Qe}(f) = \frac{V_{REF+} - V_{REF-}}{2^N\sqrt{12 f_s}}$$ □

David Goldman

EE 515 – CMOS Mixed-Signal IC Design

Problem 30.27

Question: What term is the PSD in Eqs. (30.42) and (30.44)? What are its units?

Answer: Power Spectral Density (PSD) can be found by squaring voltage spectral density (Note: this is true because dividing the voltage spectral density by a resistance of 1 Ω is understood and not explicitly written.) The variable for noise voltage spectral density is $V_{Qe}(f)$ and has units of $V/\sqrt{Hz}$. Thus, PSD in Eqs. (30.42) and (30.44) is written as $V^2_{Qe}(f)$ and has units of V^2/Hz. The position of this term in each equation is identified below.

$$\frac{V^2_{LSB}}{12} = 2\int_0^{f_s/2} \underbrace{V^2_{Qe}(f)}_{\text{PSD}} \cdot df$$

$$V^2_{Qe,RMS} = 2\int_{f_L}^{f_H} \underbrace{V^2_{Qe}(f)}_{\text{PSD}} \cdot df$$

Gexin
Ghuang@uidaho.edu

Problem 30.28:
Verify, with simulation, Ex. 30.16.

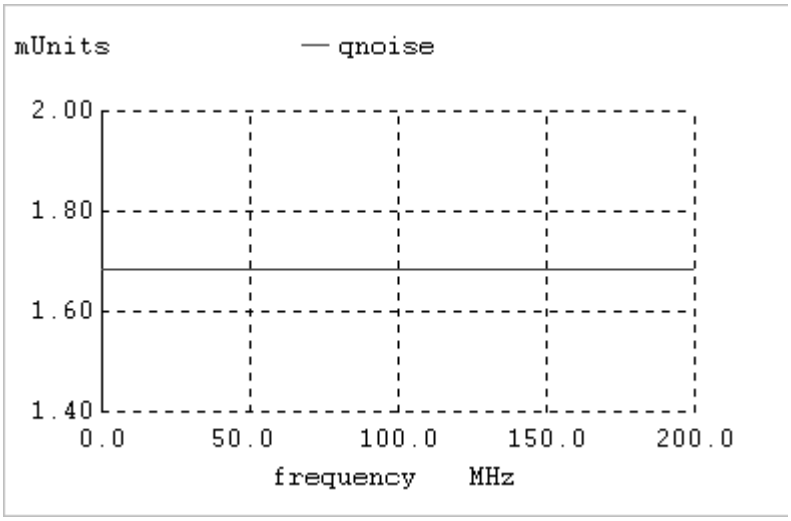
Solution for problem 30.28:

a)The simulation result for sampling frequency 100MHZ is shown below.

From the simulation result, the quantization noise is 1.7115mv.

For the 100MHZ clock signal, use the following statement in the netlist:
Vclock clock 0 DC 0 Pulse 0 1.5 0 200p 200p 4.8n 10n

The simulation result for sampling frequency 100MHZ.

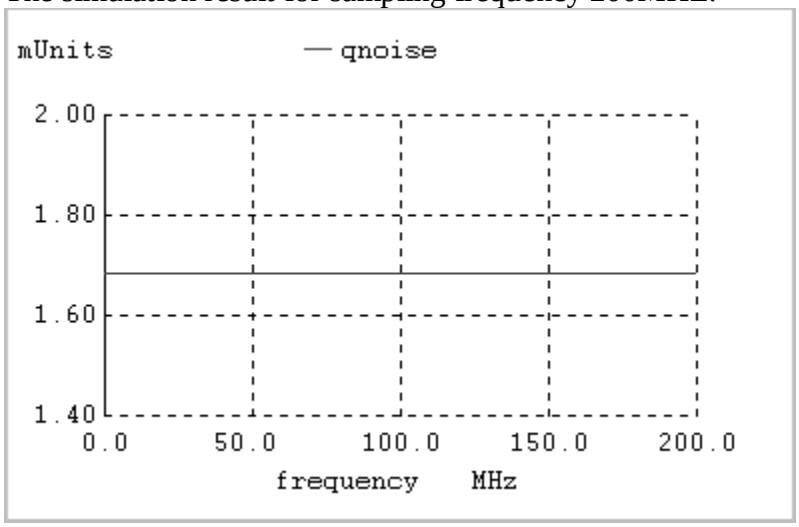


b)The simulation result for sampling frequency 200MHZ is shown below.

From the simulation result, the quantization noise is 1.683mv.

For the 200MHZ clock signal, use the following statement in the netlist:
Vclock clock 0 DC 0 Pulse 0 1.5 0 100p 100p 2.4n 5n

The simulation result for sampling frequency 200MHZ.



Brandon Roth
brandonroth@micron.com

30.29 Verify, using simple circuit analysis, that resistors can be used to implement averaging as seen in Fig. 30.59 and Eq. 30.45.

Referring to the simple circuit below. Each resistor is equal.

Using superposition,

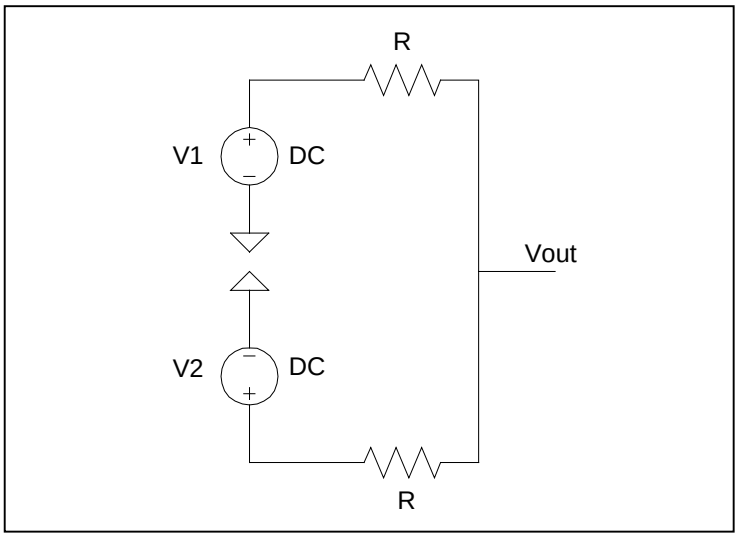
$$Vout_1 = \frac{R}{R+R} V_A = V_A/2$$

$$Vout_2 = \frac{R}{R+R} V_B = V_B/2$$

Then

$$Vout = Vout_1 + Vout_2 = V_A/2 + V_B/2 = \frac{V_A + V_B}{2}$$

If V_A = 1V and V_B = 3V then Vout = 1.5V, the average of V_A and V_B.

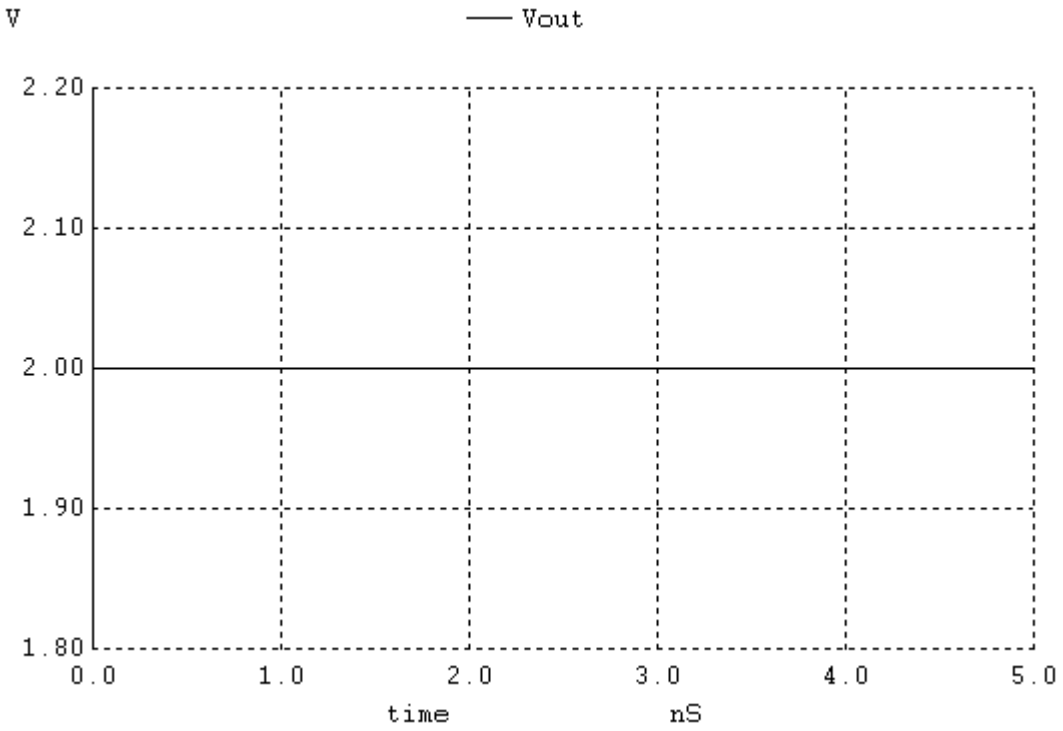


Simple Circuit to Implement the average of two signals.

Proof using Winspice: Below is a netlist and results for the above example extending the previous basic example to the average of three inputs.

```
*Chapter 30 problem 29*

*** Top Level Netlist ***
VA      1 0 DC 1
VB      2 0 DC 1
VC      3 0 DC 4
```

```
R1      1 Vout 1000
R2      2 Vout 1000
R3      3 Vout 1000

.tran 1n 5n 0 .01n
.print tran all

*#run
*#plot Vout
```



Average of V_A, V_B, and V_C.

EE515: CMOS Mixed-Signal IC Design

Problem 30.30
Jim Slupe

30.30 How does averaging K samples of a random voltage variable reduce its RMS value? How does the power contained in the same variable get reduced by averaging?

$$V_{Qe,RMS} = (1 / \text{sqrt}(K))*(V_{LSB} / \text{sqrt}(12)) \qquad (30.47)$$

Answer: The RMS Quantization Noise Voltage is reduced by one over the root of the averaging factor K in accordance with equation 30.47. The power would reduce by a factor of one over K (as opposed to the root of K) since power is in terms of $V^2_{Qe,RMS}$.

In chapter 30, example 30.16 this RMS Quantization Noise Voltage for the DAC was seen as $V_{Qe,RMS}$ = 1.69 mV (V_{LSB} / sqrt(12) = 5.86 mv / sqrt(12) = 1.69 mV). When subjected to averaging as a result of doubling the frequency spectrum range (which results in K = 2), the new value of $V_{Qe,RMS}$ as described by equation 30.47 is 1.19 mV.

It can also be seen graphically in figure 30.61 whereby the sample rate is effectively doubled (K = 2) but the new spectrum is reduced by 3 dB (20*log (1 / sqrt(2))) compared to the original spectrum.