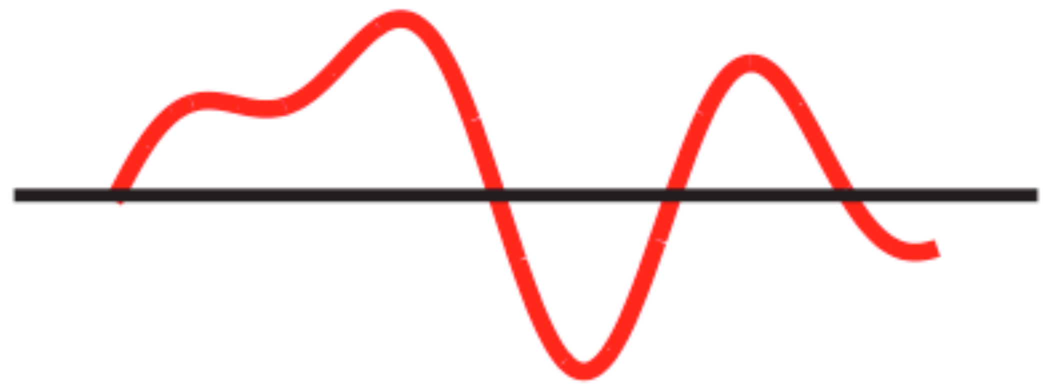
Analog Building Blocks

- Sampling theorem
- Undersampling, antialiasing
- FIR digital filters
- Quantization noise, oversampling
- OpAmps, DACs, ADCs

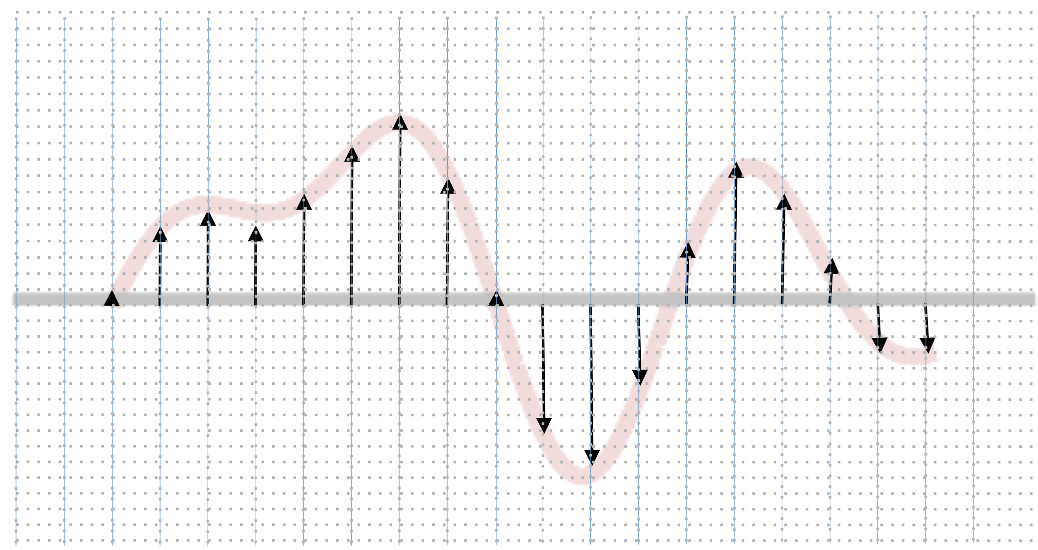
Lab #3 report due on-line @ 5pm today.

Digital Representations of Analog Waveforms

Continuous time
Continuous values



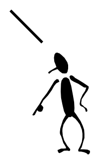
Discrete time
Discrete values



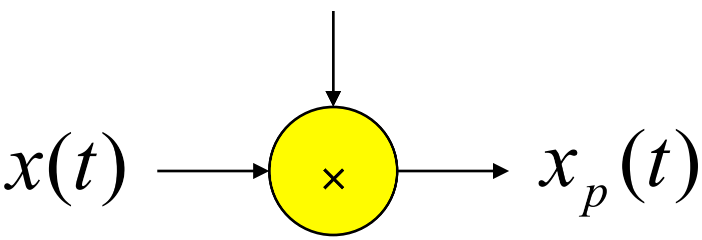
Discrete Time

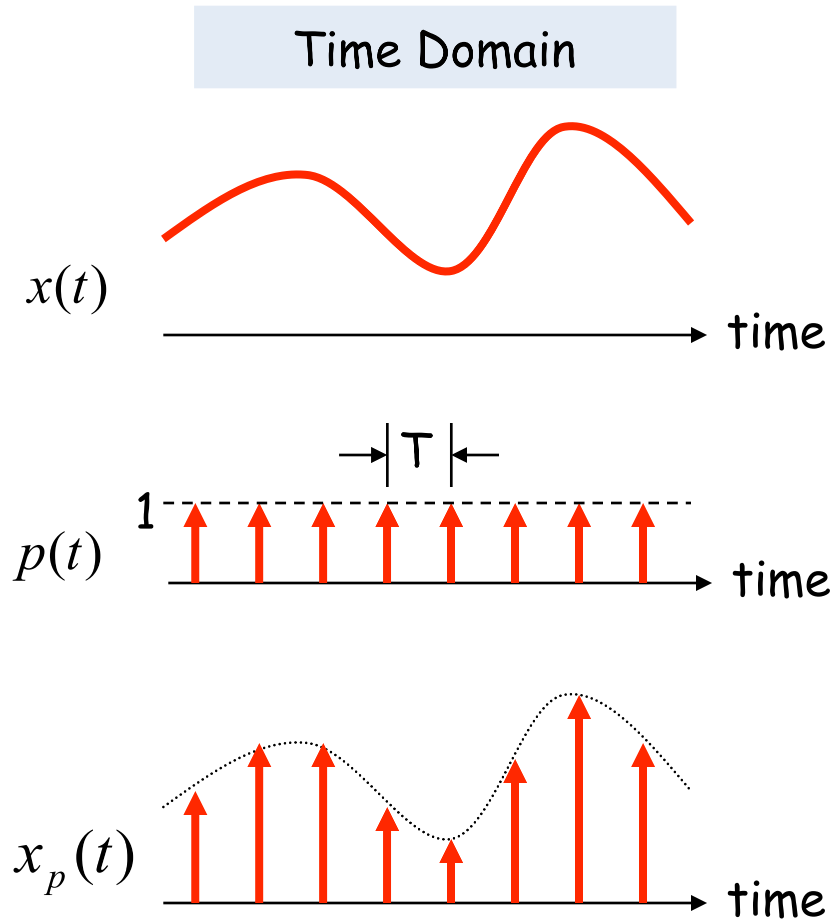
Let's use an impulse train to sample a continuous-time function at a regular interval T:

δ(x) is a narrow impulse at x=0, where $\int_{-\infty}^{\infty} f(t)\delta(t-a)\,dt = f(a)$



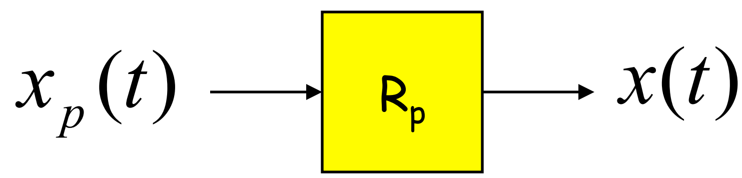
$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$



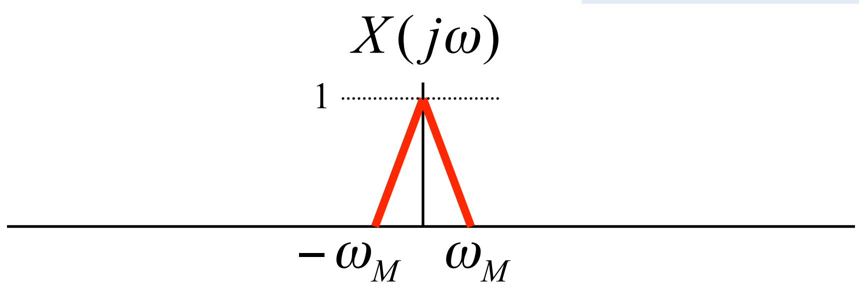


Reconstruction

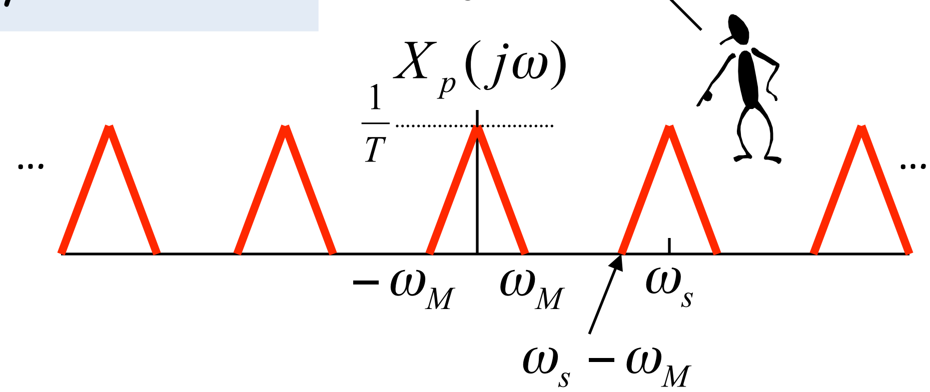
Is it possible to reconstruct the original waveform using only the discrete time samples?

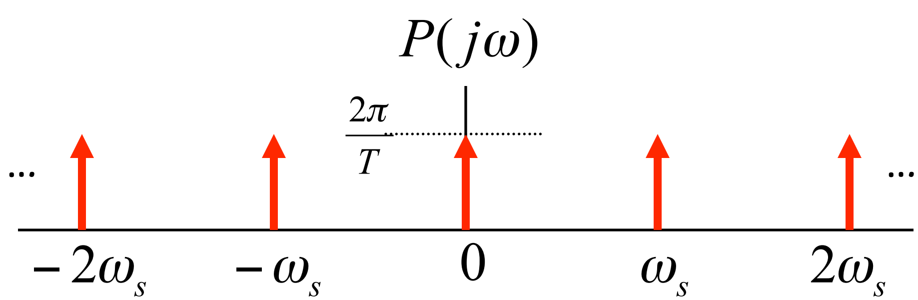


Frequency Domain



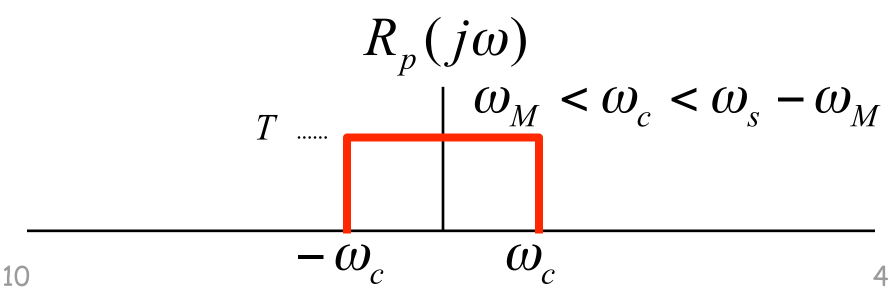
Looks like modulation by ω_s and its harmonics





$$\omega_s = \frac{2\pi}{T}$$

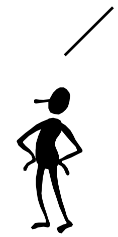
So, if $\omega_m < \omega_s - \omega_m$, we can recover the original waveform with a low-pass filter!



Sampling Theorem

Let x(t) be a band-limited signal, ie, $X(j\omega)=0$ for $|\omega| > \omega_M$. Then x(t) is uniquely determined by its samples x(nT), n = 0, ±1, ±2, …, if

$$\omega_s > 2\omega_M$$



$2\omega_M$ is called the "Nyquist rate" and $\omega_s/2$ the "Nyquist frequency"

where

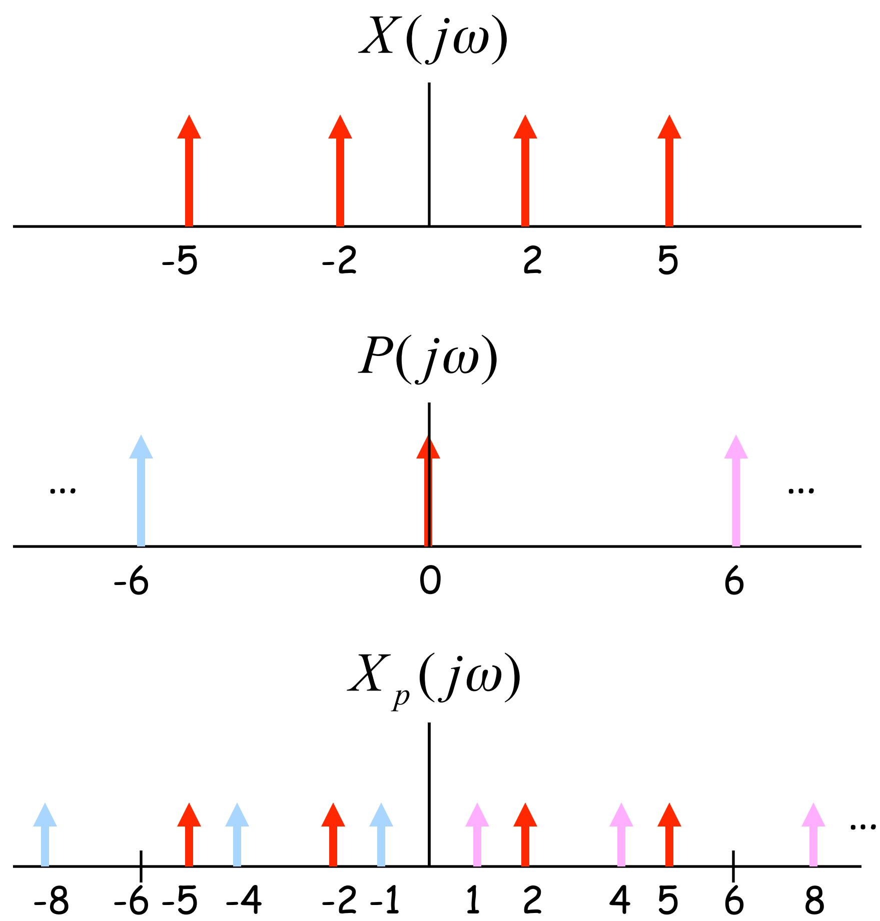
$$\omega_s = \frac{2\pi}{T}$$

Given these samples, we can reconstruct x(t) by generating a periodic impulse train in which successive impulses have amplitudes that are successive sample values, then passing the train through an ideal LPF with gain T and a cutoff frequency greater than ω_M and less than $\omega_s-\omega_M$.

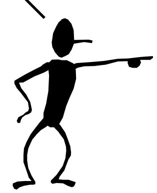
Undersampling → Aliasing

If $\omega_s \leq 2\omega_M$ there's an overlap of frequencies between one image and its neighbors and we discover that those overlaps introduce additional frequency content in the sampled signal, a phenomenon called aliasing.

$$\omega_M = 5, \omega_s = 6$$



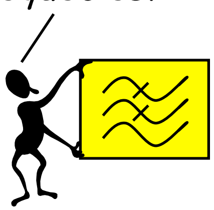
There are now tones at 1 (= 6 – 5) and 4 (= 6 – 2) in addition to the original tones at 2 and 5.



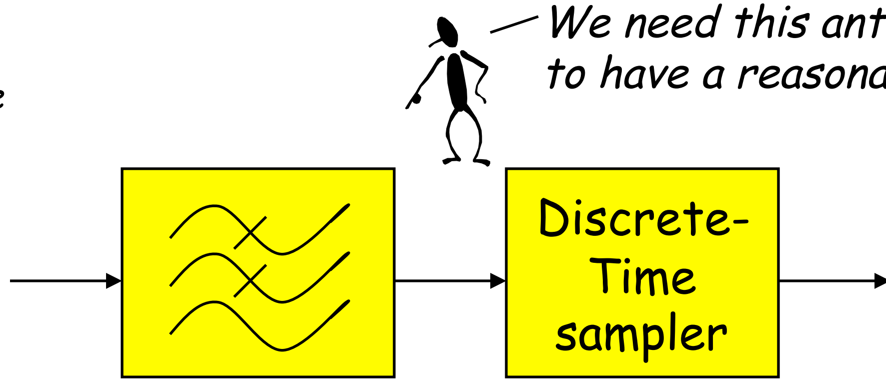
Antialias Filters

If we wish to create samples at some fixed frequency ω_s, then to avoid aliasing we need to use a low-pass filter on the original waveform to remove any frequency content $\geq \omega_s/2$.

This is the symbol for a low-pass filter – see the little "x" marks on the middle and high frequecies?



We need this antialiasing filter – it has to have a reasonably sharp cutoff



$$\omega_C = \frac{\omega_s}{2} \qquad \omega_s$$

The frequency response of human ears essentially drops to zero above 20kHz. So the "Red Book" standard for CD Audio chose a 44.1kHz sampling rate, yielding a Nyquist frequency of 22.05kHz. The 2kHz of elbow room is needed because practical antialiasing filters have finite slope…

Why 44.1kHz? See http://www.cs.columbia.edu/~hgs/audio/44.1.html

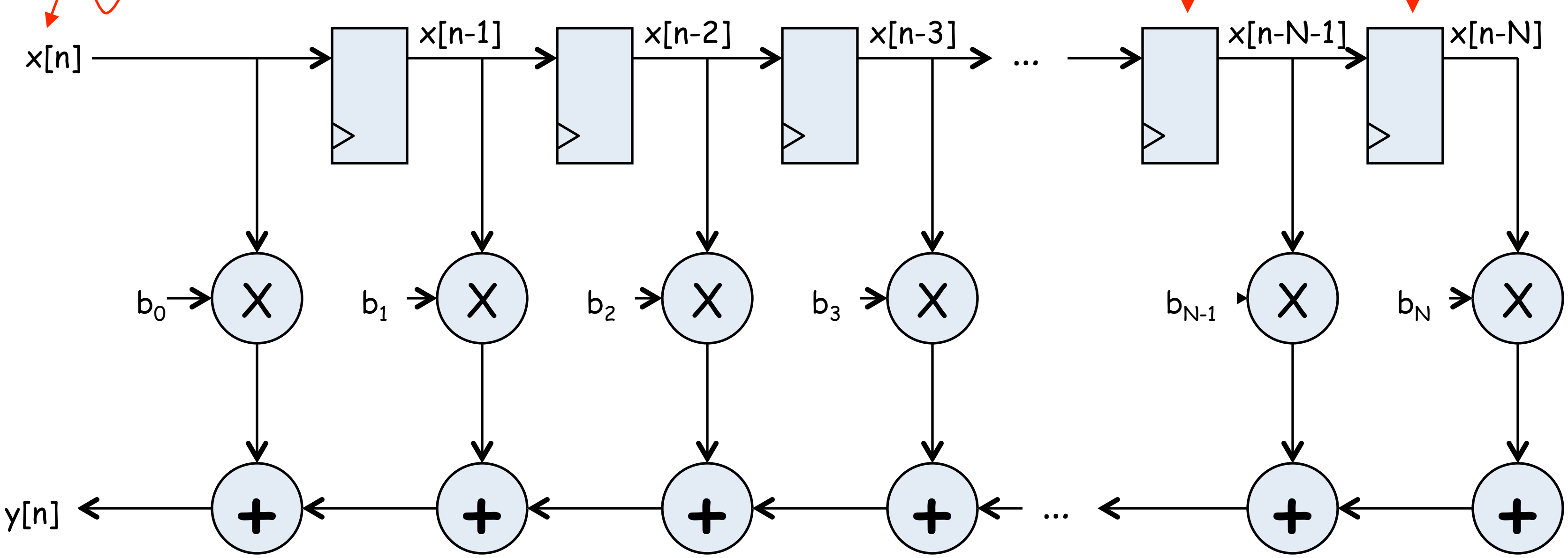
Digital Filters

Equation for an N-tap finite impulse response (FIR) filter:

$$y[n] = \sum_{k=0}^{N} b_k x[n-k]$$

Provide next x every CLK

"shift register" remembers last N values



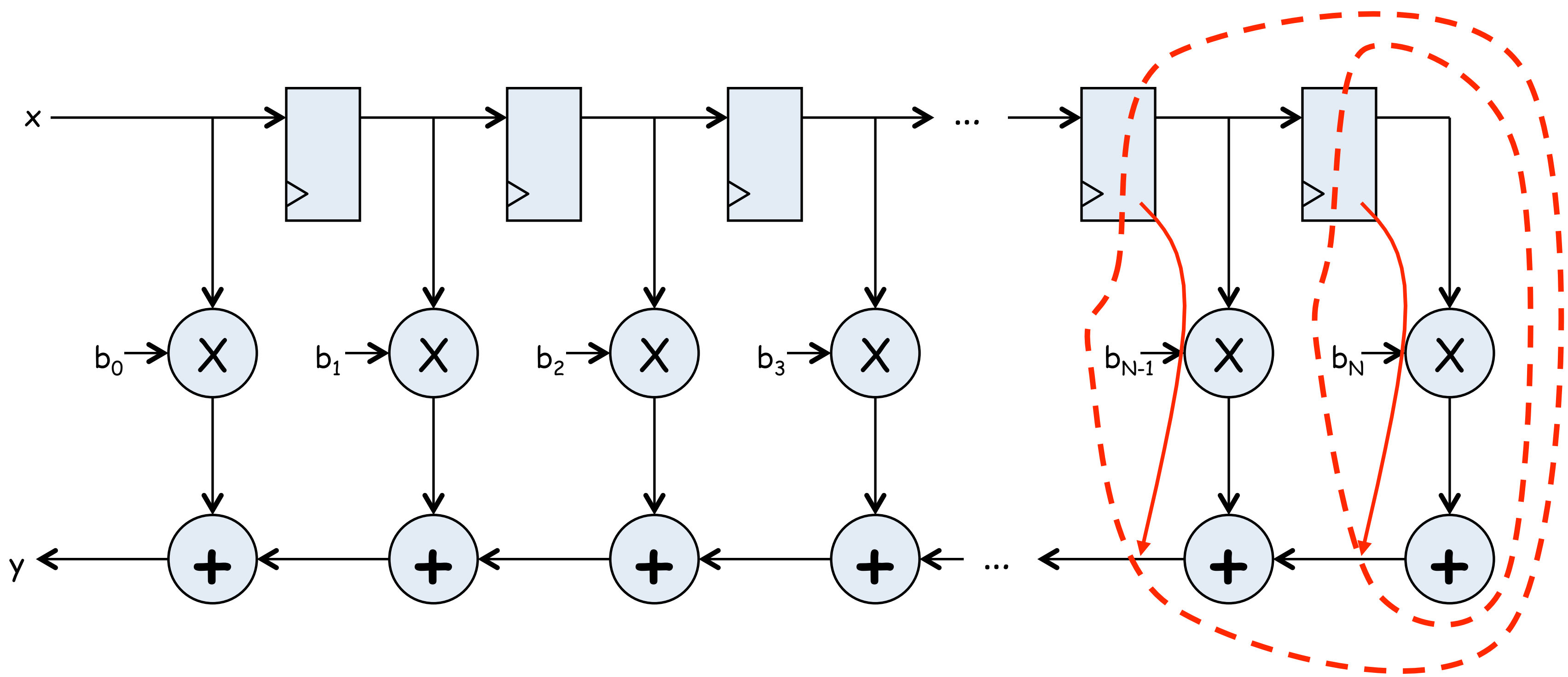
What components are part of the t_{PD} of this circuit?
How does t_{PD} grow as N gets larger?

Filter coefficients

- Use Matlab command: b = fir1(N,ω_C/(ω_S/2))
  - N is the number of taps (we'll get N+1 coefficients). Larger N gives sharper roll-off in filter response; usually want N to be as large as reasonably possible.
  - ω_C is the cutoff frequency (3kHz in Lab 4)
  - ω_S is the sample frequency (48kHz in Lab 4)
  - The second argument to the fir1 command is the cutoff frequency as a fraction of the Nyquist frequency (i.e., half the sample rate).
  - By default you get a lowpass filter, but can also ask for a highpass, bandpass, bandstop.
- The b coefficients are real numbers between 0 and 1. But since we don't want to do floating point arithmetic, we usually scale them by some power of two and then round to integers.
  - Since coefficients are scaled by 2^S, we'll have to re-scale the answer by dividing by 2^S. But this is easy – just get rid of the bottom S bits!

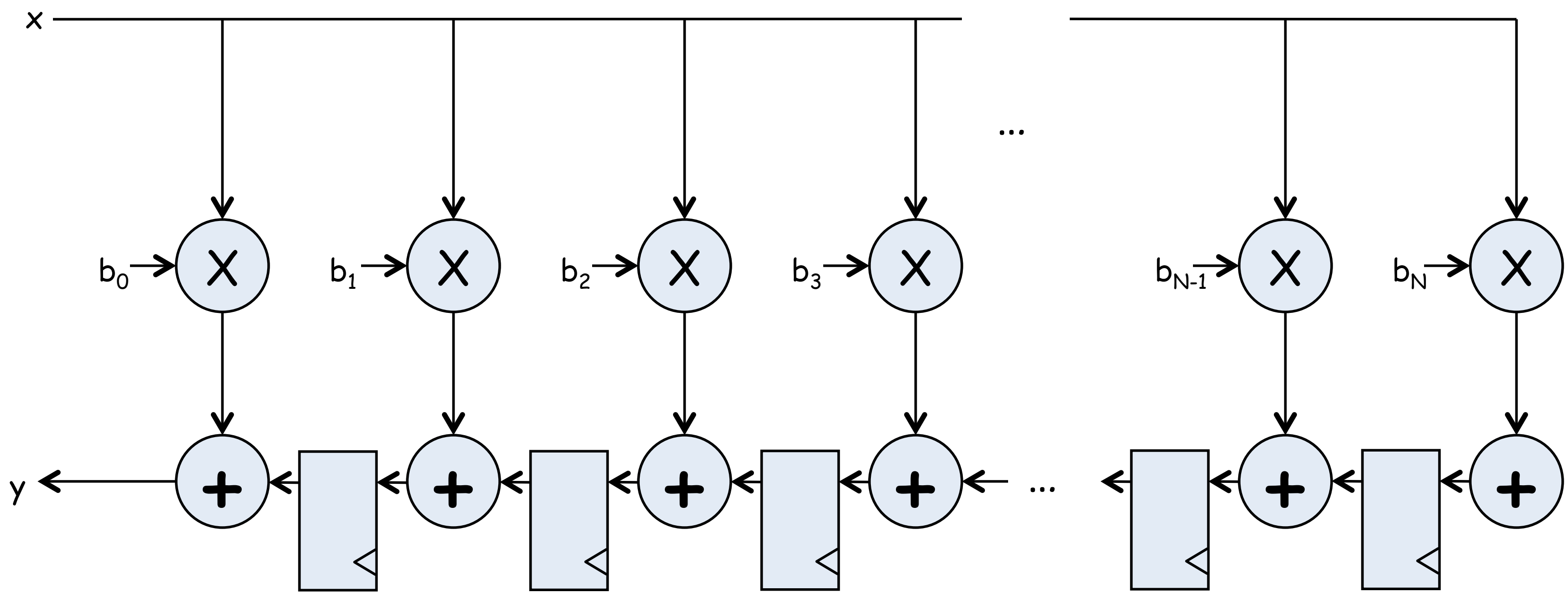
Retiming the FIR circuit

Apply the cut-set retiming transformation repeatedly…



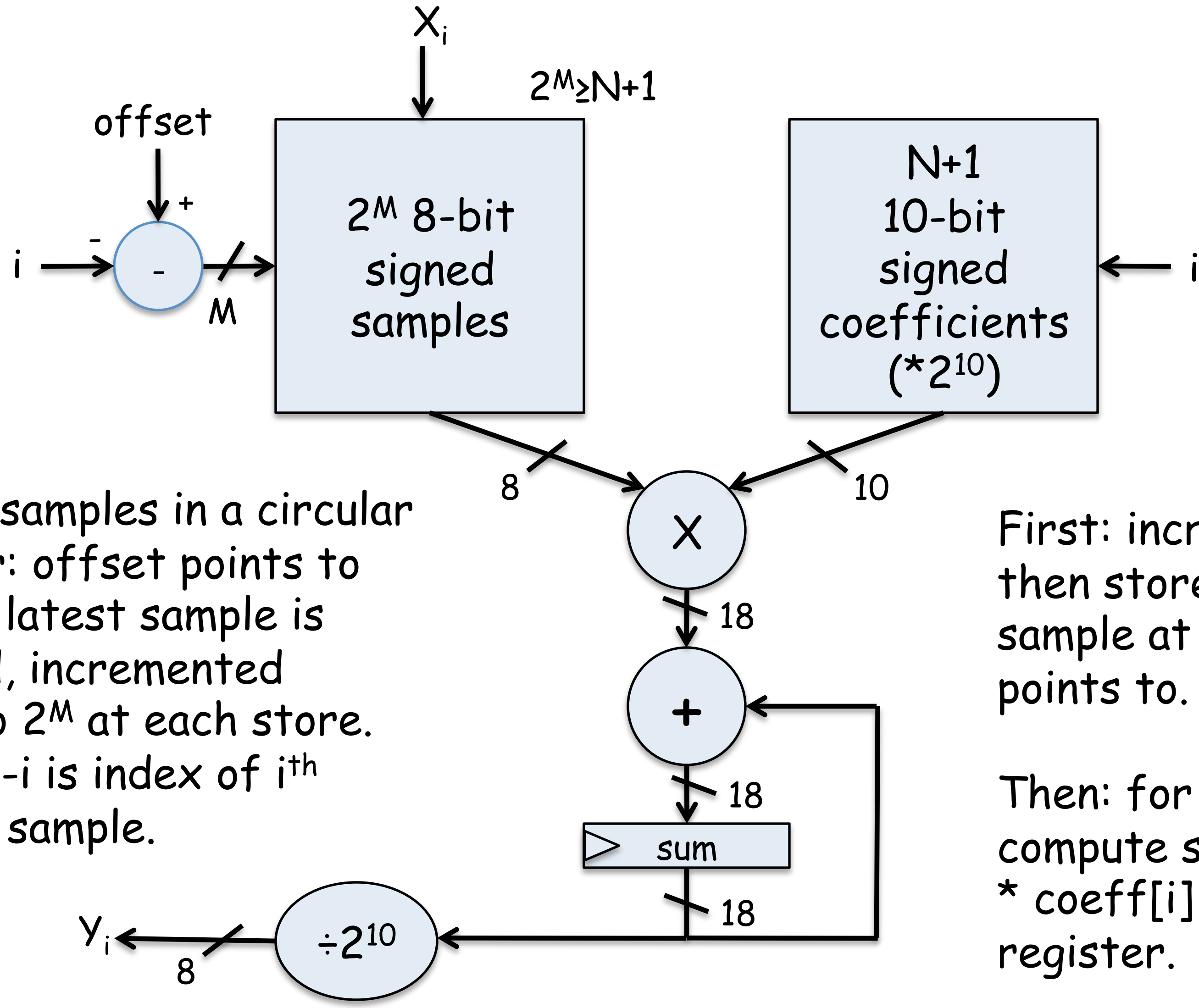
Retimed FIR filter circuit

"Transposed Form" of a FIR filter



What components are part of the t_{PD} of this circuit?
How does t_{PD} grow as N gets larger?

N-tap FIR: less hardware, N+1 cycles…



Store samples in a circular buffer: offset points to where latest sample is stored, incremented modulo 2^M at each store. offset-i is index of i^{th} oldest sample.

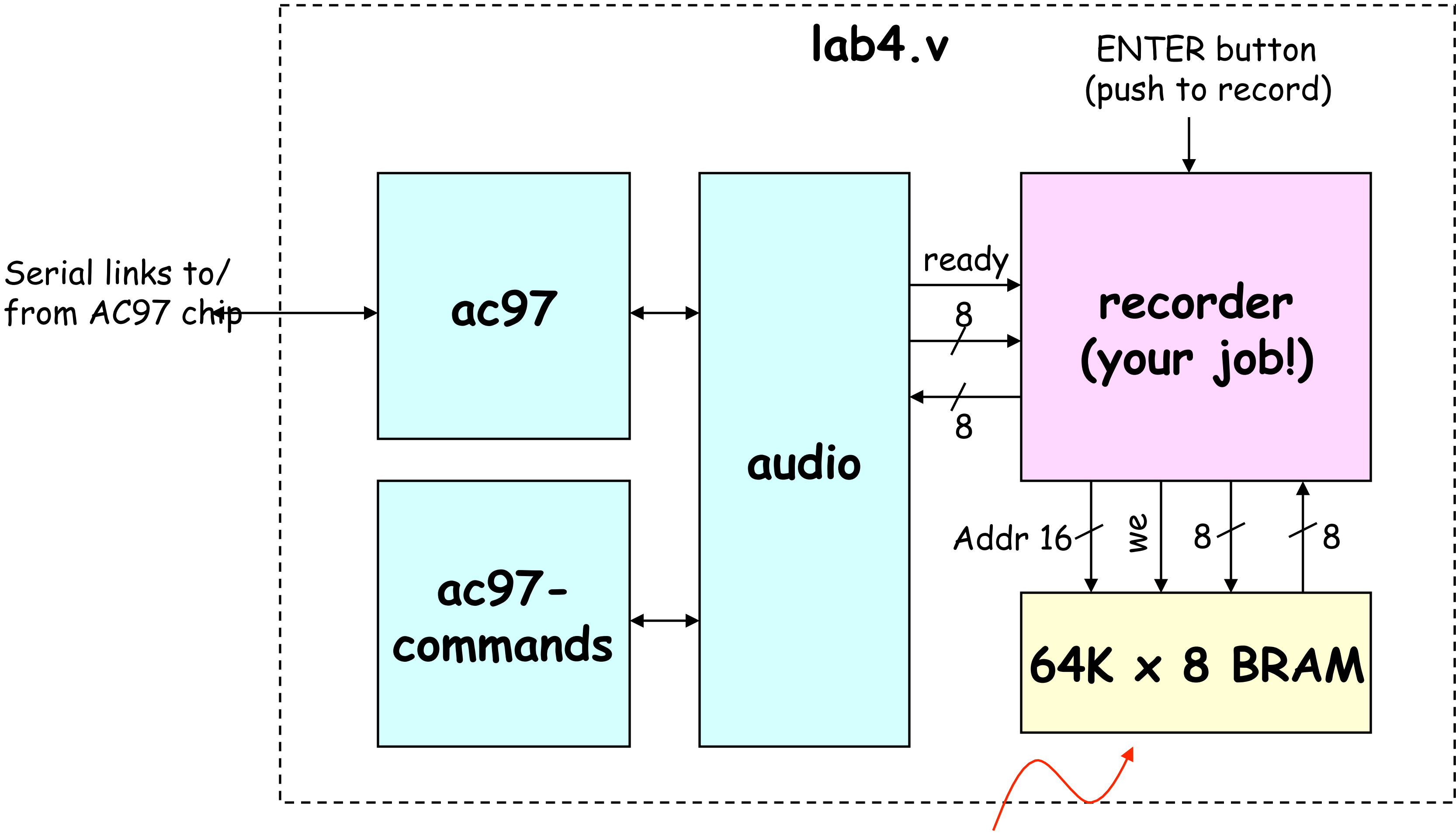
First: increment offset, then store incoming sample at location it points to. Clear sum.

Then: for i from 0 to N, compute sample[offset-i] * coeff[i] and add to register.

Finally: result in sum

Lab 4 overview

Assignment: build a voice recorder that records and plays back 8-bit PCM data @ 6KHz

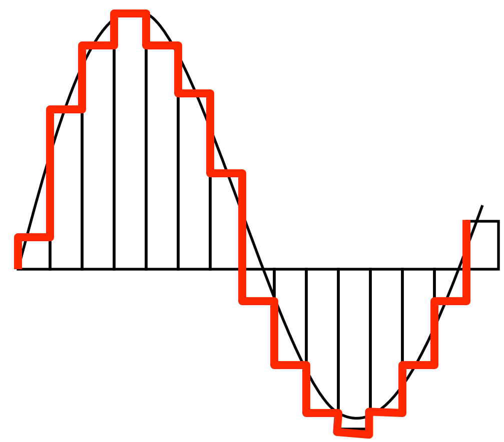


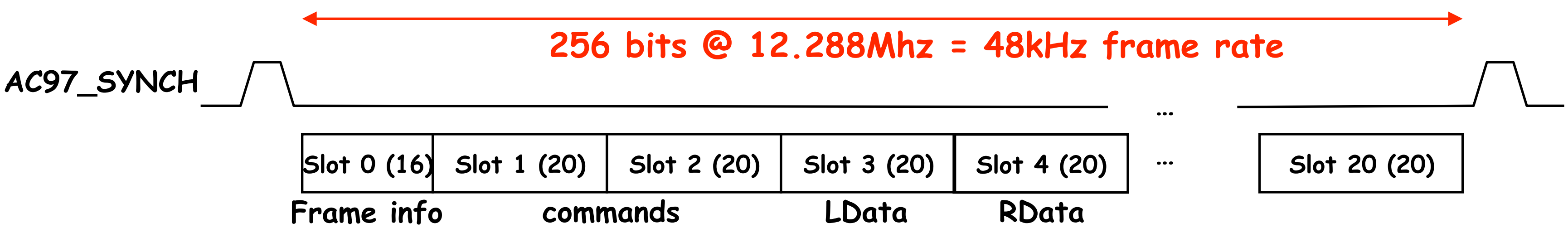
About 11 seconds of speech @ 6KHz

AC97: PCM data

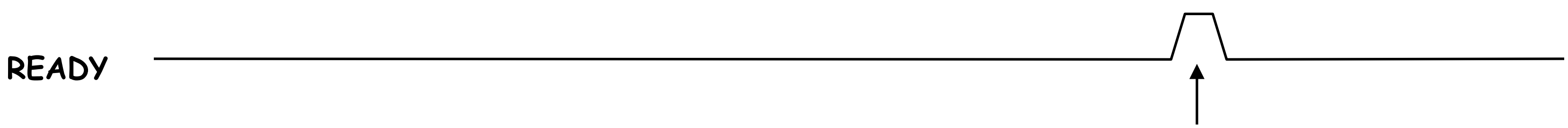
PCM = pulse code modulation

Sample waveform at 48kHz, encode results as an N-bit signed number. For our AC97 chip, N = 18.





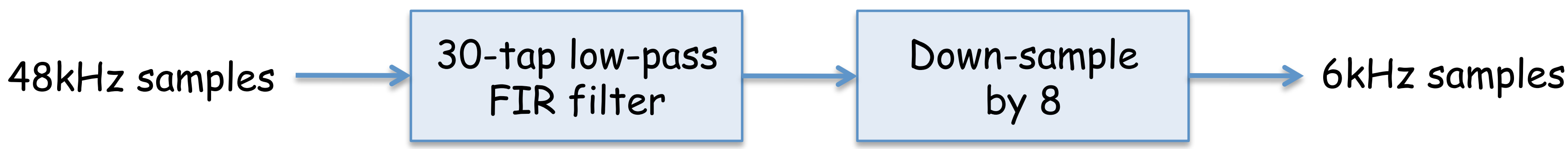
FPGA sends output frame to AC97 while AC97 sends input frame to FPGA



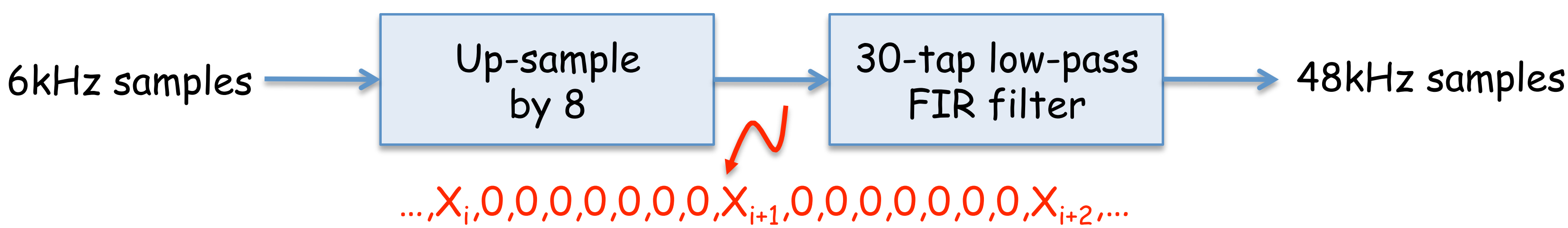
ready selects a particular *clock_27mhz* clock edge when you should store input data from the AC97 (*from_ac97_data*) and provide new output to the AC97 (*to_ac97_data*).

Lab 4 w/ FIR filter

- Since we're down-sampling by a factor of 8, to avoid aliasing (makes the recording sound "scratchy") we need to pass the incoming samples through a low-pass antialiasing filter to remove audio signal above 3kHz (Nyquist frequency of a 6kHz sample rate).



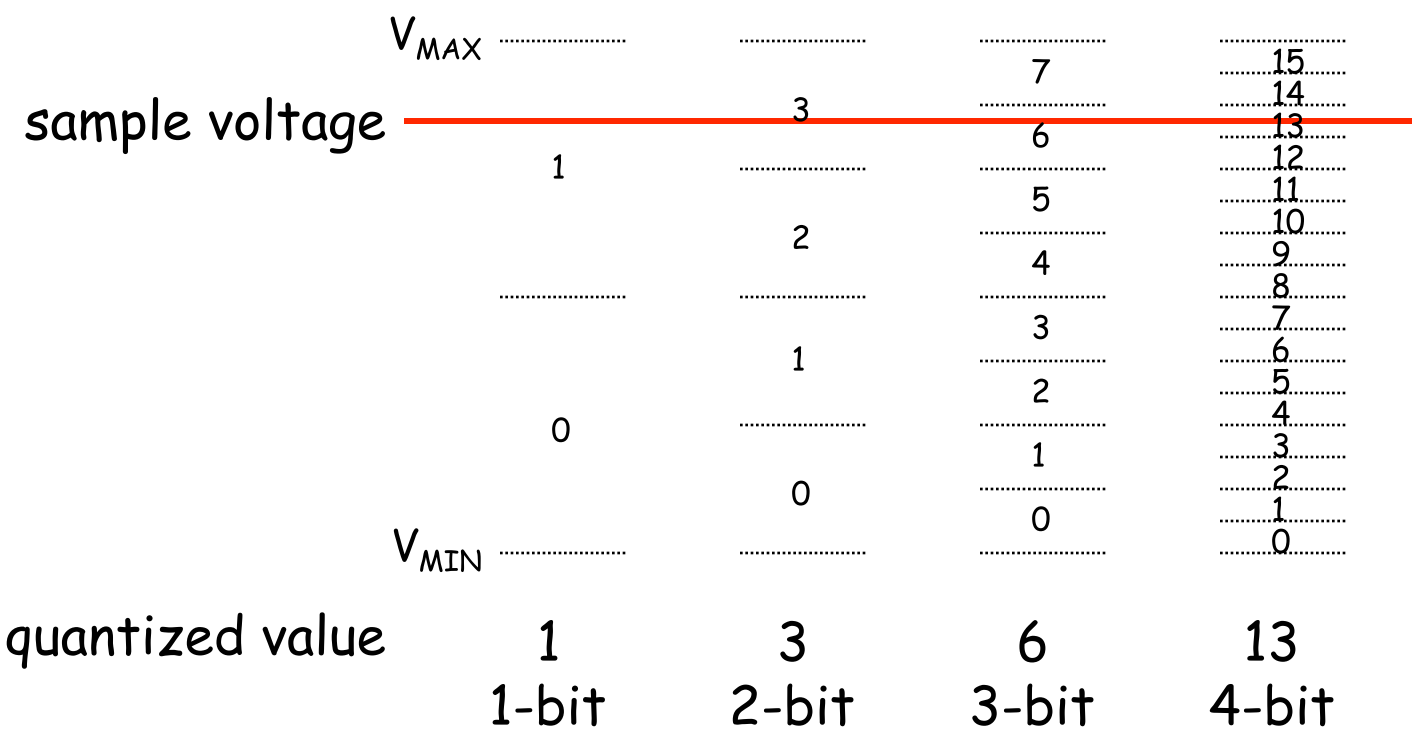
- We need a low-pass reconstruction filter (the same filter as for antialiasing!) when playing back the 6kHz samples. Actually we'll run it at 48kHz and achieve a 6kHz playback rate by feeding it a sample, 7 zeros, the next sample, 7 more zeros, etc.



Discrete Values

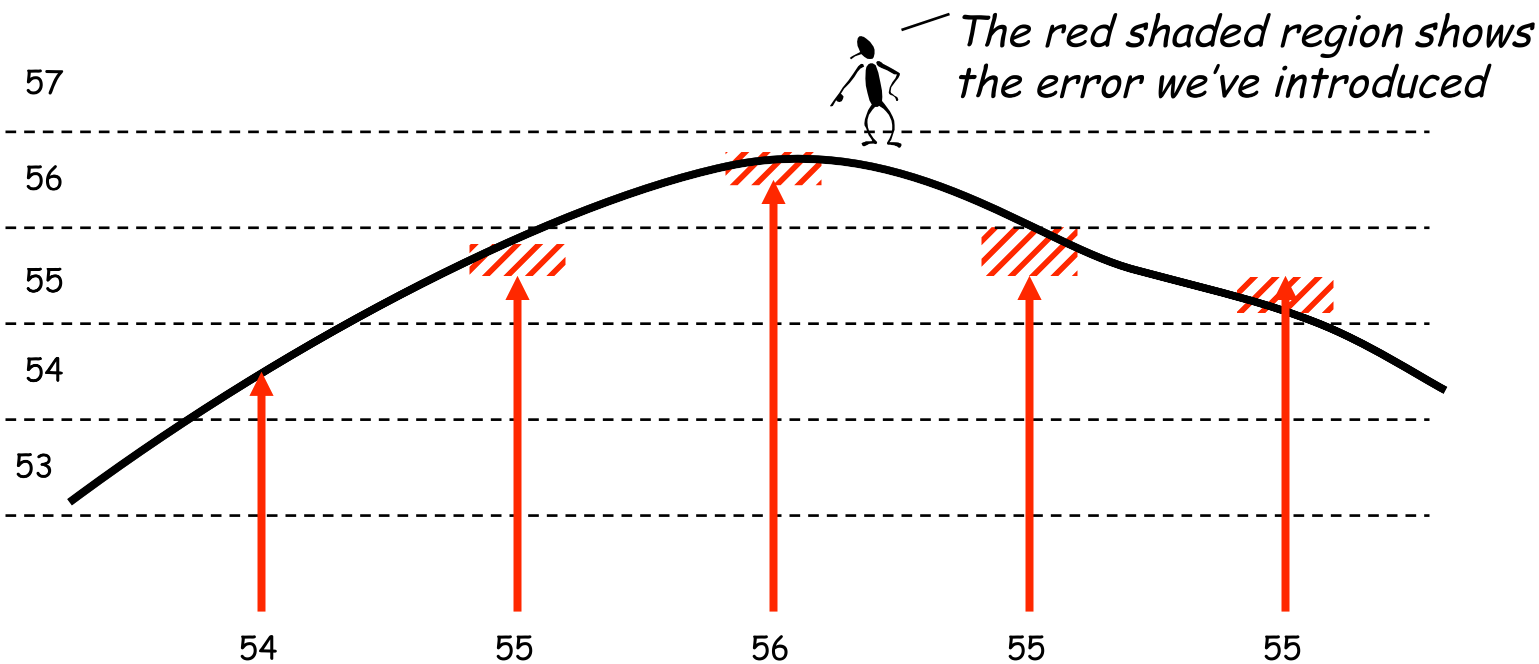
If we use N bits to encode the magnitude of one of the discrete-time samples, we can capture 2^N possible values.

So we'll divide up the range of possible sample values into 2^N intervals and choose the index of the enclosing interval as the encoding for the sample value.

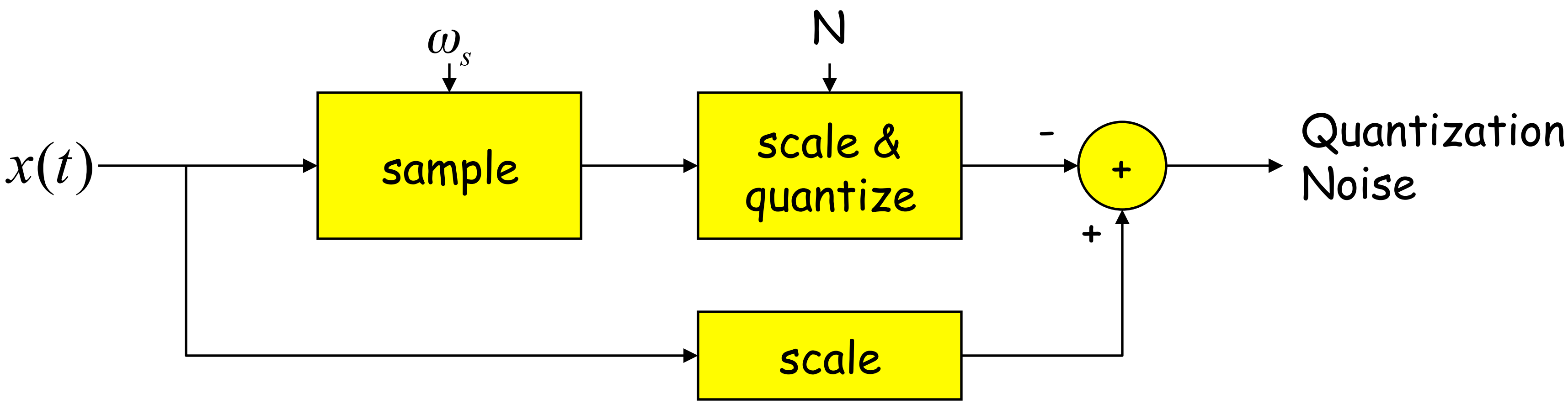


Quantization Error

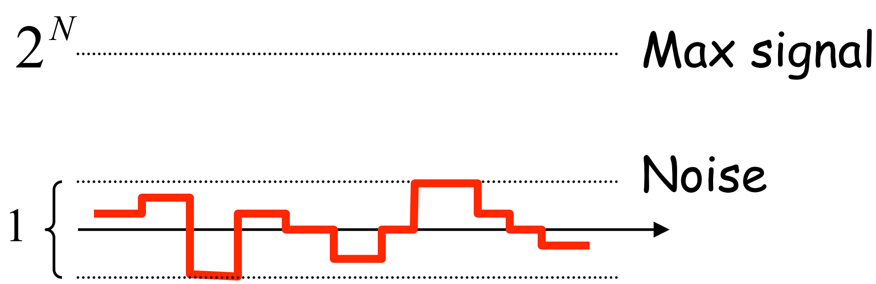
Note that when we quantize the scaled sample values we may be off by up to $\pm\frac{1}{2}$ step from the true sampled values.



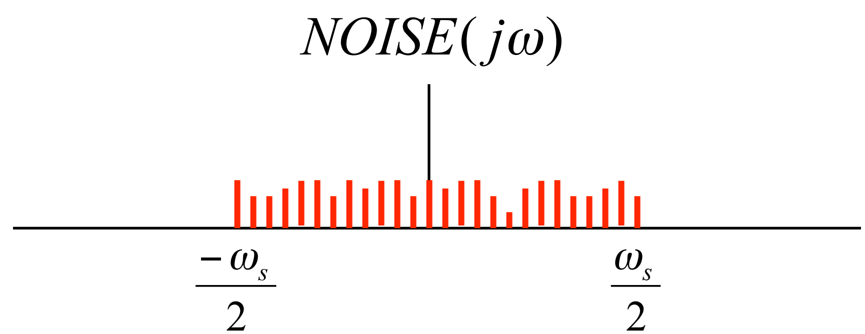
Quantization Noise



Time Domain



Freq. Domain



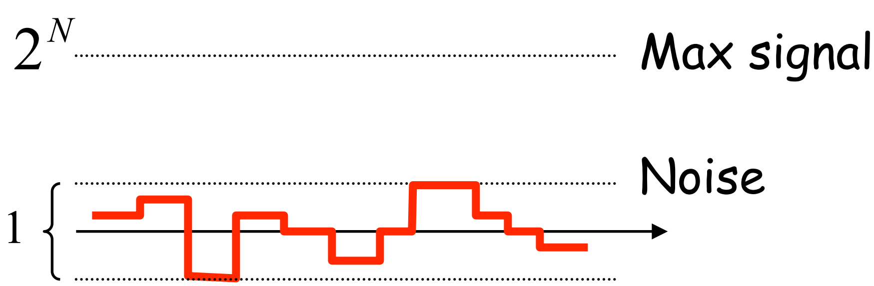
In most cases it's "white noise" with a uniform frequency distribution

SNR: Signal-to-Noise Ratio

$$SNR = 10\log_{10}\left(\frac{P_{SIGNAL}}{P_{NOISE}}\right) = 10\log_{10}\left(\frac{A_{SIGNAL}^2}{A_{NOISE}^2}\right) = 20\log_{10}\left(\frac{A_{SIGNAL}}{A_{NOISE}}\right)$$

RMS amplitude

SNR is measured in decibels (dB). Note that it's a logarithmic scale: if SNR increases by 3dB the ratio has increased by a factor 2. When applied to audible sounds: the ratio of normal speech levels to the faintest audible sound is 60-70 dB.



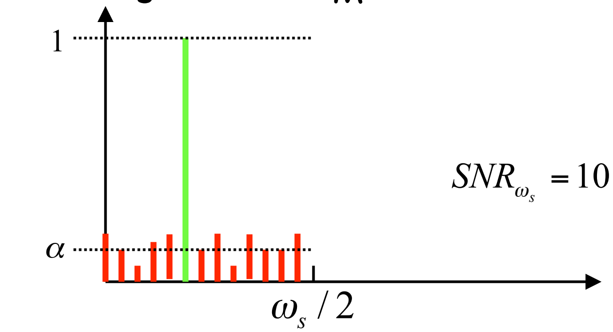
$$SNR = 20\log_{10}\left(\frac{A_{signal}}{A_{noise}}\right) \approx 20\log_{10}(2^N)$$

$$\approx N \cdot 6.02dB$$

Oversampling

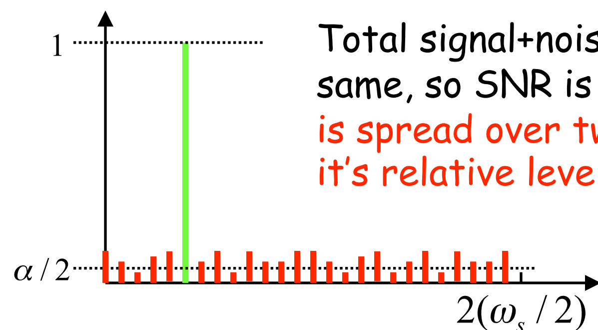
To avoid aliasing we know that ω_s must be at least $2\omega_M$. Is there any advantage to oversampling, i.e., $\omega_s = K \cdot 2\omega_M$?

Suppose we look at the frequency spectrum of quantized samples of a sine wave: (sample freq. = ω_s)



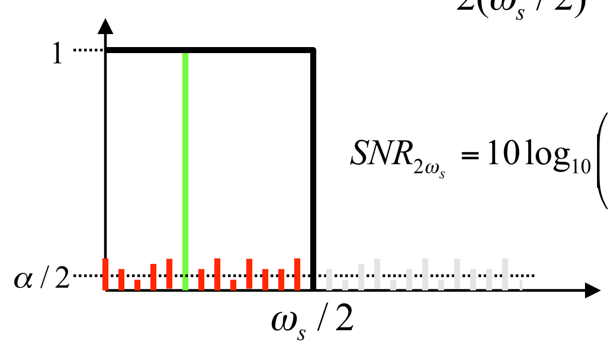
$$SNR_{\omega_s} = 10\log_{10}\left(\frac{P_{SIGNAL}}{P_{NOISE}}\right)$$

Let's double the sample frequency to $2\omega_s$.



Total signal+noise power remains the same, so SNR is unchanged. But noise is spread over twice the freq. range so it's relative level has dropped.

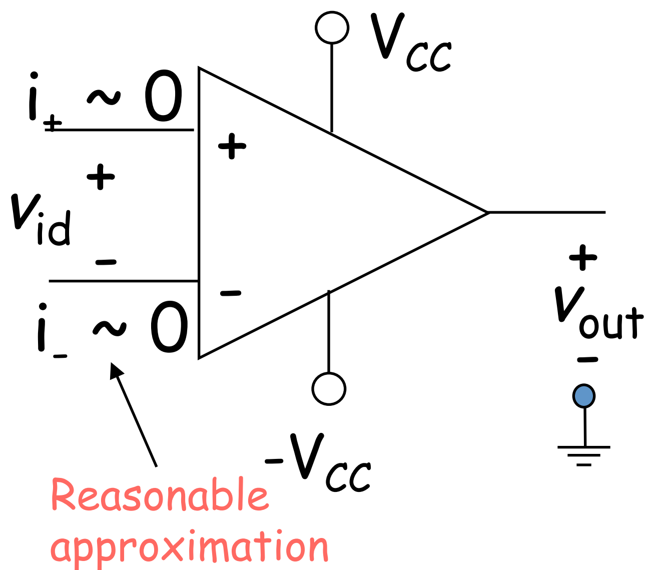
Now let's use a low pass filter to eliminate half the noise! Note that we're not affecting the signal at all…

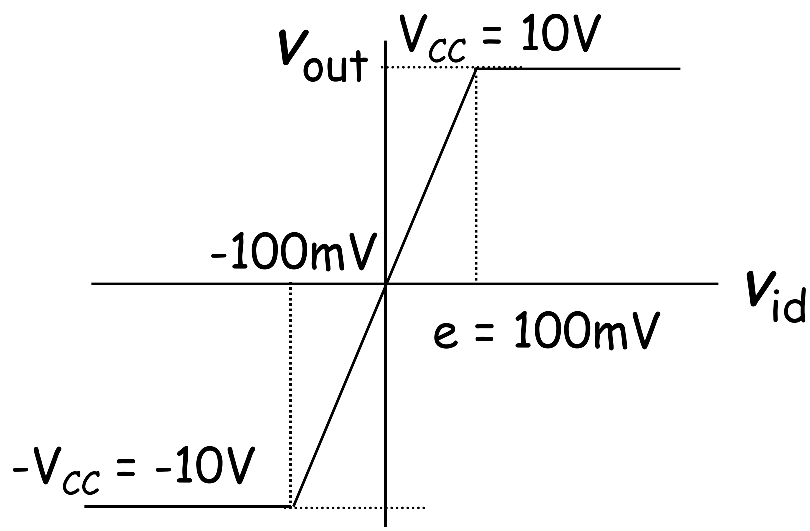


$$SNR_{2\omega_s} = 10\log_{10}\left(\frac{P_{SIGNAL}}{P_{NOISE}/2}\right) = SNR_{\omega_s} + 3dB$$

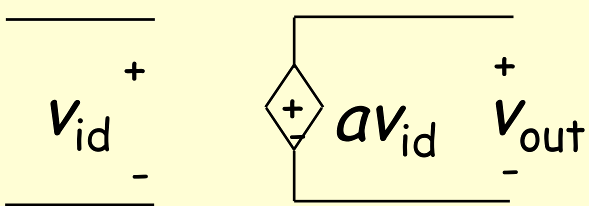
Oversampling+LPF reduces noise by 3dB/octave

Our Analog Building Block: OpAmp



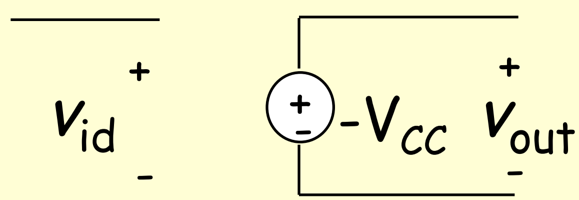


Linear Mode



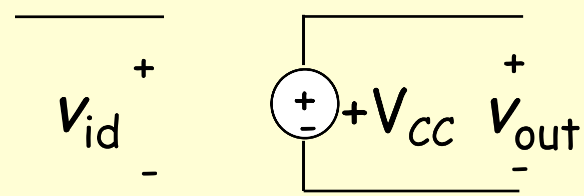
If $-V_{CC} < v_{out} < V_{CC}$

Negative Saturation



$v_{id} < -e$

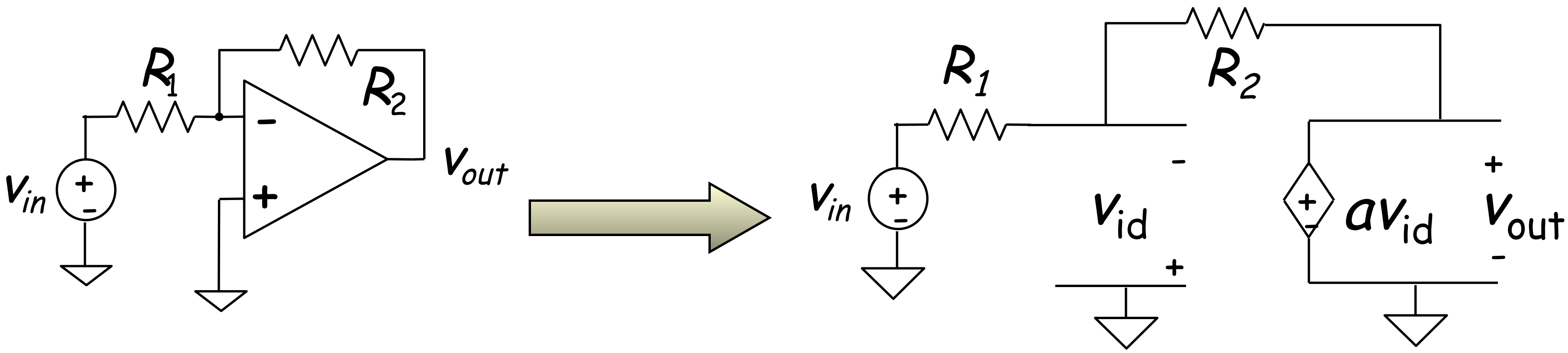
Positive Saturation



$v_{id} > e$

Very small input range for "open loop" configuration

The Power of (Negative) Feedback



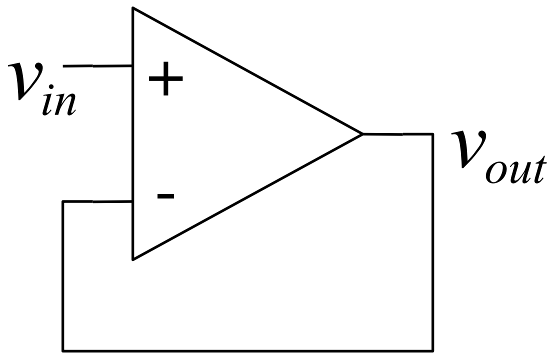
$$\frac{v_{in} + v_{id}}{R_1} + \frac{v_{out} + v_{id}}{R_2} = 0 \qquad v_{id} = \frac{v_{out}}{a} \qquad \frac{v_{in}}{R_1} = -\frac{v_{out}}{a}\left[\frac{1}{R_1} + \frac{a}{R_2} + \frac{1}{R_2}\right]$$

$$\boxed{\frac{v_{out}}{v_{in}} = -\frac{R_2 a}{(1+a)R_1 + R_2} \approx -\frac{R_2}{R_1} \left(if \quad a >> 1\right)}$$

- Overall (closed loop) gain does not depend on open loop gain
- Trade gain for robustness
- Easier analysis approach: "virtual short circuit approach"
  - $v_+ = v_- = 0$ if OpAmp is linear

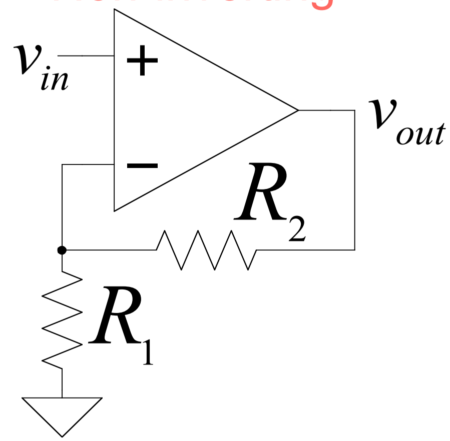
Basic OpAmp Circuits

Voltage Follower (buffer)



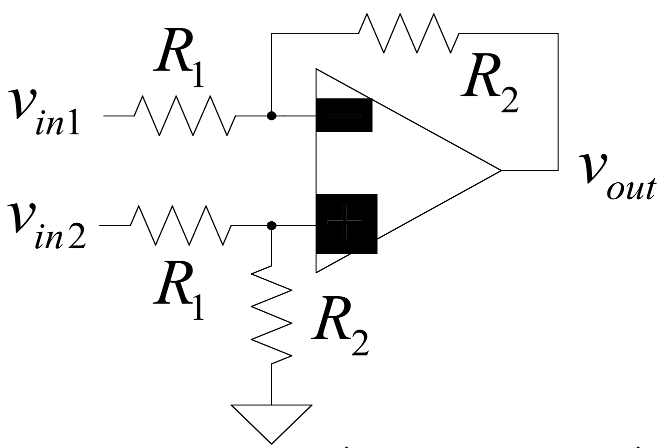
$$v_{out} \approx v_{in}$$

Non-inverting



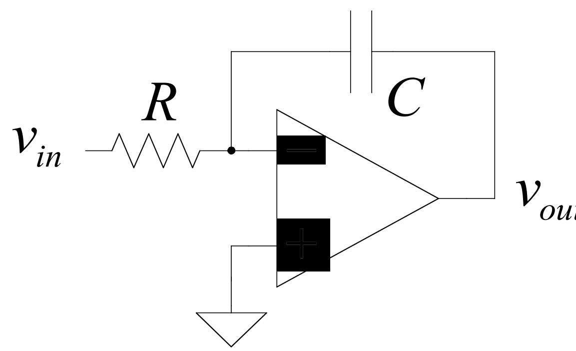
$$v_{out} \approx \frac{R_1 + R_2}{R_1} v_{in}$$

Differential Input



$$v_{out} \approx \frac{R_2}{R_1}\left(v_{in2} - v_{in1}\right)$$

Integrator

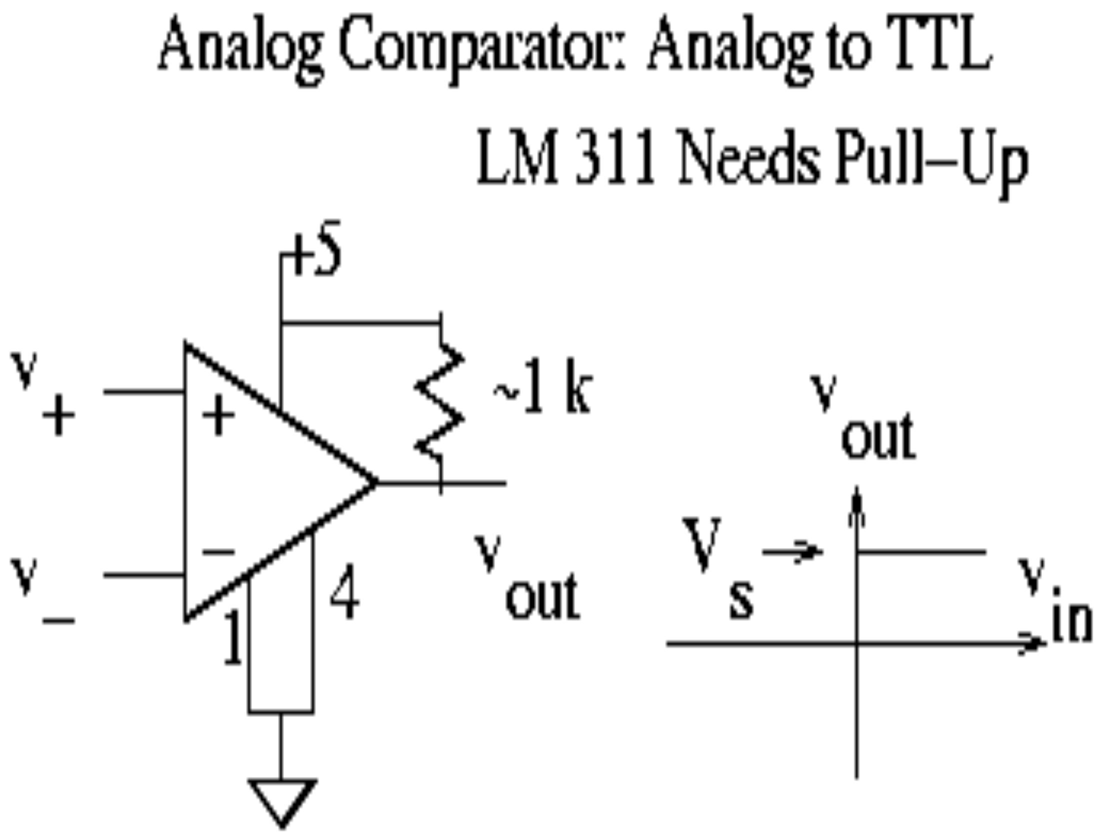


$$v_{out} \approx -\frac{1}{RC}\int_{-\infty}^{t} v_{in}\,dt$$

OpAmp as a Comparator

Analog Comparator:

Is V+ > V- ? The Output is a DIGITAL signal



LM311 uses a single supply voltage

Digital to Analog

- Common metrics:
  - Conversion rate – DC to ~500 MHz (video)
  - # bits – up to ~24
  - Voltage reference source (internal / external; stability)
  - Output drive (unipolar / bipolar / current) & settling time
  - Interface – parallel / serial
  - Power dissipation

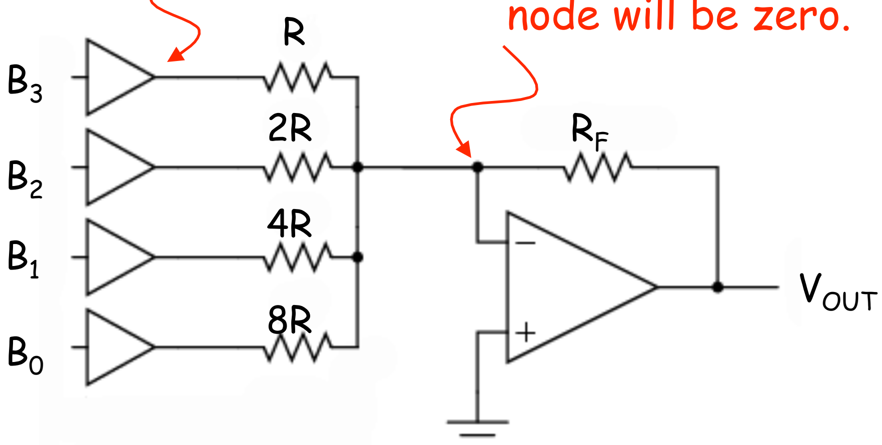
- Common applications:
  - Real world control (motors, lights)
  - Video signal generation
  - Audio / RF "direct digital synthesis"
  - Telecommunications (light modulation)
  - Scientific & Medical (ultrasound, …)

DAC: digital to analog converter

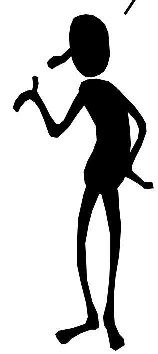
How can we convert a N-bit binary number to a voltage?

V_i = 0 volts if B_i = 0
V_i = V volts if B_i = 1

OPAMP will vary V_{OUT} to maintain this node at 0V, i.e., the sum of the currents flowing into this node will be zero.



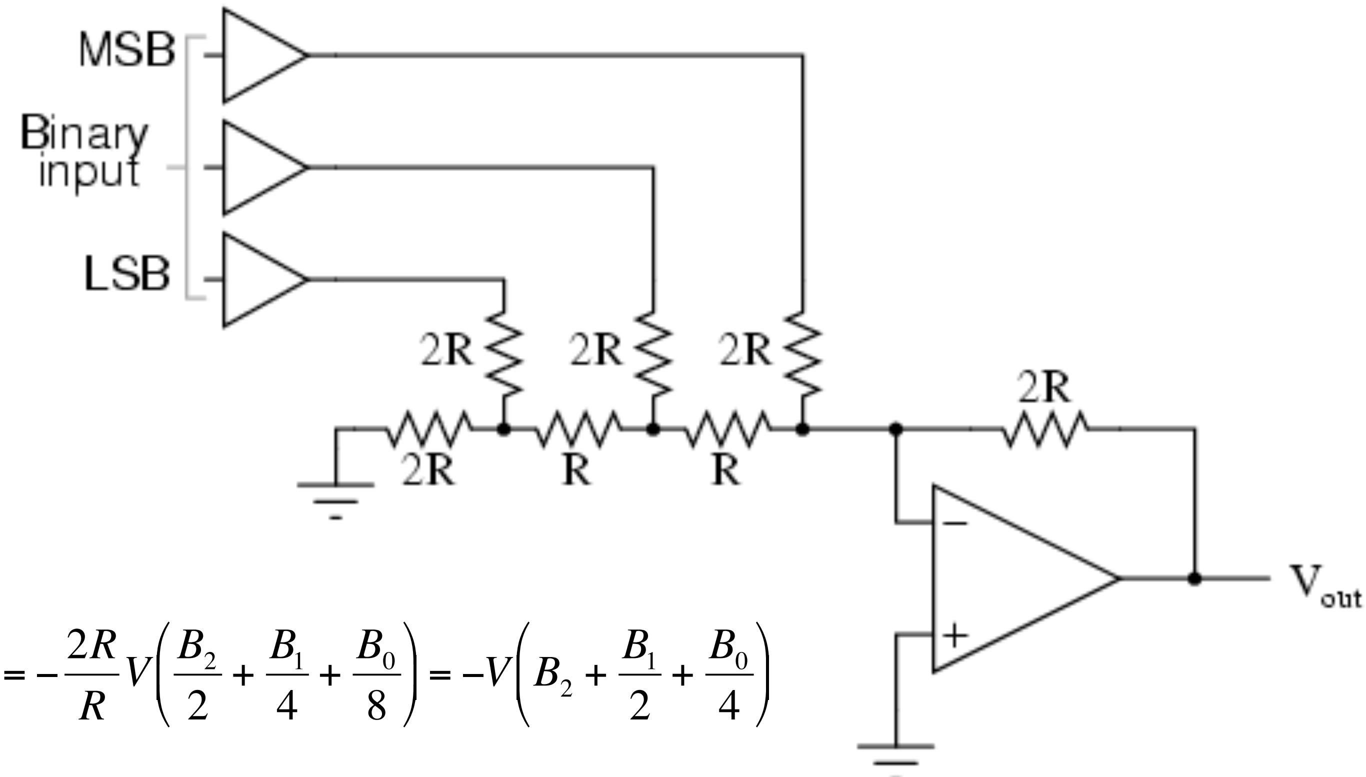
OKAY, this'll work, but the voltages produced by the drivers and various R's must be carefully matched in order to get equal steps.



$$\frac{V_{OUT}}{R_F} + \frac{B_3 V}{R} + \frac{B_2 V}{2R} + \frac{B_1 V}{4R} + \frac{B_0 V}{8R} = 0$$

$$V_{OUT} = -\frac{R_F}{R} V \left(B_3 + \frac{B_2}{2} + \frac{B_1}{4} + \frac{B_0}{8} \right)$$

R-2R Ladder DAC Architecture

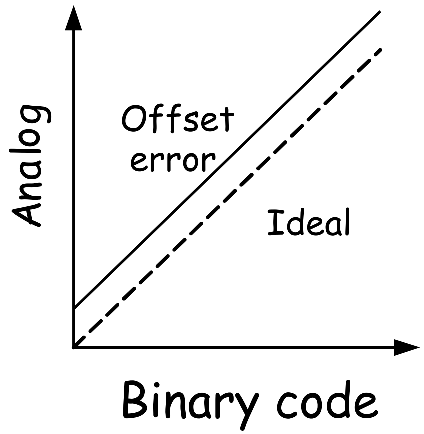


$$V_{OUT} = -\frac{2R}{R}V\left(\frac{B_2}{2}+\frac{B_1}{4}+\frac{B_0}{8}\right) = -V\left(B_2+\frac{B_1}{2}+\frac{B_0}{4}\right)$$

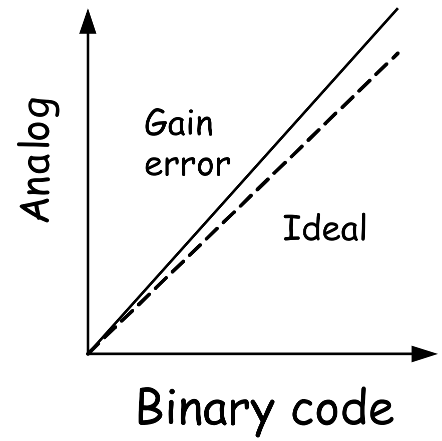
R-2R Ladder achieves large current division ratios with only two resistor values

Non-idealities in Data Conversion

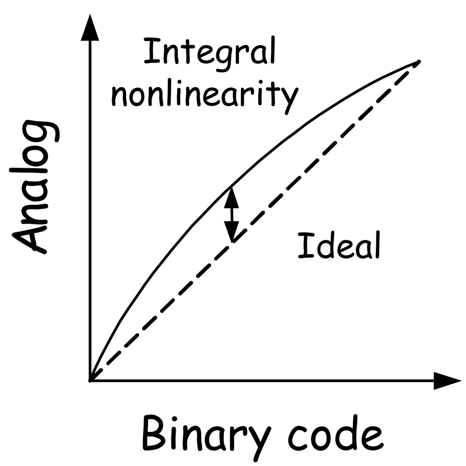
Offset – a constant voltage offset that appears at the output when the digital input is 0



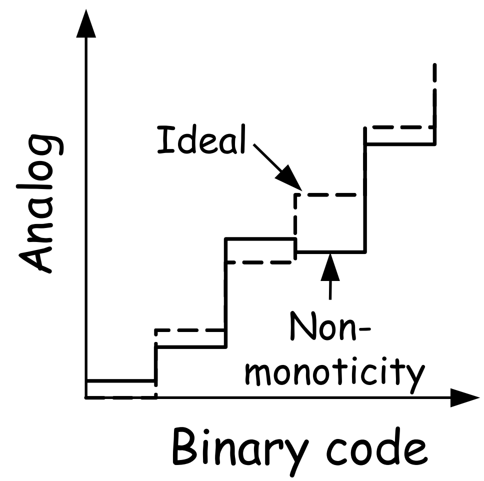
Gain error – deviation of slope from ideal value of 1



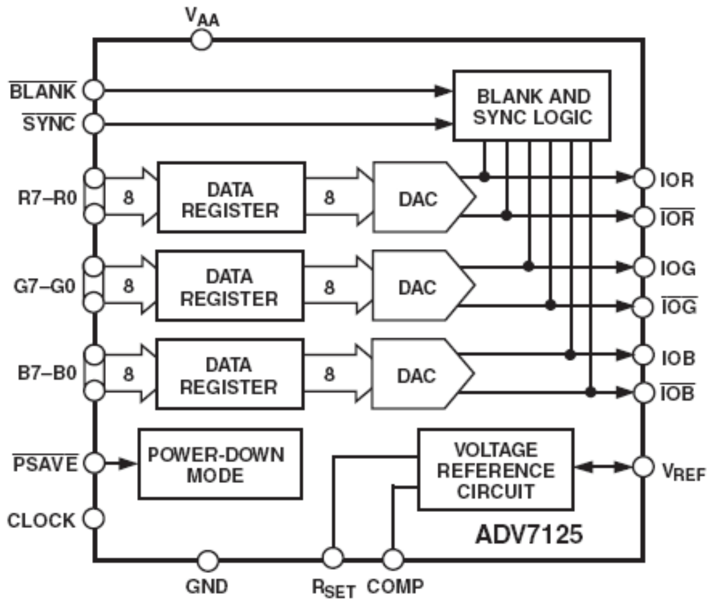
Integral Nonlinearity – maximum deviation from the ideal analog output voltage



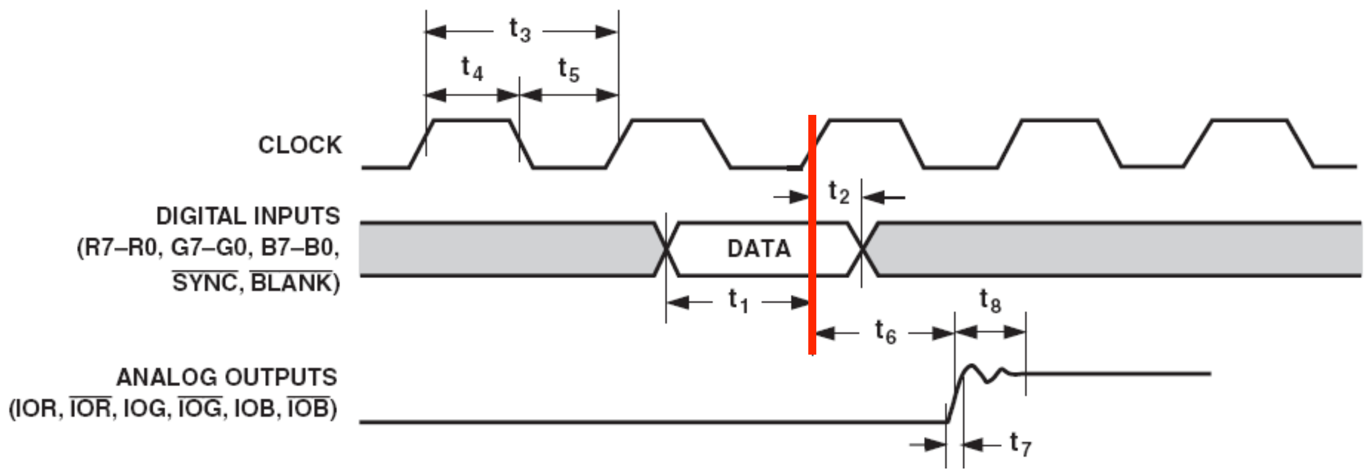
Differential nonlinearity – the largest increment in analog output for a 1-bit change



Labkit: ADV7125 Triple Out Video DAC



- Three 8-bit DACs
- Single Supply Op.: 3.3 to 5V
- Internal bandgap voltage ref
- Output: 2-26 mA
- 330 MSPS (million samples per second)
- Simple edge-triggered register -based interface

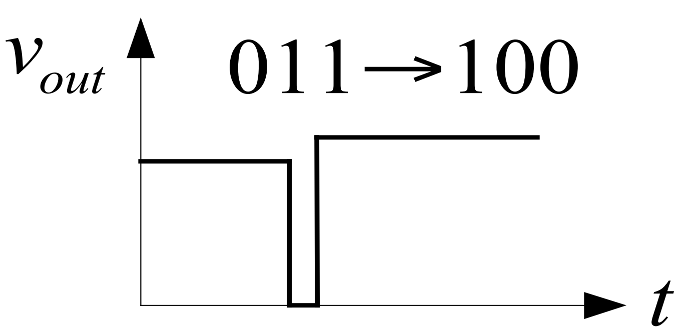


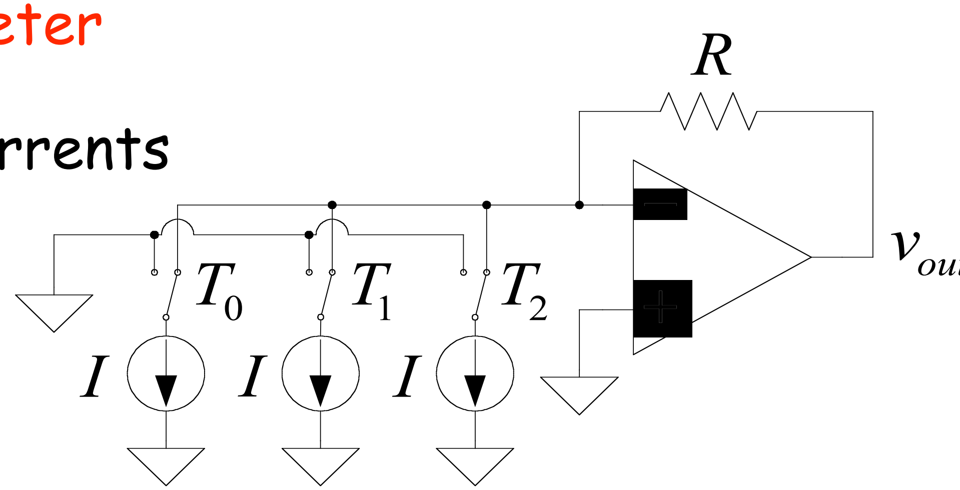
Glitching and Thermometer D/A

- Glitching is caused when switching times in a D/A are not synchronized
- Example: Output changes from 011 to 100 – MSB switch is delayed

Binary		Thermometer		
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1

- Filtering reduces glitch but increases the D/A settling time
- One solution is a thermometer code D/A – requires $2^N - 1$ switches but no ratioed currents

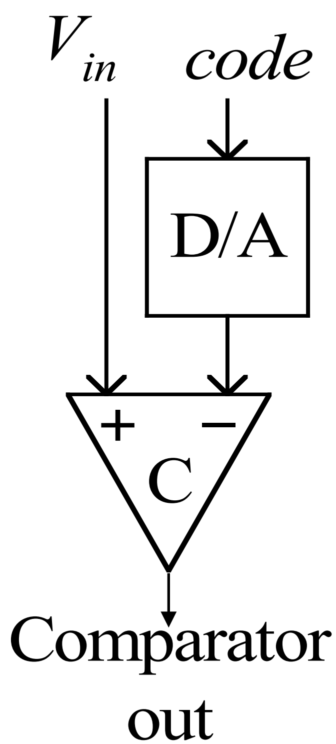


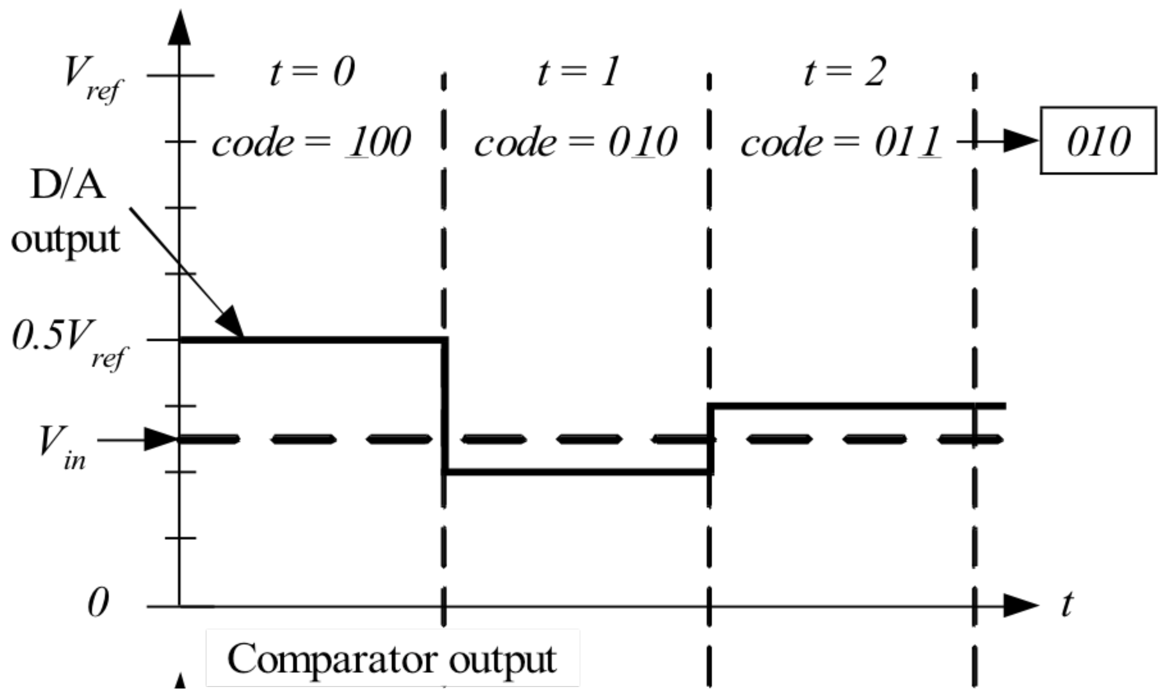


$$v_{out} = -IR(T_0 + T_1 + T_2)$$

Successive-Approximation A/D

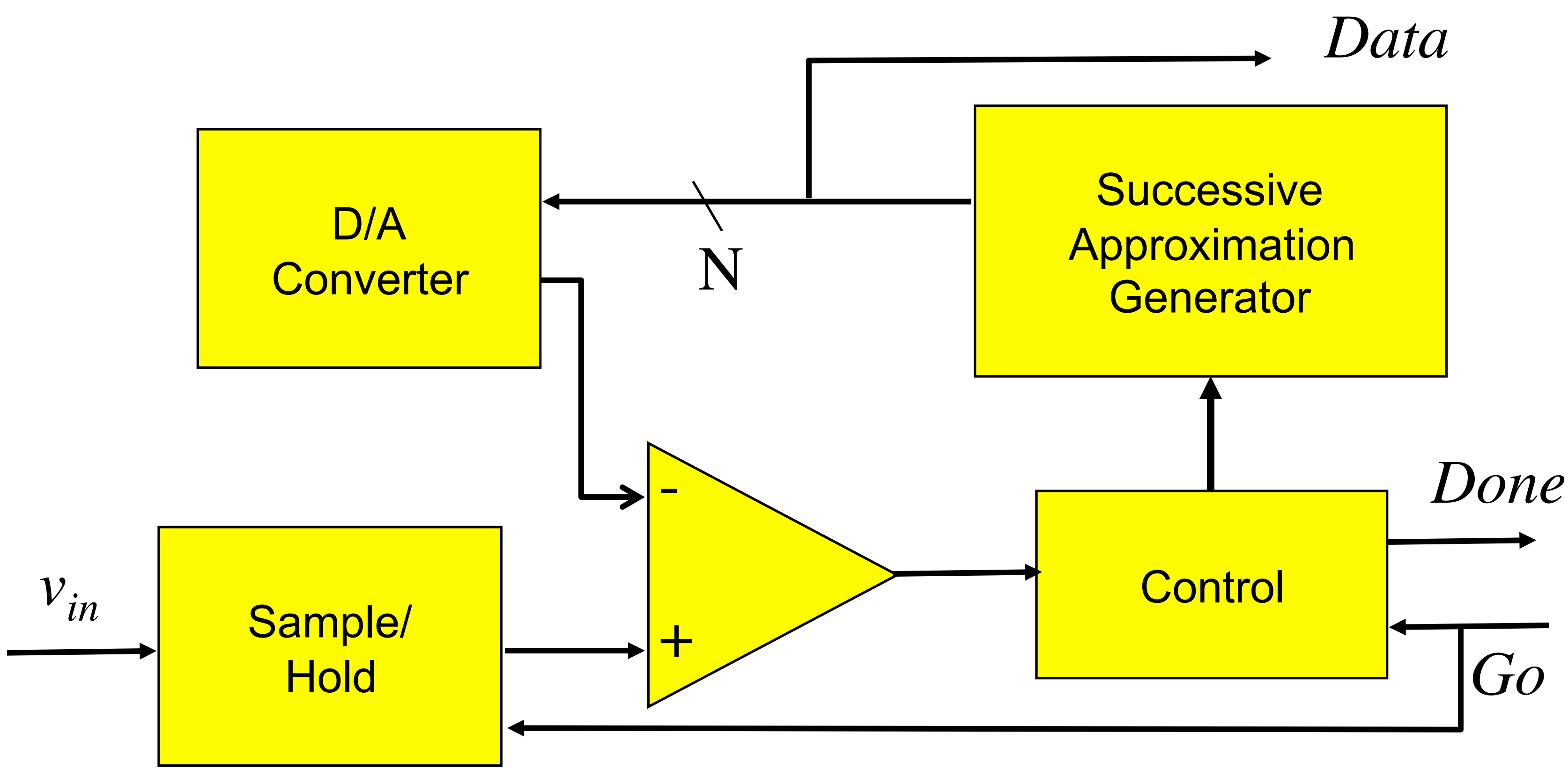
- D/A converters are typically compact and easier to design. Why not A/D convert using a D/A converter and a comparator?
- DAC generates analog voltage which is compared to the input voltage
- If DAC voltage > input voltage then set that bit; otherwise, reset that bit
- This type of ADC takes a fixed amount of time proportional to the bit length





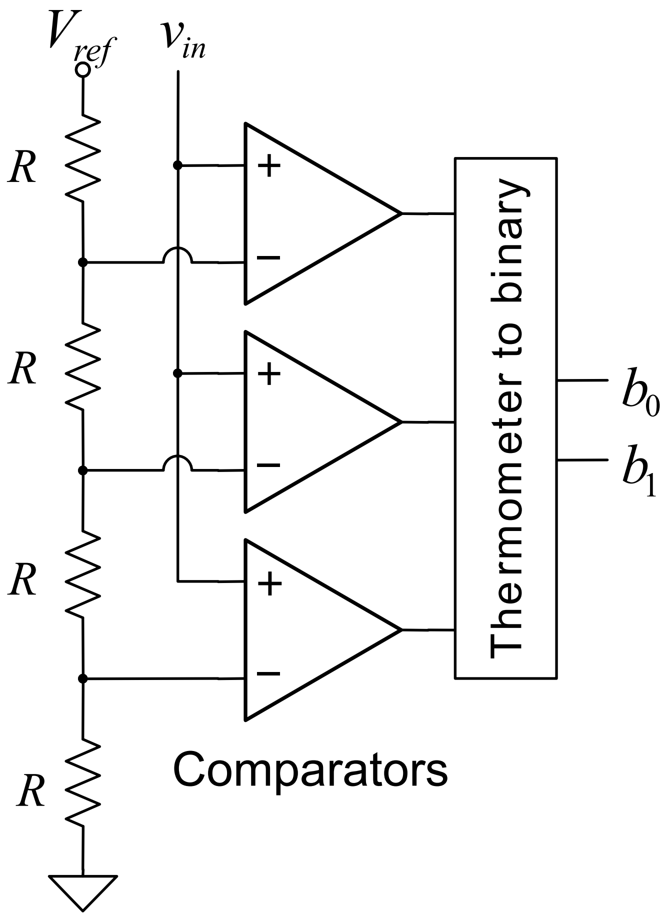
Example: 3-bit A/D conversion, 2 LSB < V_{in} < 3 LSB

Successive-Approximation A/D



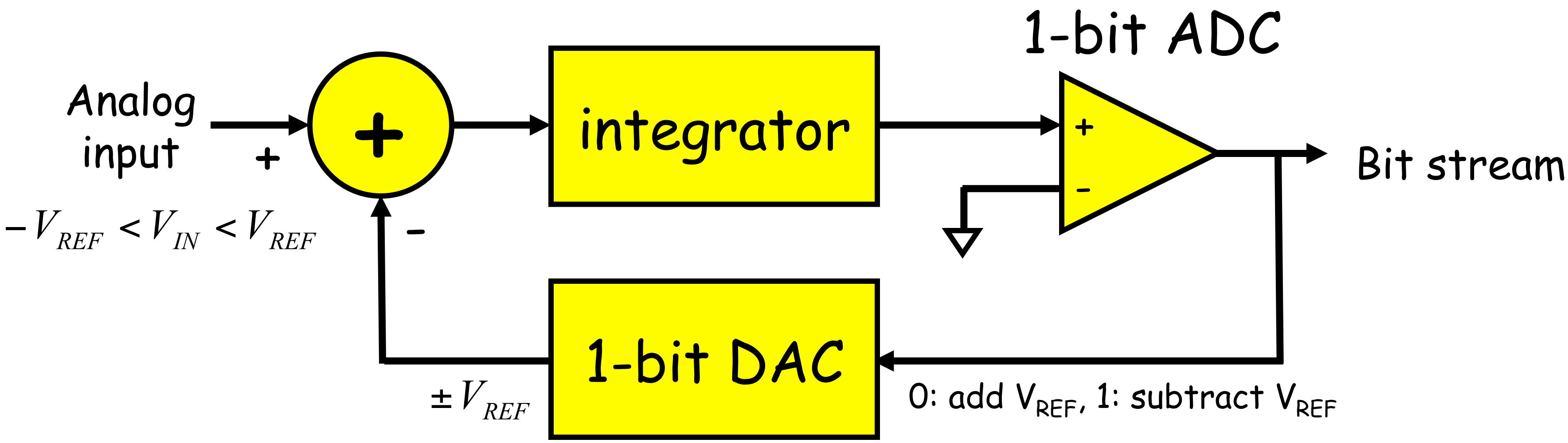
Serial conversion takes a time equal to $N(t_{D/A} + t_{comp})$

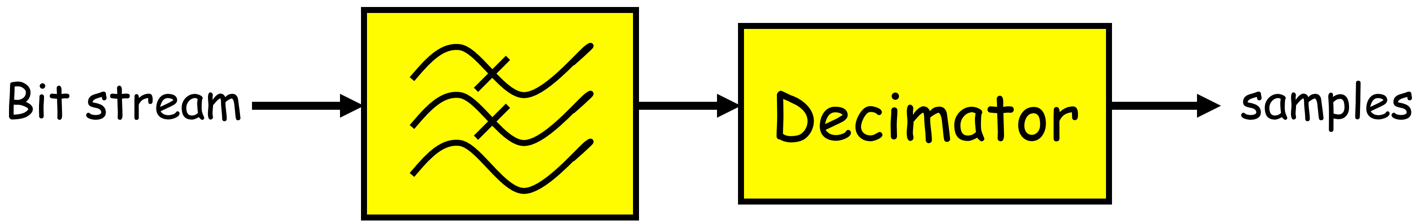
Flash A/D Converter



- Brute-force A/D conversion
- Simultaneously compare the analog value with every possible reference value
- Fastest method of A/D conversion
- Size scales exponentially with precision
(requires 2^N comparators)

Sigma Delta ADC





Computes average, produces N-bit result

Only need to keep enough samples to meet Nyquist rate

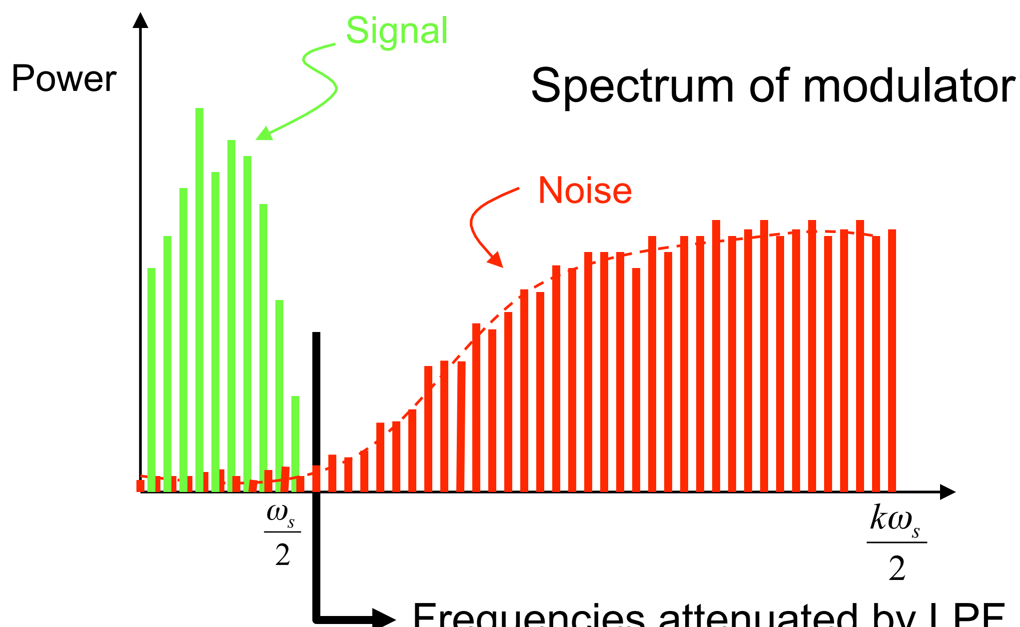
Average of bit stream ($1=V_{REF}$, $0=-V_{REF}$) gives voltage

With V_{REF}=1V: V_{IN}=0.5: 1110…, V_{IN}=-0.25: 00100101…, V_{IN}=0.6: 11110

http://www.analog.com/Analog_Root/static/techSupport/designTools/interactiveTools/sdtutorial/sdtutorial.html

So, what's the big deal?

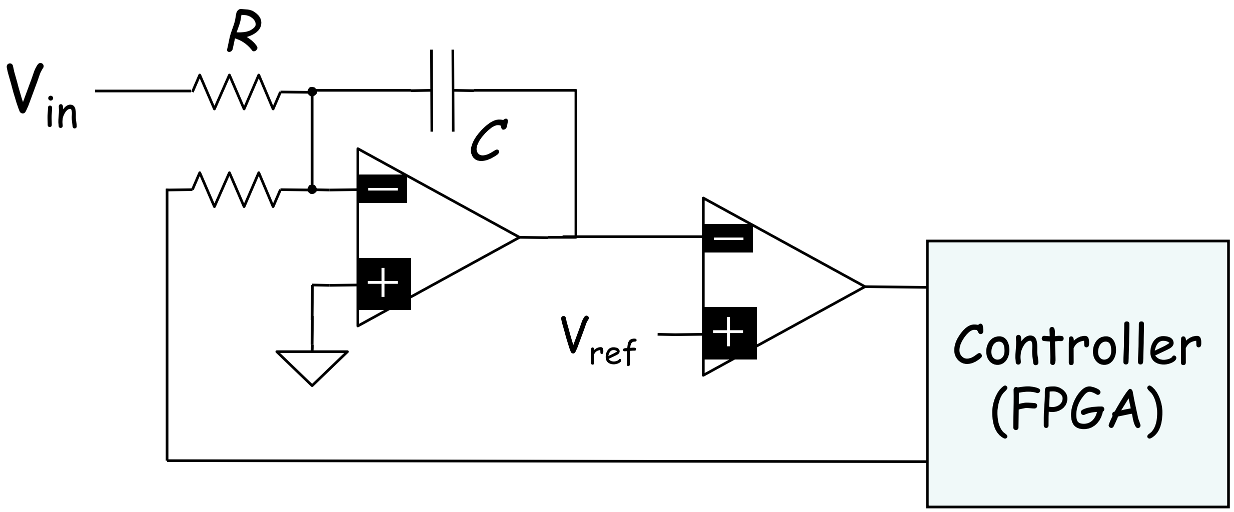
- Can be run at high sampling rates, oversampling by, say, 8 or 9 octaves for audio applications; low power implementations
- Feedback path through the integrator changes how the noise is spread across the sampling spectrum.



- Pushing noise power to higher frequencies means more noise is eliminated by LPF: N^{th} order ΣΔ SNR = (3+N*6)dB/octave

Sigma Delta ADC

- A simple ADC:



- Poor Man's ADC: