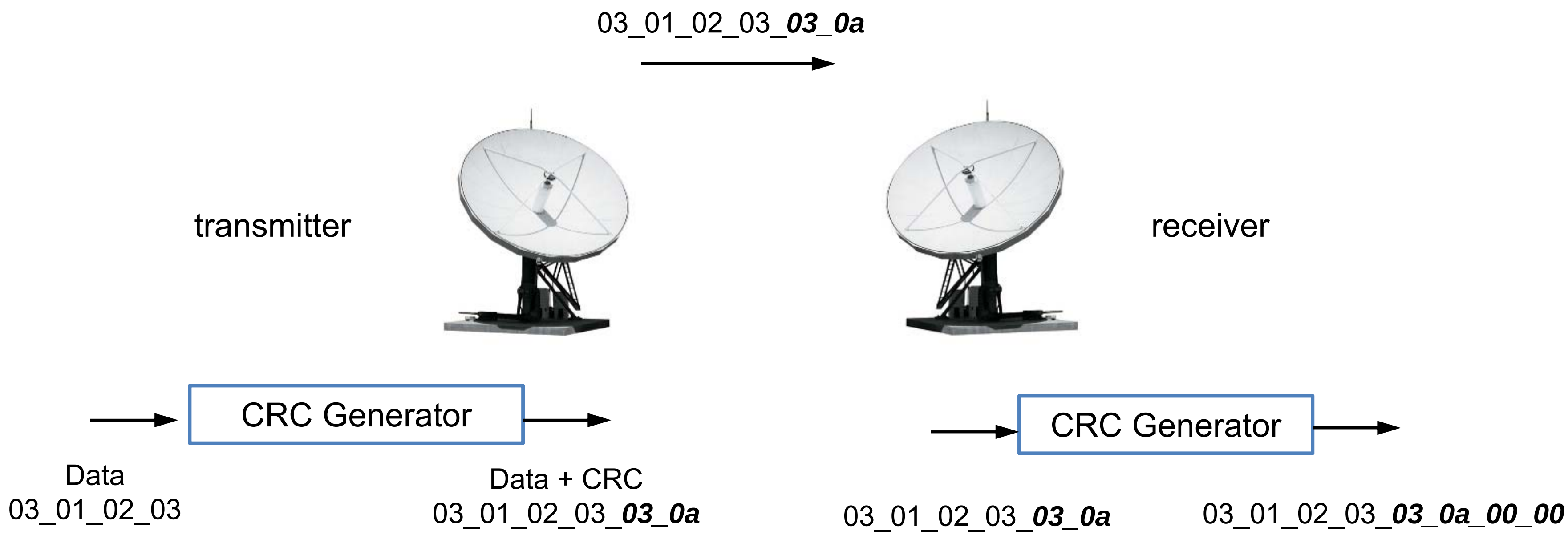
Analog Building Blocks

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

Handouts
- lecture slides,
- Lpset 8

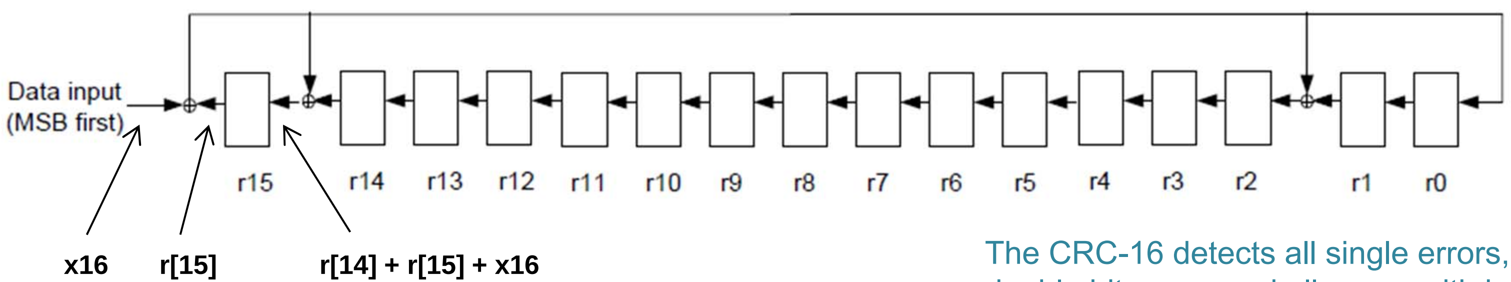
Thu/Fri: Lab 4 Checkoff
Mon: email project teams

CRC


03_01_02_03_***03_0a***
transmitter
receiver
CRC Generator
Data
03_01_02_03
Data + CRC
03_01_02_03_***03_0a***
CRC Generator
03_01_02_03_***03_0a***
03_01_02_03_***03_0a_00_00***

Cyclic redundancy check - CRC

CRC16 (x16 + x15 + x2 + 1)



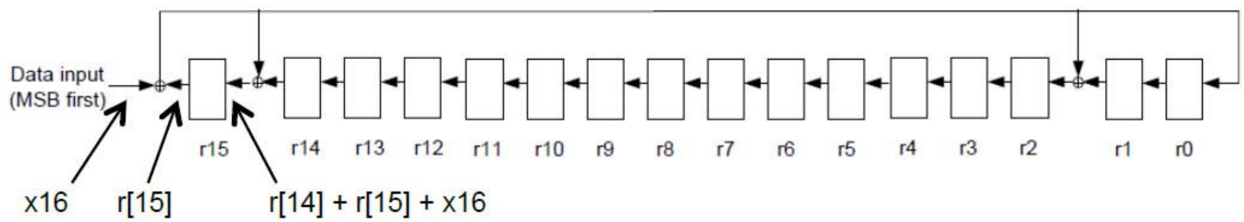
The CRC-16 detects all single errors, all double bit errors and all errors with burst less than 16 bits in length.

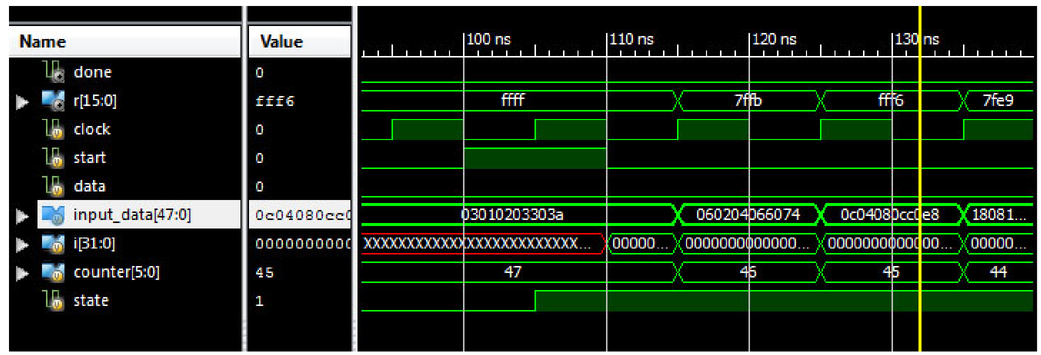
- Each “r” is a register, all clocked with a common clock. Common clock not shown
- As shown, for register r15, the output is r[15] and the input is the sum of r[14], r[15] and data input x16, etc
- The small round circles with the plus sign are adders implemented with XOR gates.
- Initialize r to 16’hFFFF at start

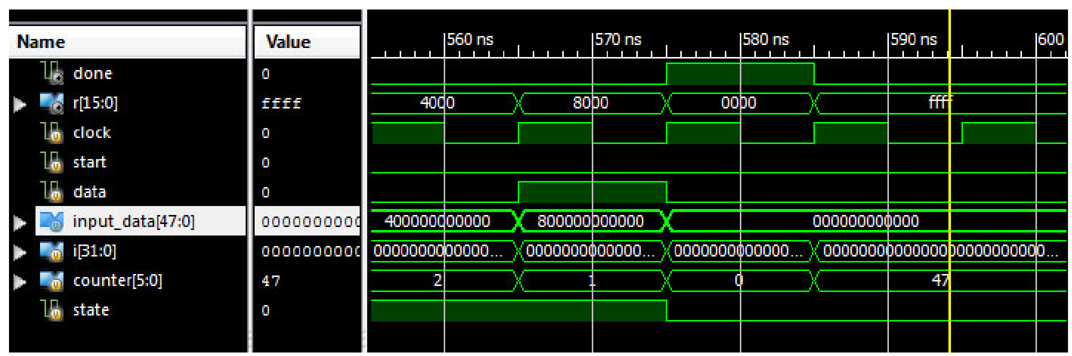
CRC Solution

CRC16: x16+x15+x2+1

```
module lpset6(
    input clock,
    input start,
    input data,
    output done,
    output reg [15:0] r
    );
    parameter IDLE=0;
    parameter CRC_CALC=1;

    wire x16 = data;
    reg state=0;
    reg [5:0] counter=0; //my counter

    always @ (posedge clock) begin
       case (state)
        IDLE: begin
          state <= (start) ? CRC_CALC : IDLE;
          r <= 16'hFFFF; //start reset FSM
          counter <= 47;
        end

        CRC_CALC: begin
          r[15] <= r[14] + r[15] + x16;
          r[14:3] <= r[13:2];
          r[2] <= r[1] + r[15] + x16;
          r[1] <= r[0];
          r[0] <= r[15] + x16;
          counter <= counter - 1;
          state <= (counter == 1)? IDLE:CRC_CALC;
         end

       endcase
      end

      assign done = (counter == 0);
endmodule
```

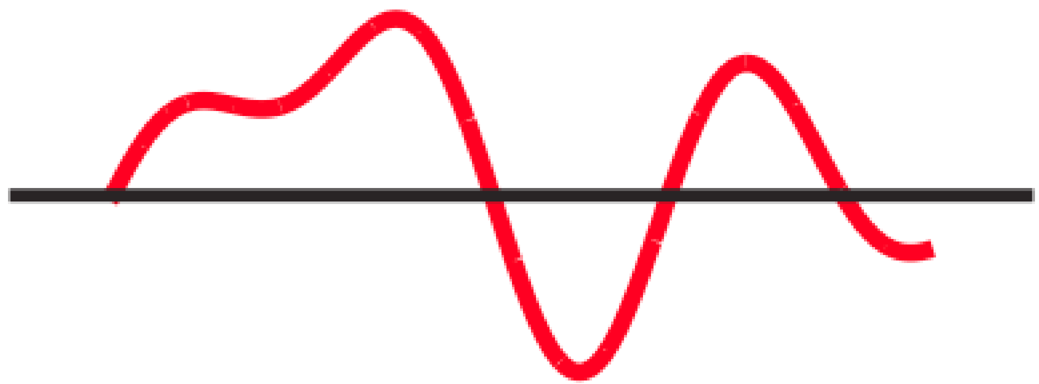




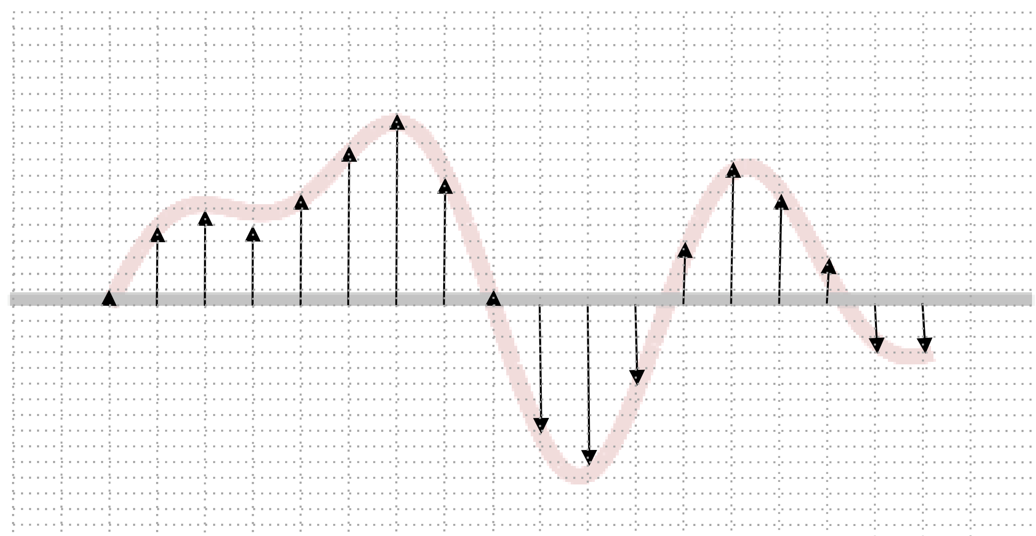


Digital Representations of Analog Waveforms

Continuous time
Continuous values



Discrete time
Discrete values

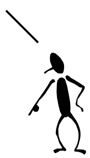


Discrete Time

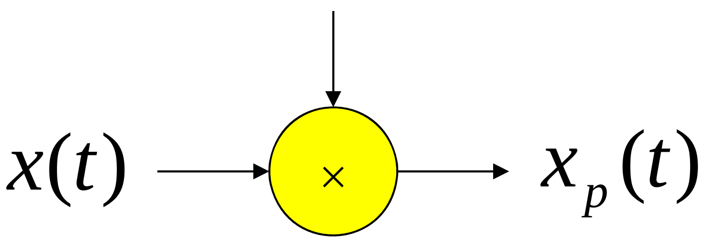
Let's use an impulse train to sample a continuous-time function at a regular interval T:

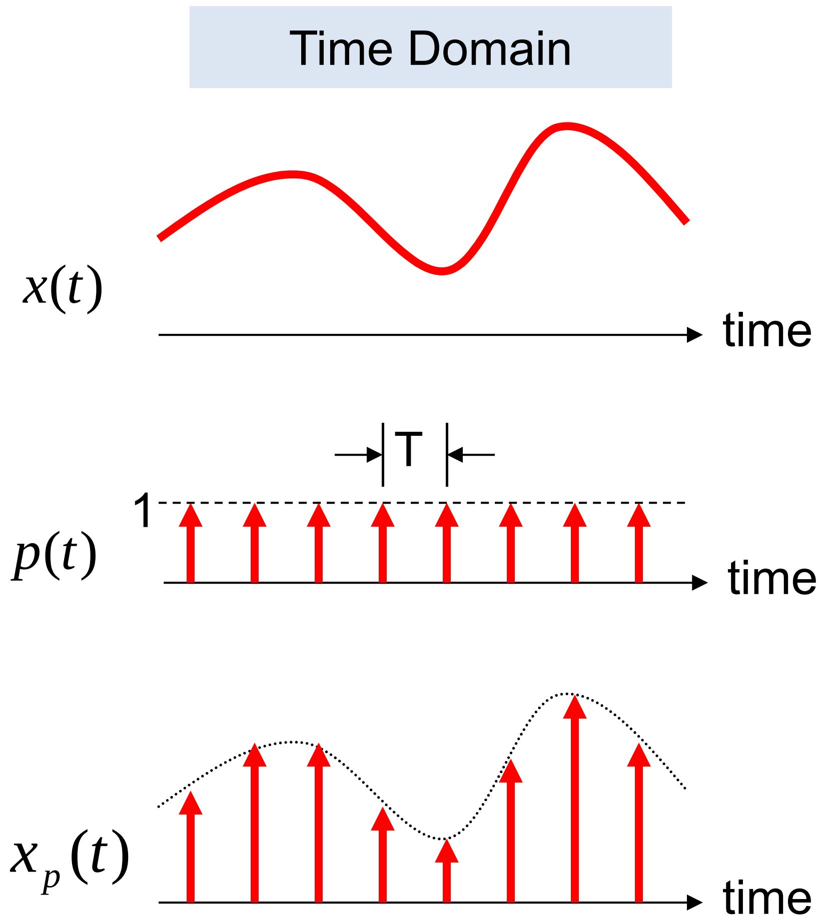
δ(x) is a narrow impulse at x=0, where

$$\int_{-\infty}^{\infty} f(t)\delta(t-a)\,dt = f(a)$$



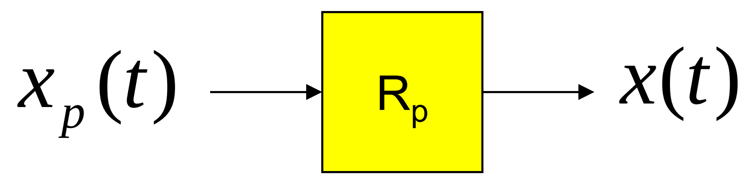
$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t-nT)$$





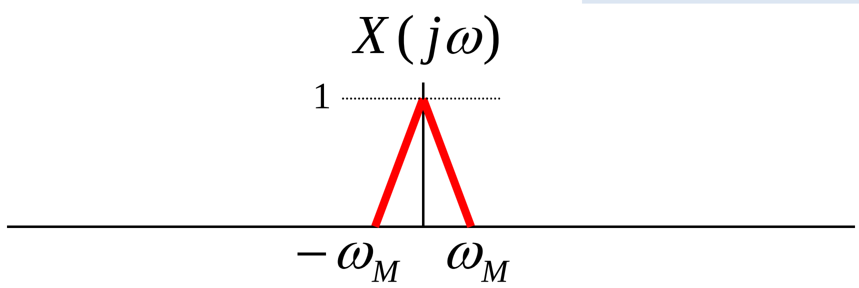
Reconstruction

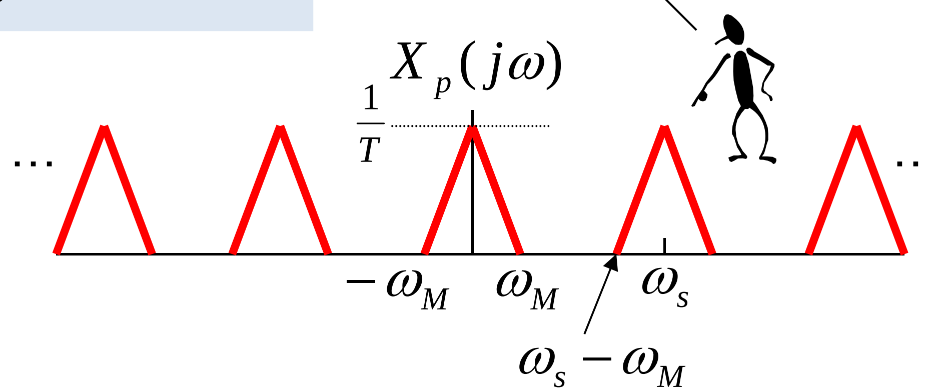
Is it possible to reconstruct the original waveform using only the discrete time samples?

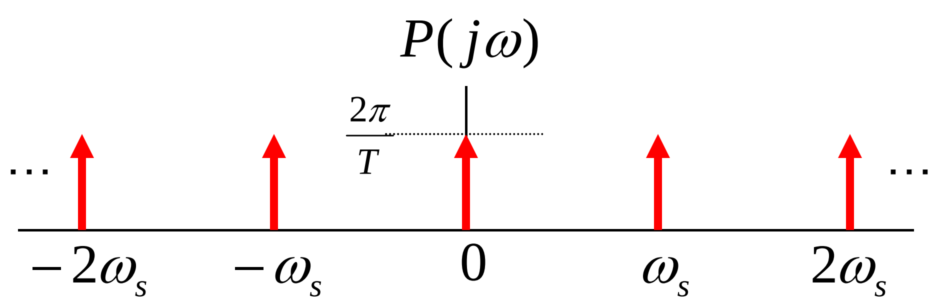


Frequency Domain

Looks like modulation by ω_s and its harmonics

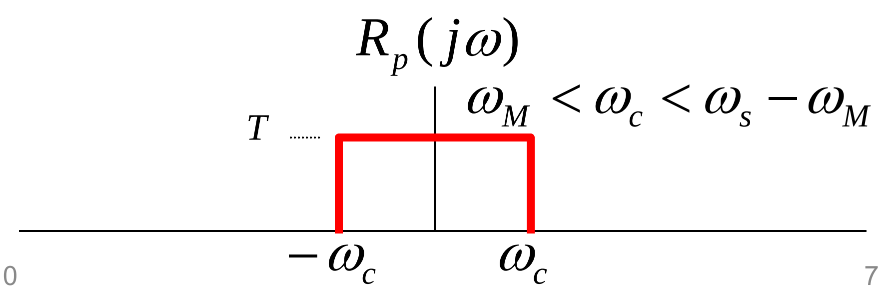






$$\omega_s = \frac{2\pi}{T}$$

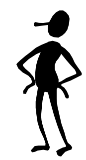
So, if $\omega_m < \omega_s - \omega_m$, we can recover the original waveform with a low-pass filter!



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

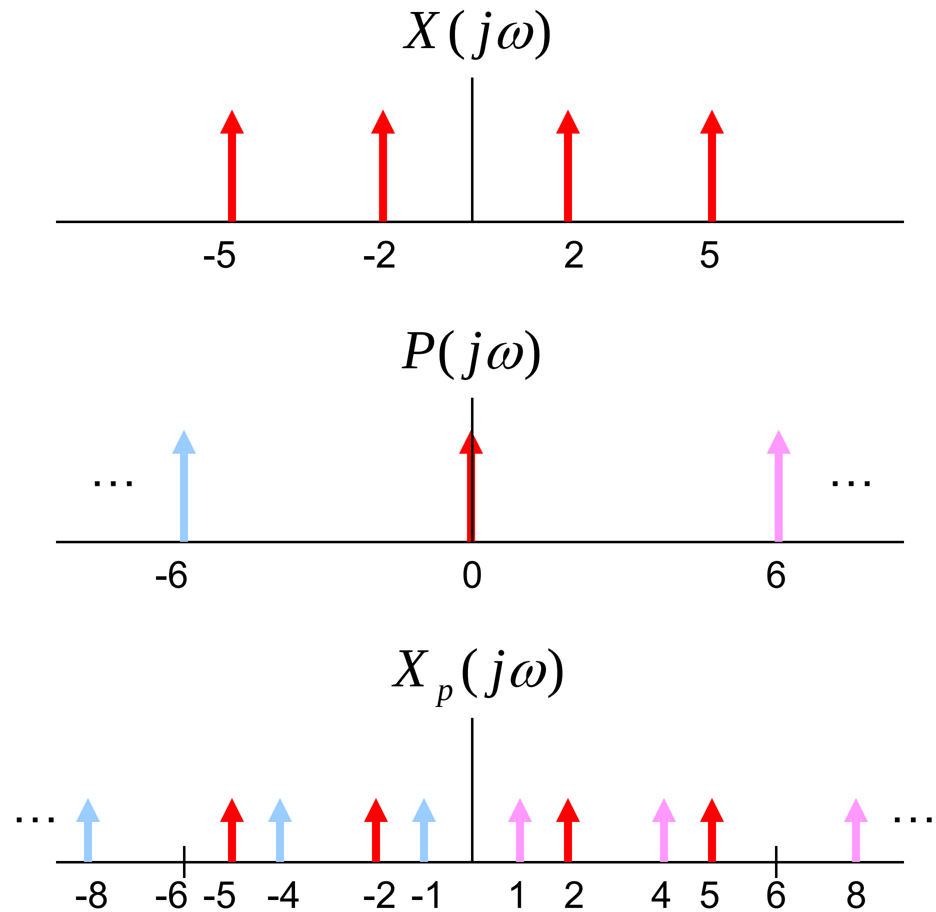
$$\omega_s = \frac{2\pi}{T}$$

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 ω_s-ω_M.

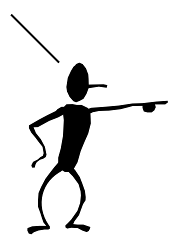
Undersampling $\rightarrow$ Aliasing

If $\omega_s \le 2\omega_M$ 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.

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



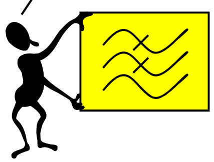
There are now tones at 1 (= 6 – 5) and 4 (= 6 – 2) in addition to the original tones at 2 and 5.

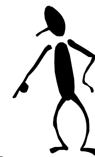


Antialias Filters

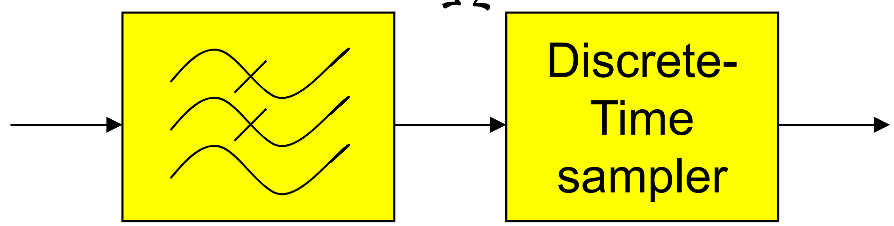
If we wish to create samples at some fixed frequency ω_s, then to avoid aliasing we need to use a low-pass filter on the original waveform to remove any frequency content ≥ $\omega_s/2$.

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…

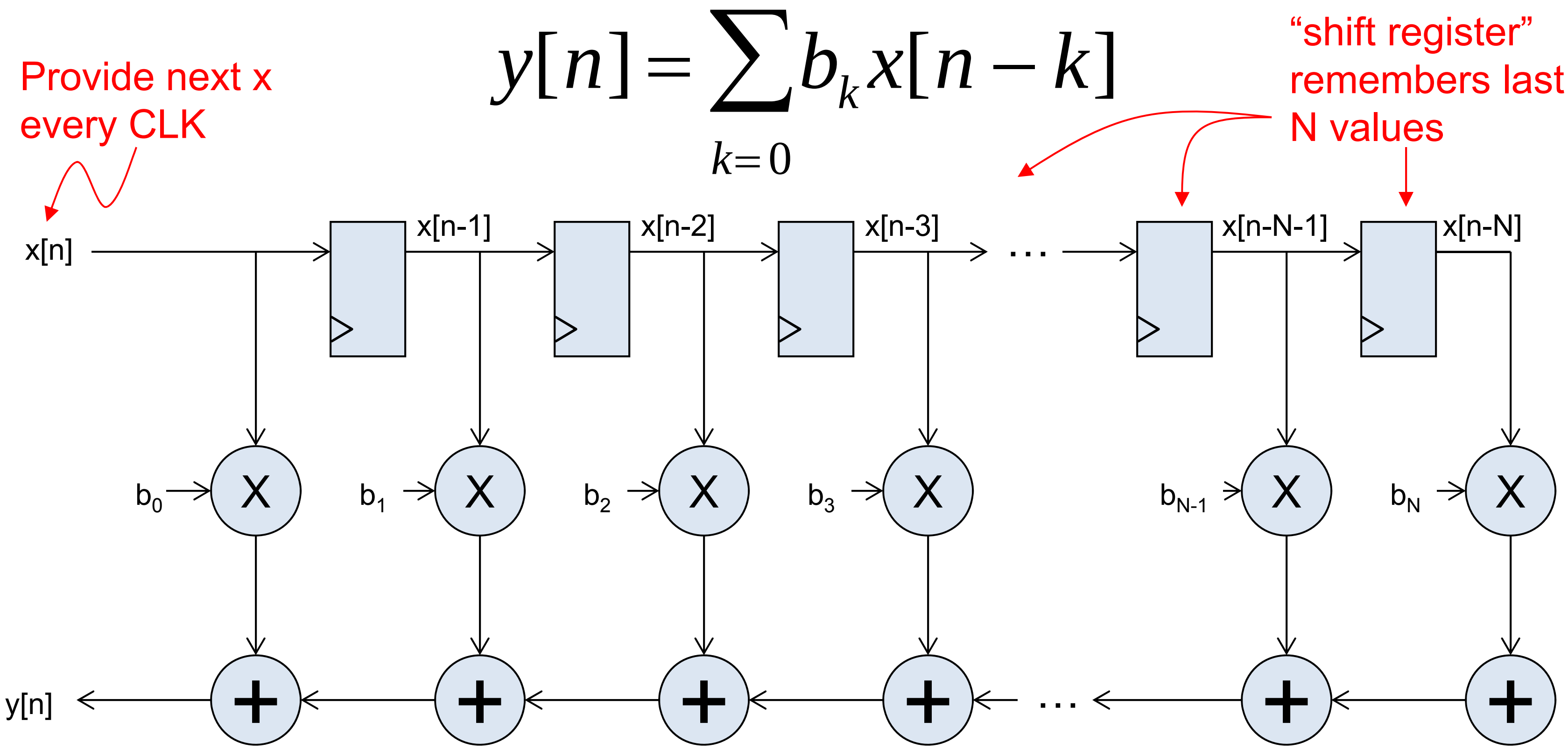
$$f\text{s} = (3\ \text{samples/line})(490\ \text{lines/frame})(30\ \text{frames/s}) = 44.1\ \text{kHz}$$

More info: http://www.cs.columbia.edu/~hgs/audio/44.1.html

Digital Filters

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

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



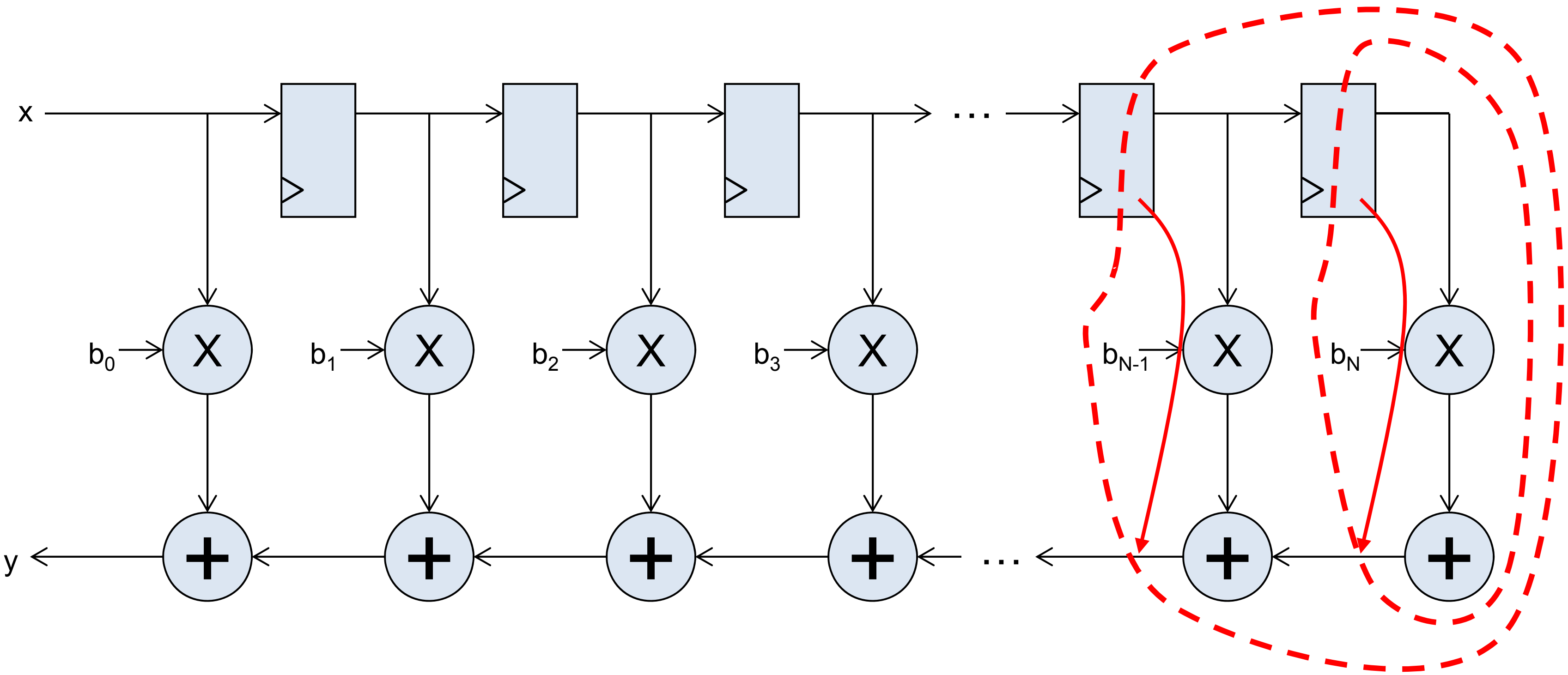
What components are part of the t_{PD} of this circuit?
How does t_{PD} grow as N gets larger?

Filter coefficients

- Use Matlab command: $b = \mathrm{fir1}(N,\omega_C/(\omega_S/2))$
  - 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 5)
  - ω_S is the sample frequency (48kHz in Lab 5)
  - 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!

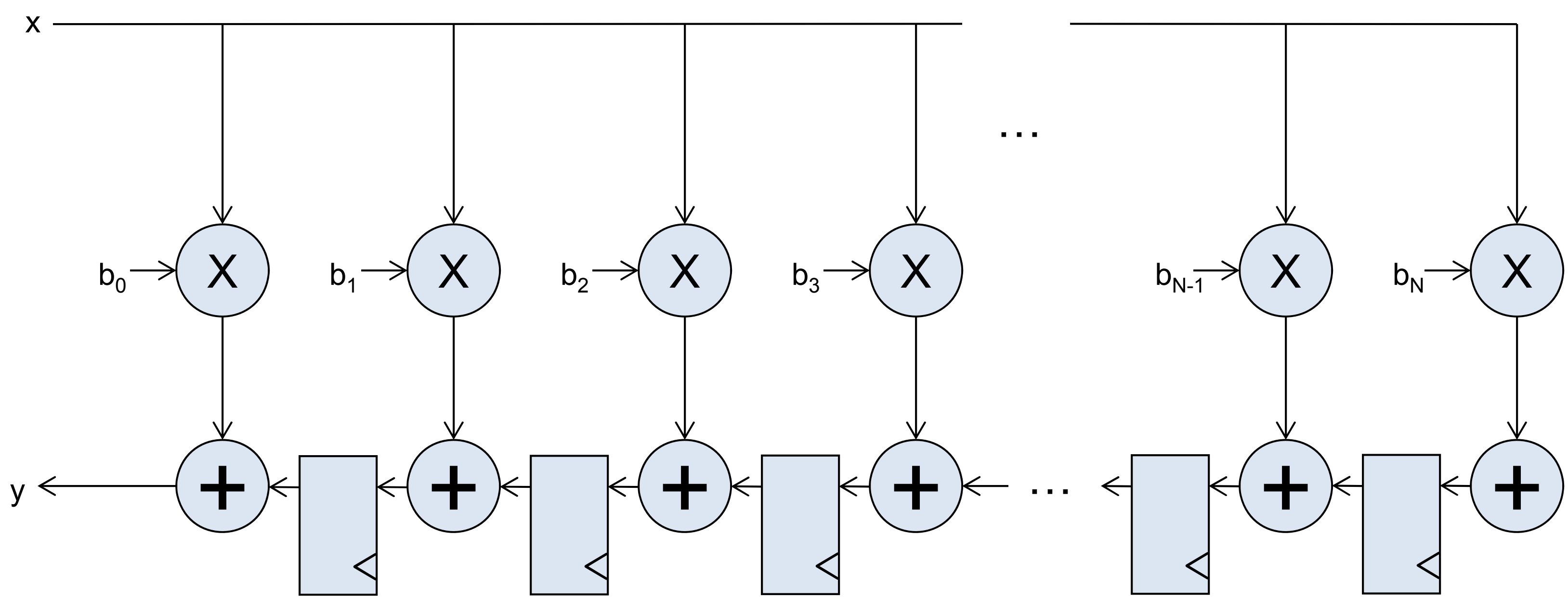
Retiming the FIR circuit

Apply the cut-set retiming transformation repeatedly…



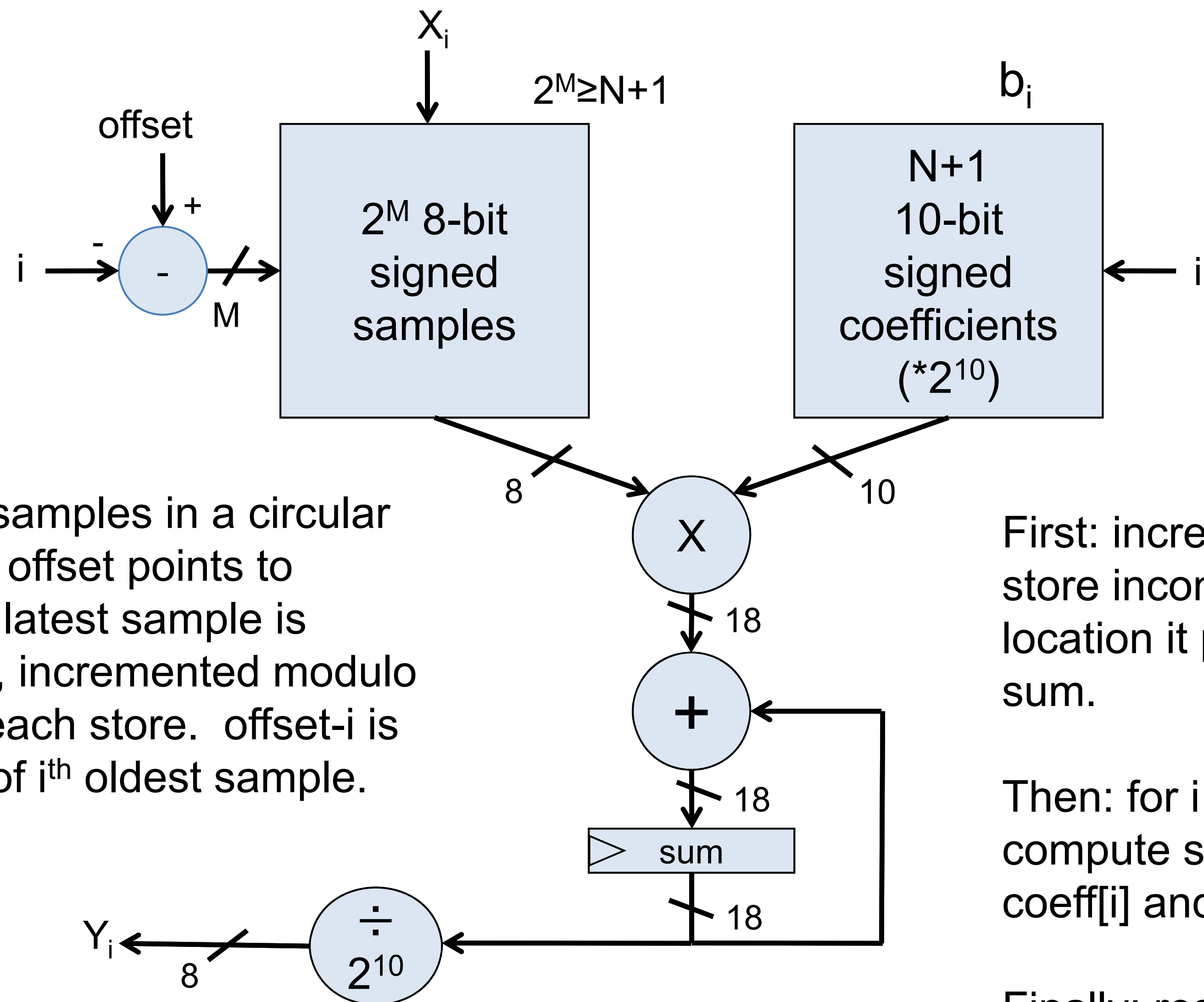
Retimed FIR filter circuit

"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…

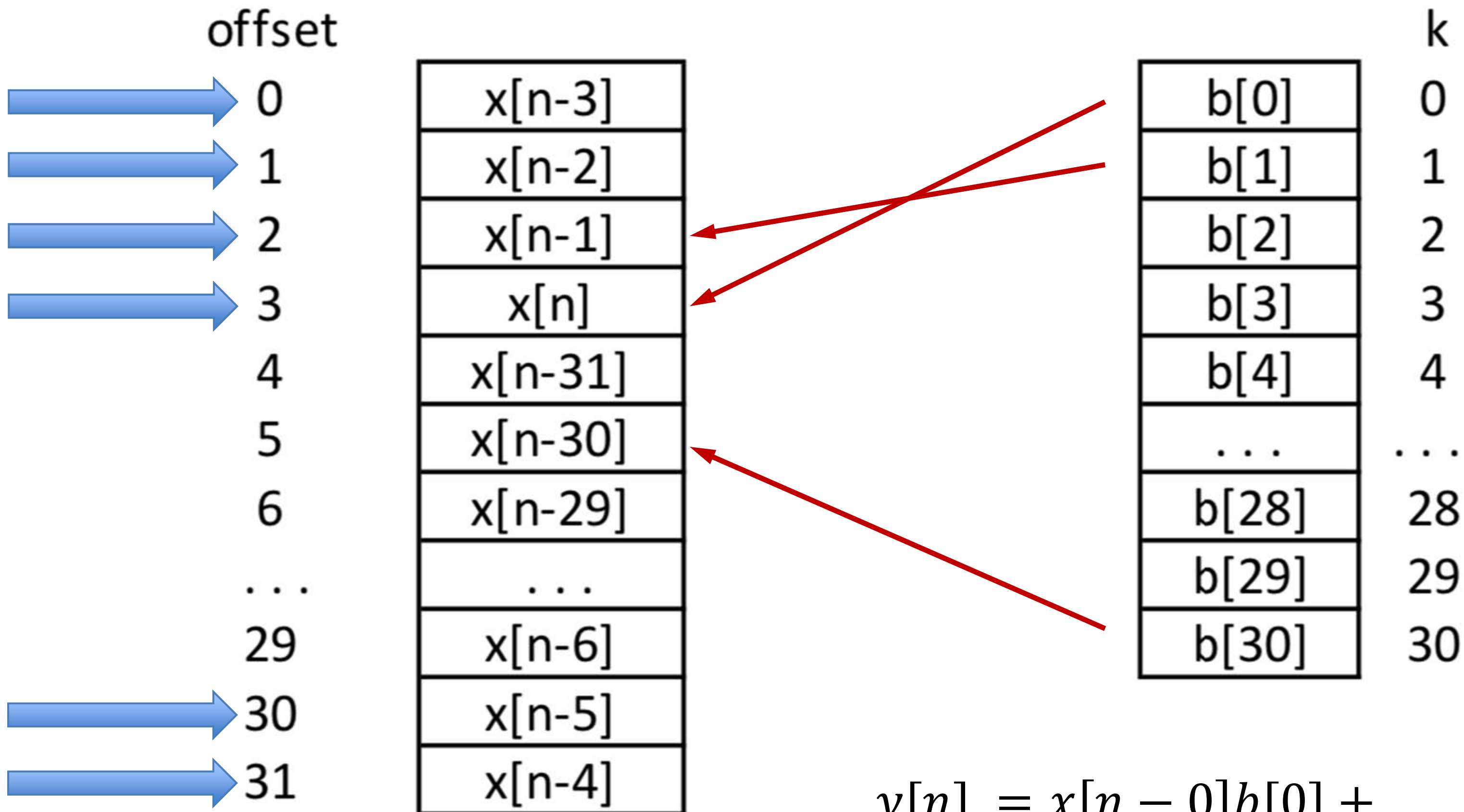


Store samples in a circular buffer: offset points to where latest sample is stored, incremented modulo 2^M at each store. offset-i is index of i^{th} oldest sample.

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

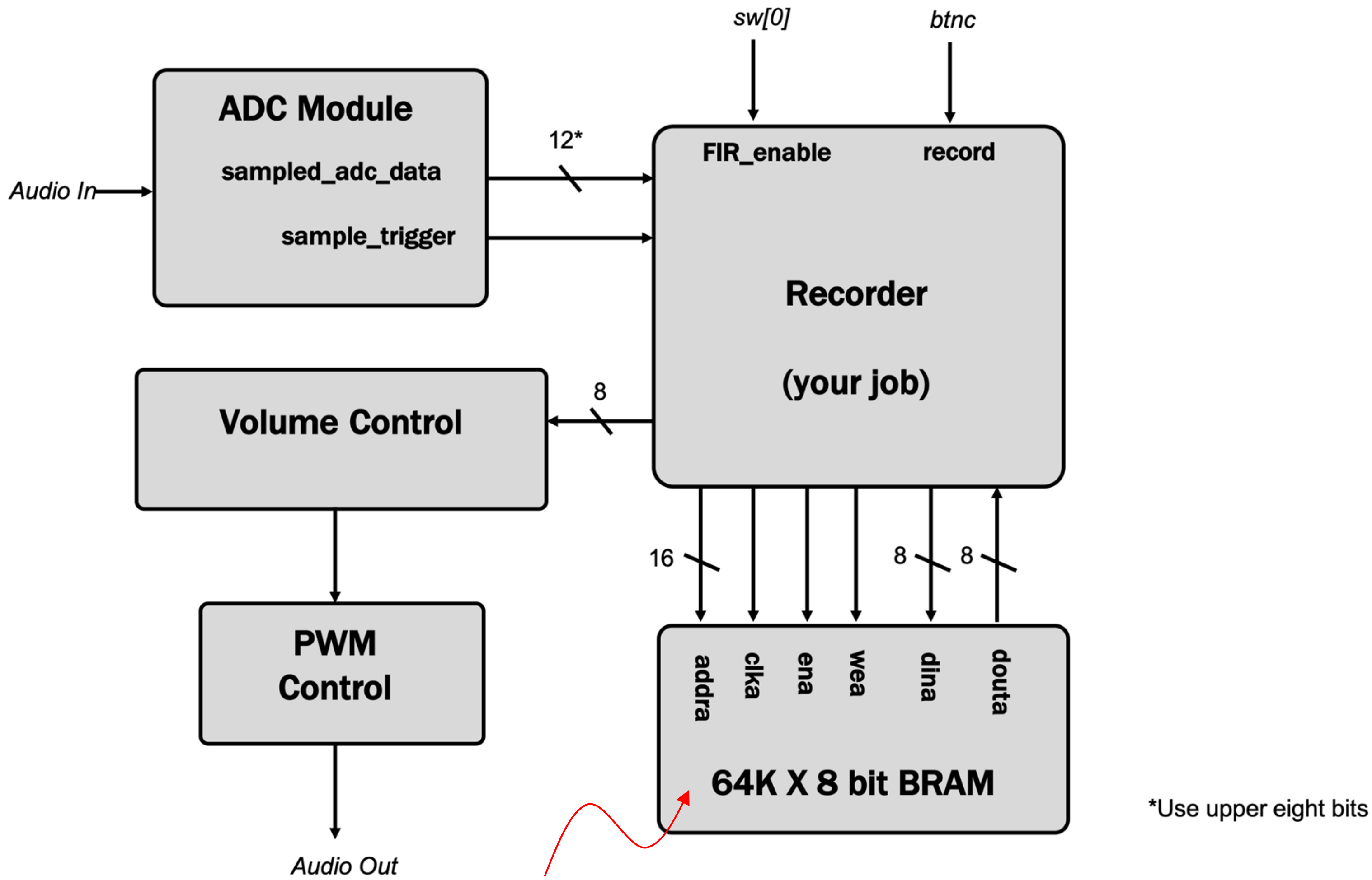


$$y[n] = \sum_{k=0}^{30} x[n-k]b[k]$$

$$\begin{aligned} y[n] &= x[n-0]b[0] + \\ &\quad x[n-1]b[1] + \ldots \\ &\quad x[n-30]b[30] + \end{aligned}$$

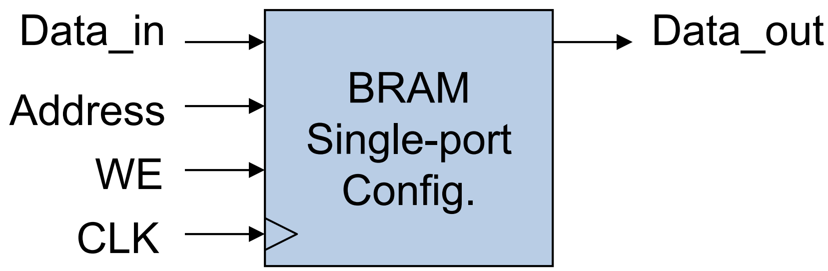
Lab 5a overview

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



About 11 seconds of speech @ 6KHz

BRAM Operation



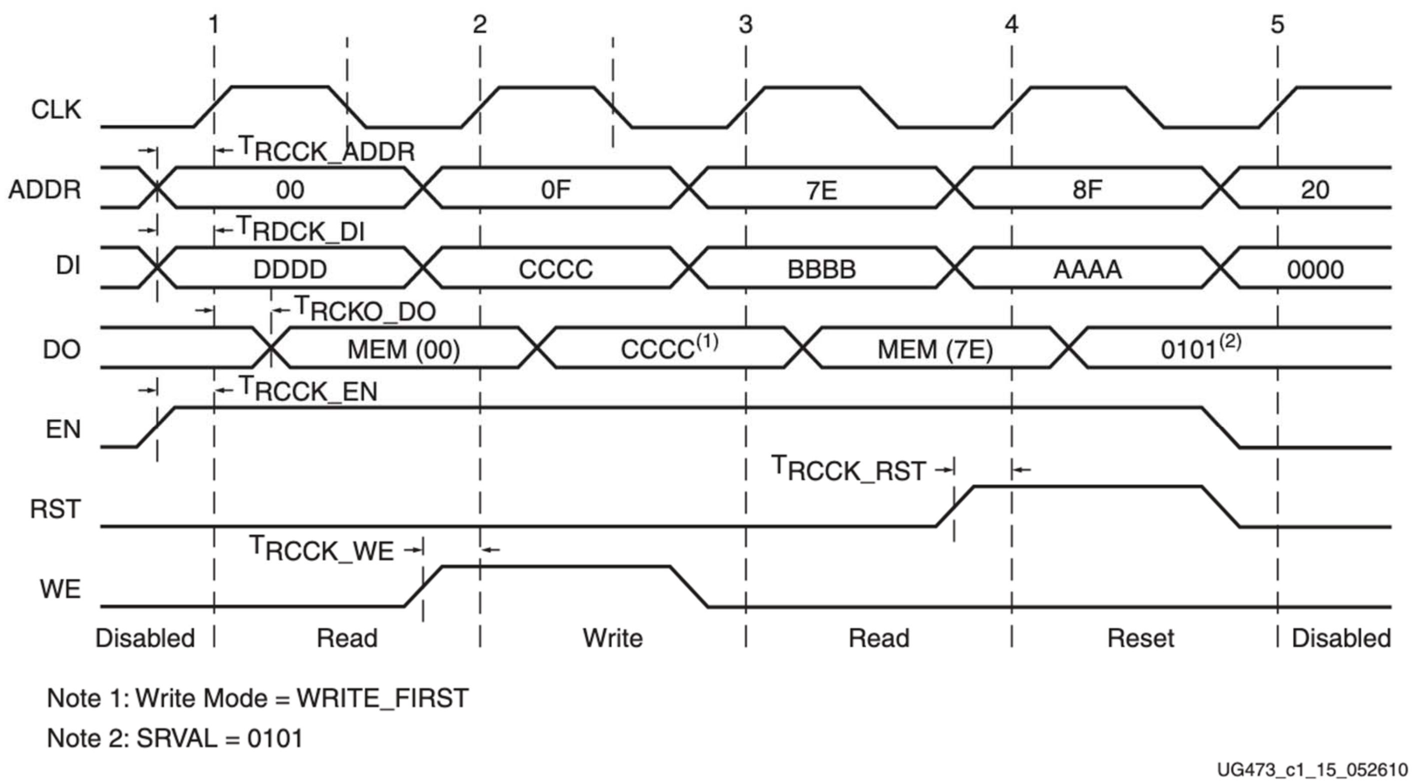


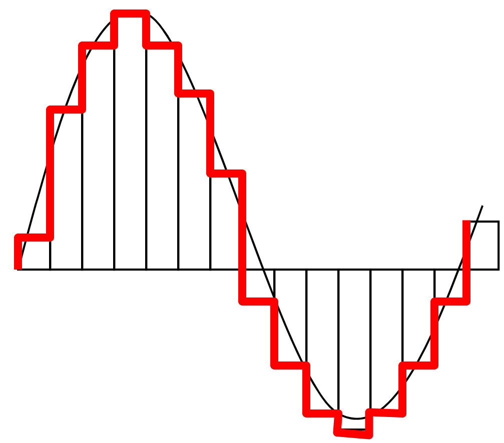
Figure 1-15: **Block RAM Timing Diagram**

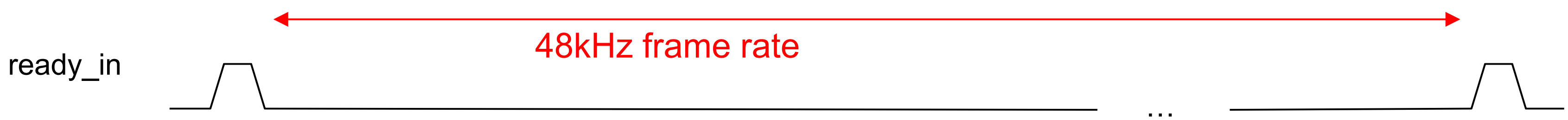
Source: Xilinx UG473 (v1.14) July 3, 2019

AC97: PCM data

PCM = pulse code modulation

Sample waveform at 48kHz, encode results as an N-bit signed number. For XADC chip, N = 12.



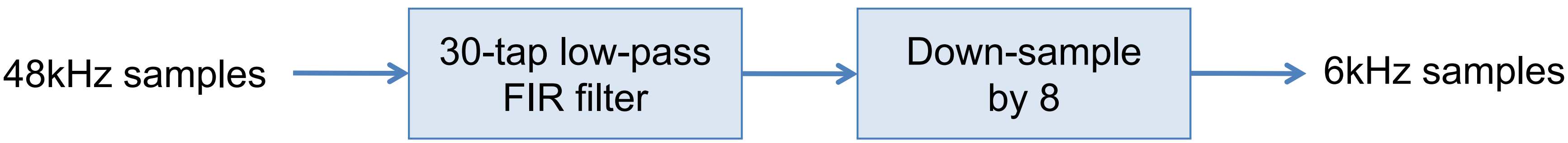


Record: when the ready_in input is asserted, a new sample from the microphone is available on the mic_in[7:0]

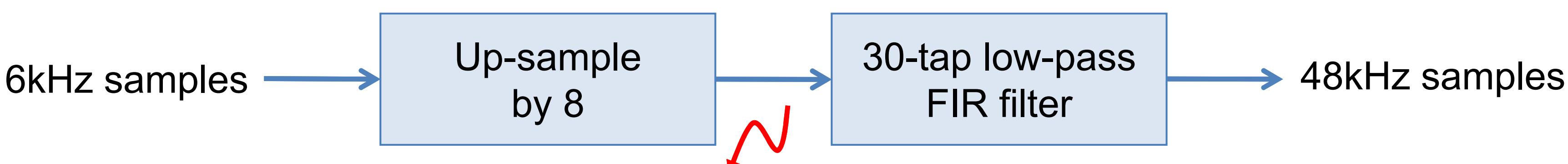
Playback: when the ready_in input is asserted, supply a 8-bit sample on the data_out[7:0] output and hold it there until the next sample is requested.

Lab 5a* 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).



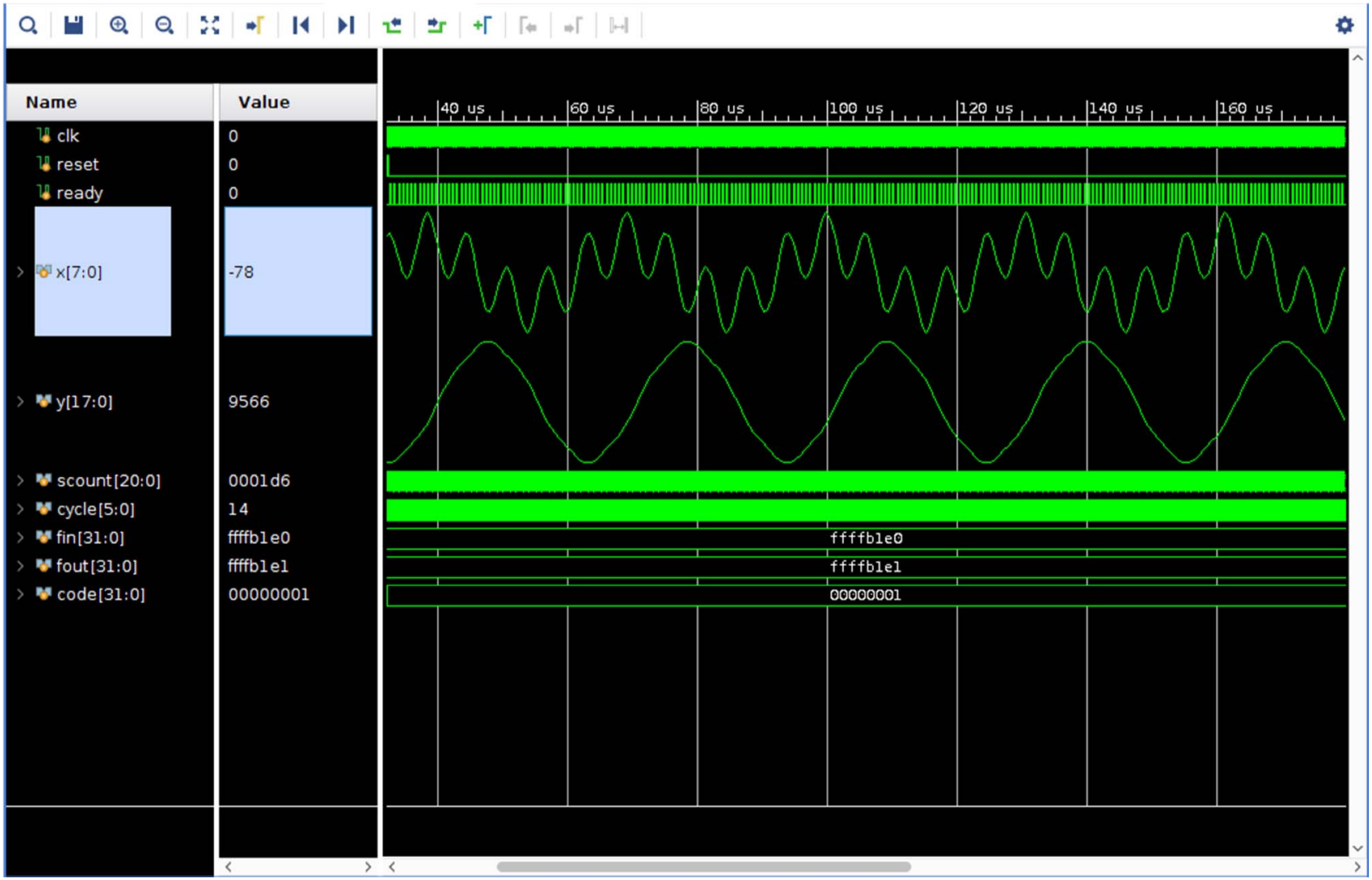
- We need a low-pass reconstruction filter (the same filter as for antialiasing!) when playing back the 6kHz samples. Actually we'll run it at 48kHz and achieve a 6kHz playback rate by feeding it a sample, 7 zeros, the next sample, 7 more zeros, etc.



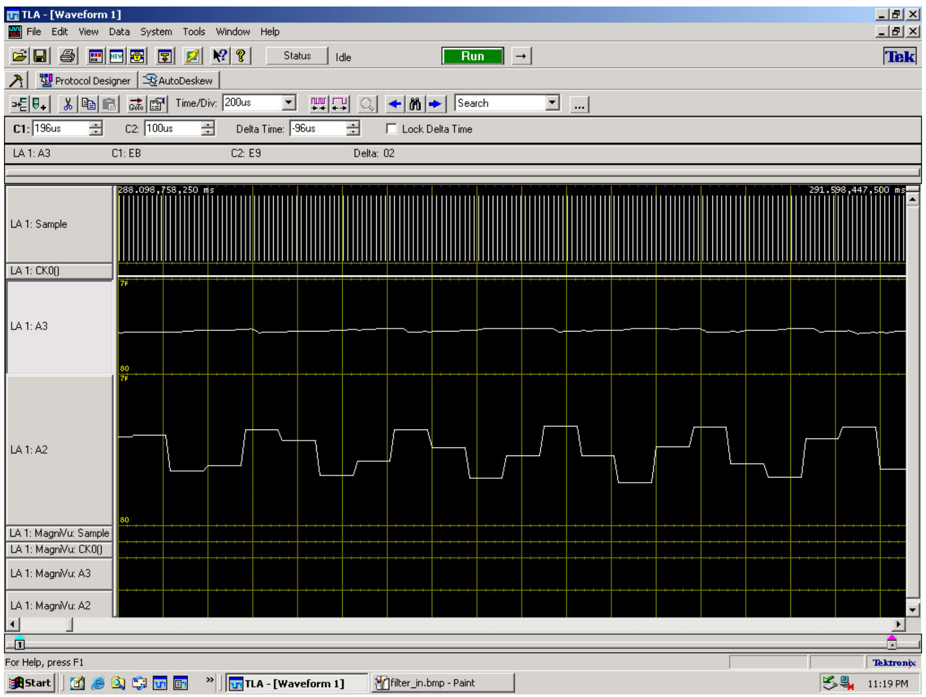
*Choose Lab5a or Lab5b

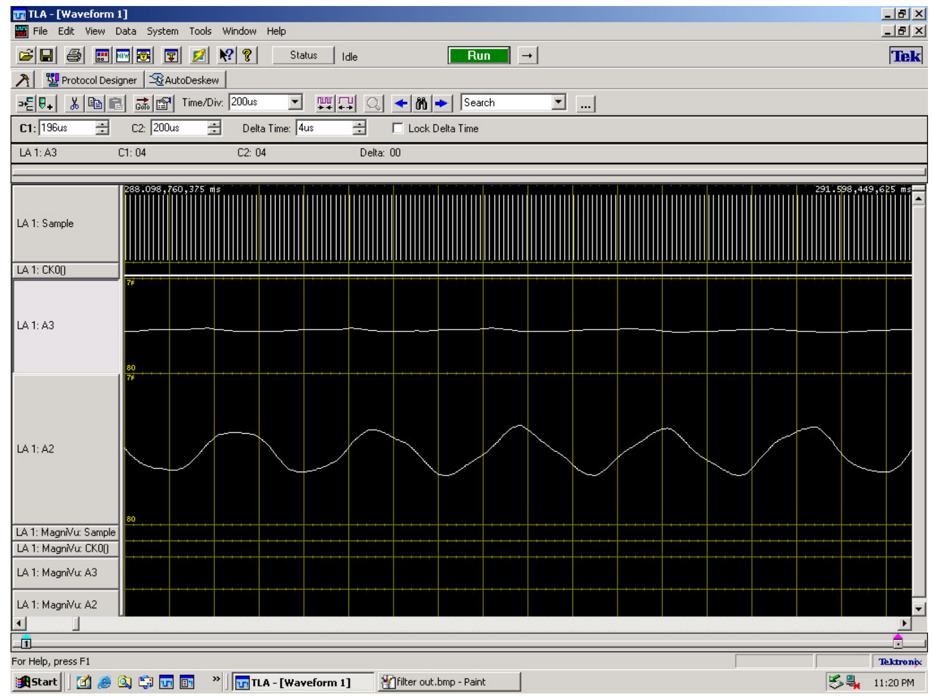
$\ldots, X_i, 0,0,0,0,0,0,0, X_{i+1}, 0,0,0,0,0,0,0, X_{i+2}, \ldots$

FIR Filter – Data Input



FIR Filter – Playback

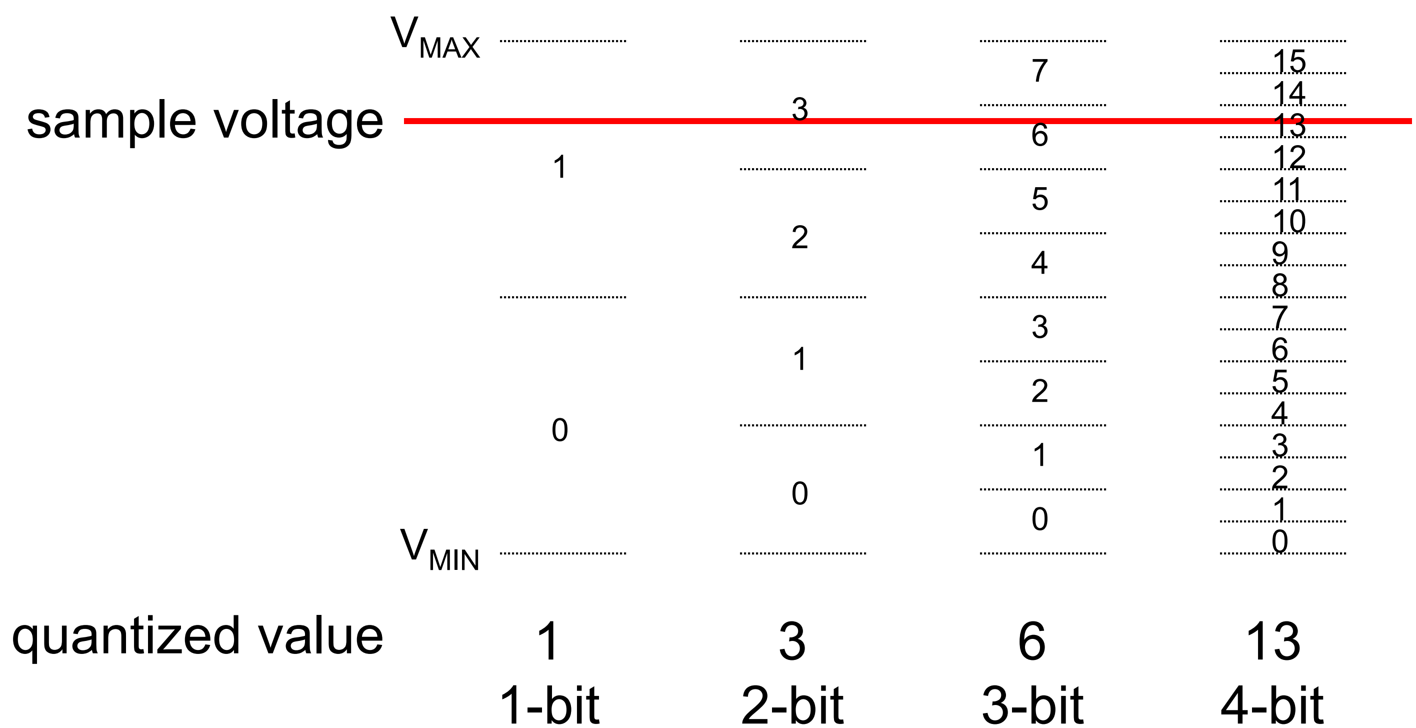




Discrete Values

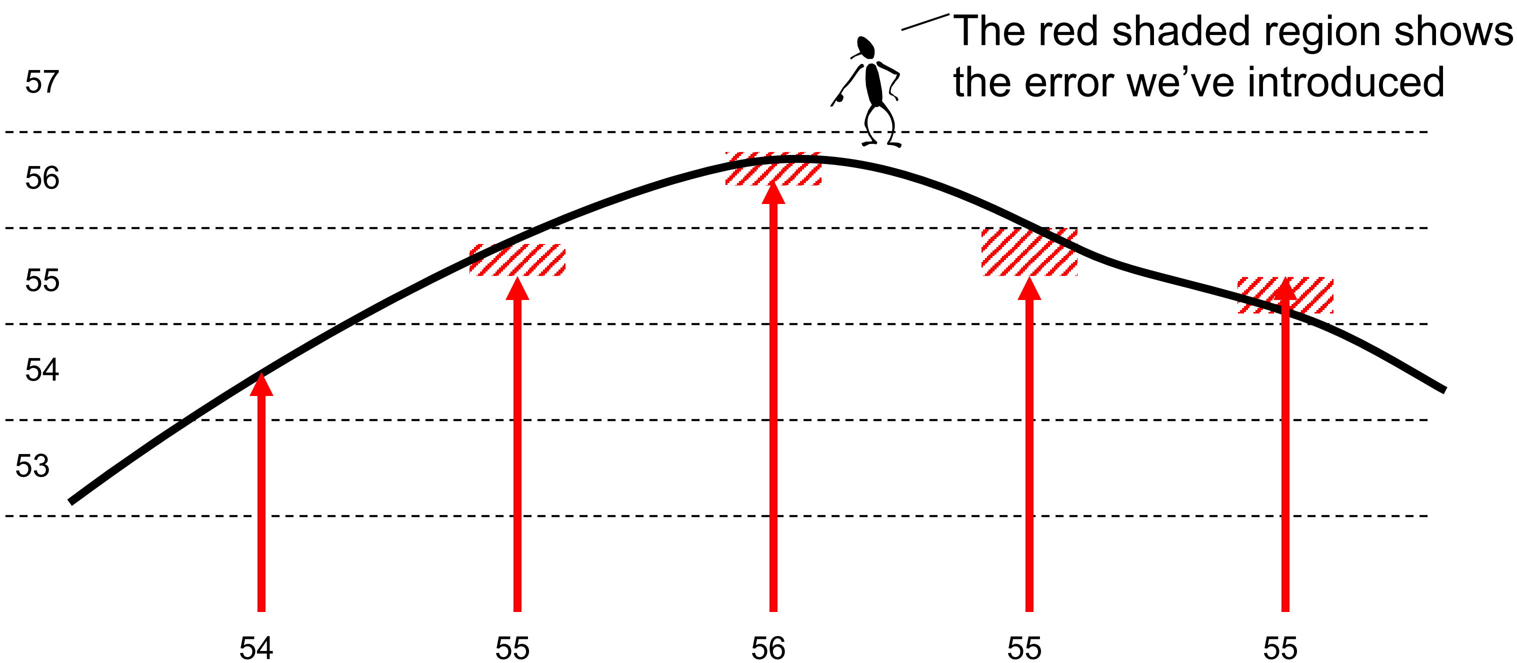
If we use N bits to encode the magnitude of one of the discrete-time samples, we can capture 2^N possible values.

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.

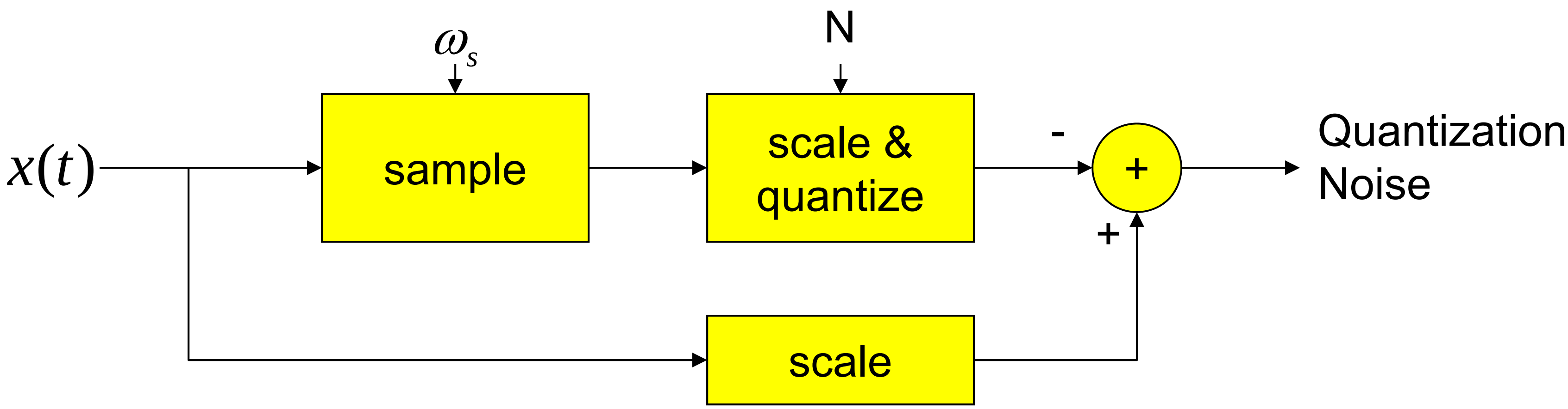


Quantization Error

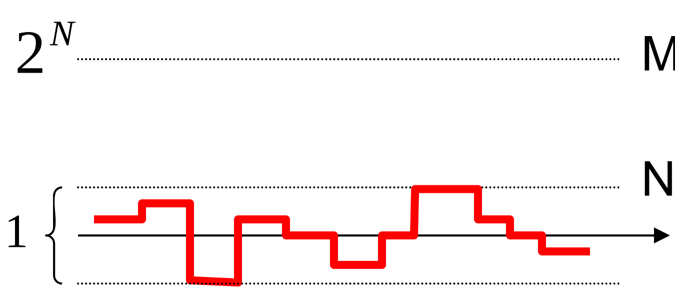
Note that when we quantize the scaled sample values we may be off by up to $\pm$½ step from the true sampled values.



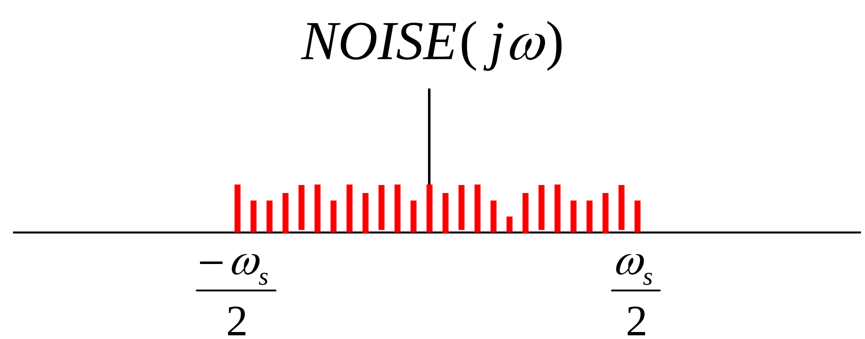
Quantization Noise



Time Domain



Freq. Domain



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

Decibel (dB) – 3dB point

$$dB = 20\log\left(\frac{V_o}{V_i}\right)$$

$$dB = 10\log\left(\frac{P_o}{P_i}\right)$$

$\log_{10}(2)$=.301

3 dB point = ?

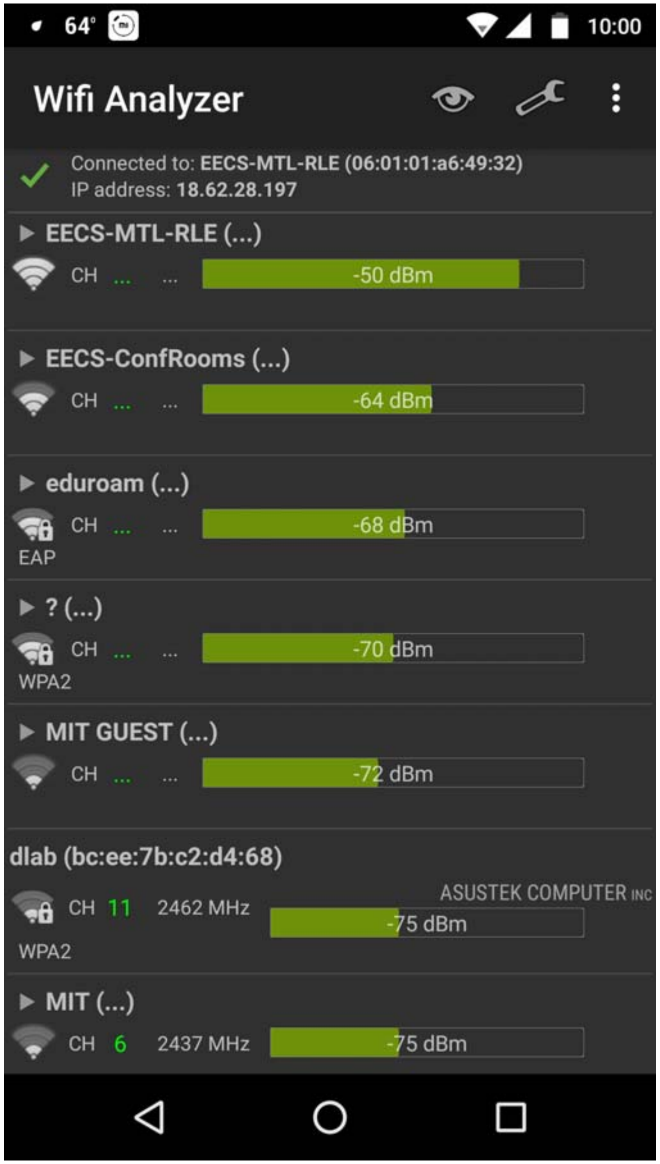
half power point

100 dB =	100,000	= 10^5
80 dB =	10,000	= 10^4
60 dB =	1,000	= 10^3
40 dB =	100	= 10^2

Common Decibel Units

dB UNIT	reference	application
dbV	1 Volt rms	routine voltage measurements [comparisons!]
dBm	1 mW into 50Ω [0.224V] or 600Ω [0.775V]	radio-frequency [50Ω] or audio [600Ω] power measurements [in England, the dBu is used to mean 0.775V reference without regard to impedance or power]
dB mV	1 millivolt rms	signal levels in cable systems
dbW	1 Watt	audio power amplifier output [usually into 8, 4, or 2Ω impedances]
dBf	1 femtowatt [10^{-15} watt]	communications and stereo receiver sensitivity [usually 50Ω, 75Ω unbalanced, or 300Ω balanced antenna input impedances]
dB (SPL)	0.0002μbar, = 20μPa [=Pascals] [1 bar = 10^6 dynes/cm^2 ~1AT]	Sound Pressure Level measurements: the reference is the "threshold of hearing".

Wifi Signal Strength



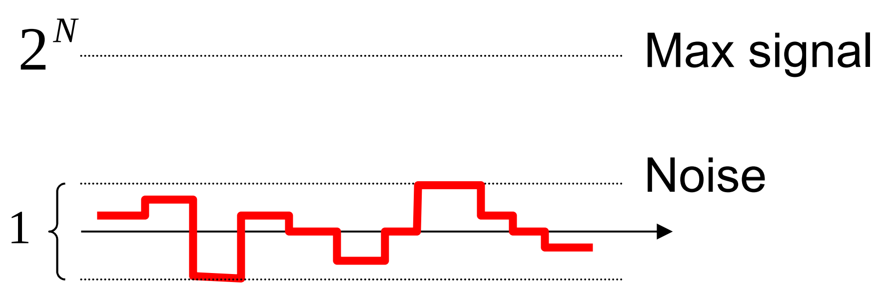
-60dBm = 1 uWatt

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

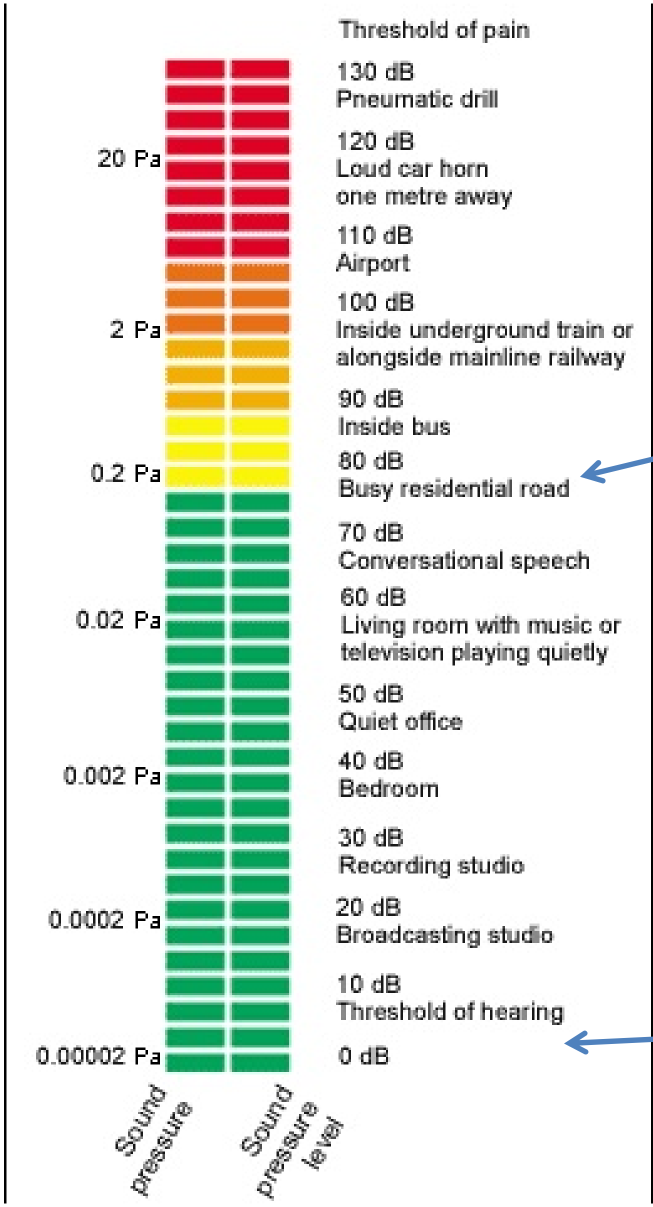
SNR is measured in decibels (dB). Note that it’s a logarithmic scale: if SNR increases by 3dB the ratio has increased by a factor 2. When applied to audible sounds: the ratio of normal speech levels to the faintest audible sound is 60-70 dB.



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

$$\approx N \cdot 6.02 dB$$

Sound Levels*



noise induced hearing loss (NIHL)

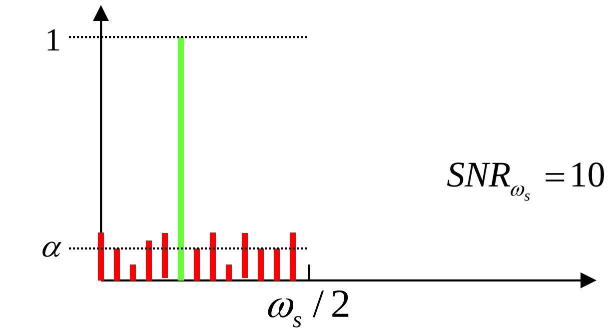
mosquito at 3 yards

* www.osha.gov

Oversampling

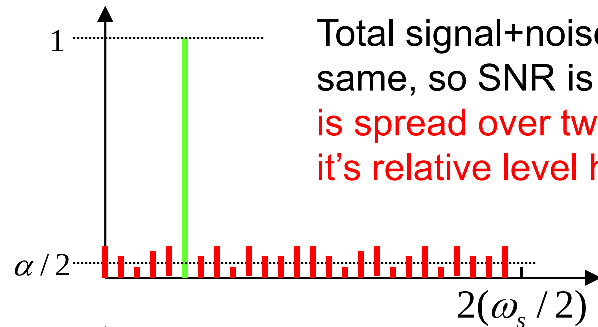
To avoid aliasing we know that ω_s must be at least $2\omega_M$. Is there any advantage to oversampling, i.e., $\omega_s = K \cdot 2\omega_M$?

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



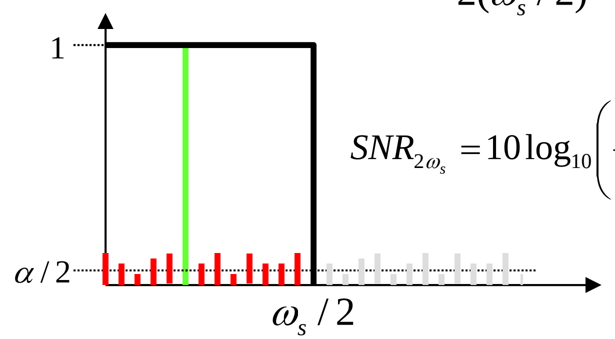
$$SNR_{\omega_s} = 10\log_{10}\left(\frac{P_{SIGNAL}}{P_{NOISE}}\right)$$

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…



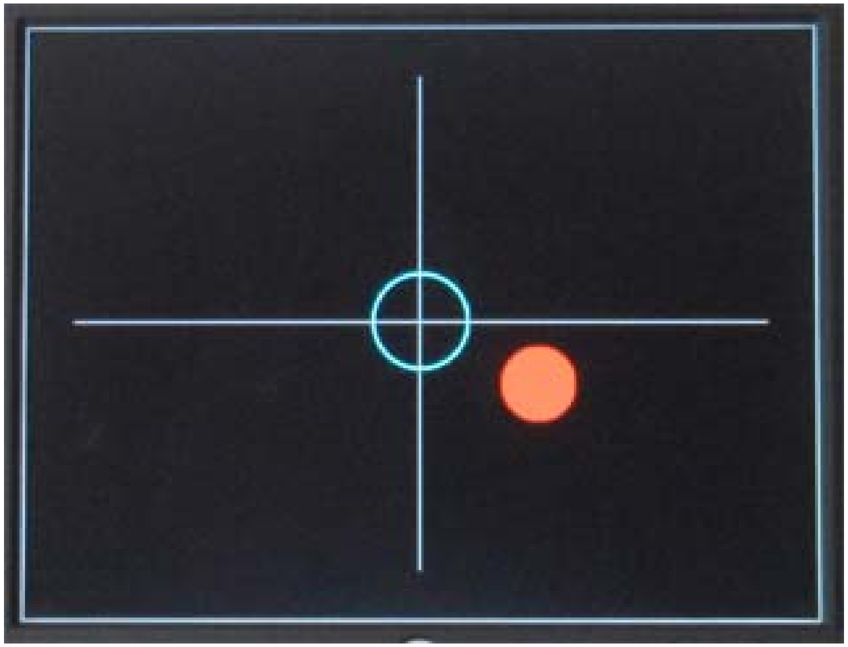
$$SNR_{2\omega_s} = 10\log_{10}\left(\frac{P_{SIGNAL}}{P_{NOISE}/2}\right) = SNR_{\omega_s} + 3dB$$

Oversampling+LPF reduces noise by 3dB/octave

Lab 5b Overview

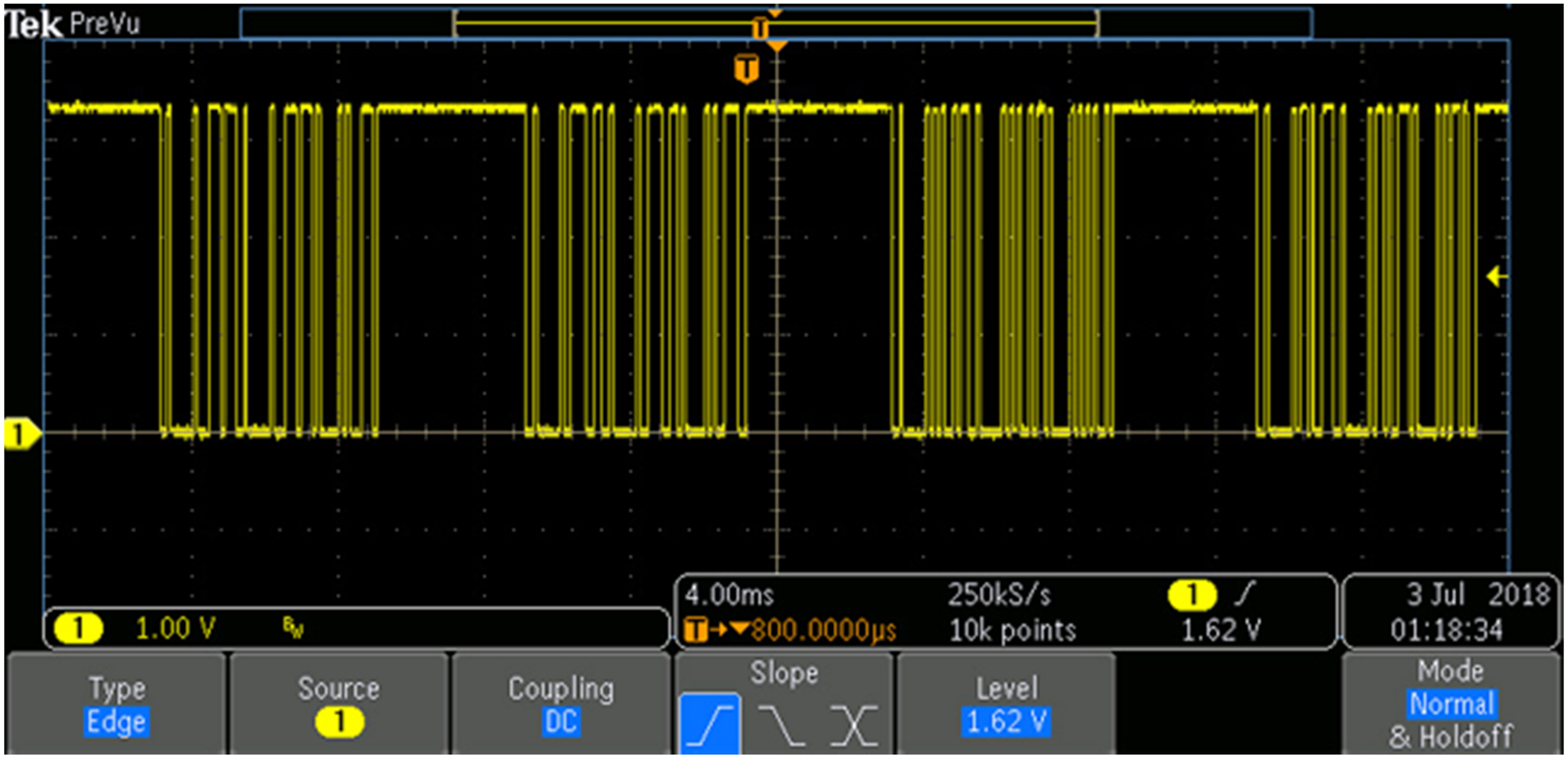
Assignment: Design a digital bubble level using data from an inertial measurement unit (IMU) and display the results on a monitor.

- MPU-9250 IMU
  - 3 axis accelerometer
  - 3 axis gyro
  - 3 axis magnetometer
- Data sent via i2c to Teensy
- Data transmitted by Teensy via serial protocol at 100hz



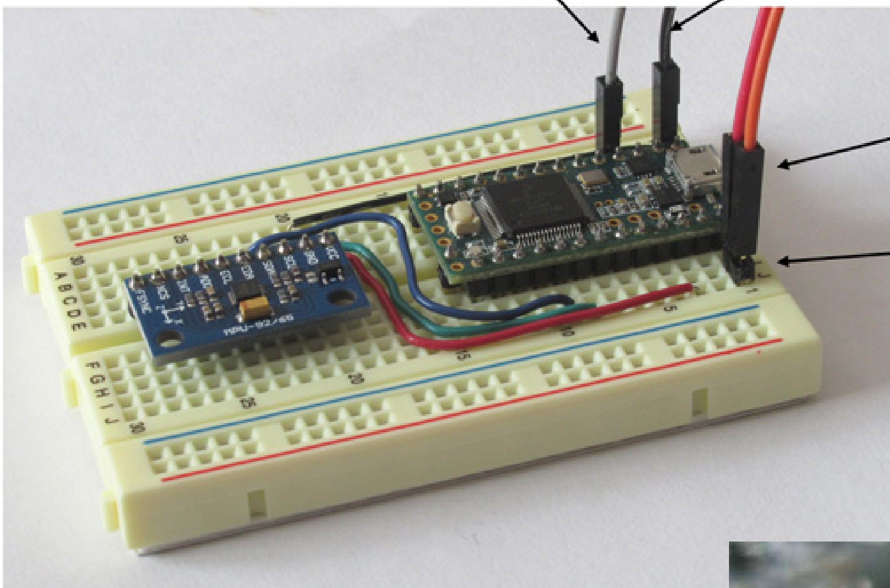
Bubble Level

Lab 5c Data Format



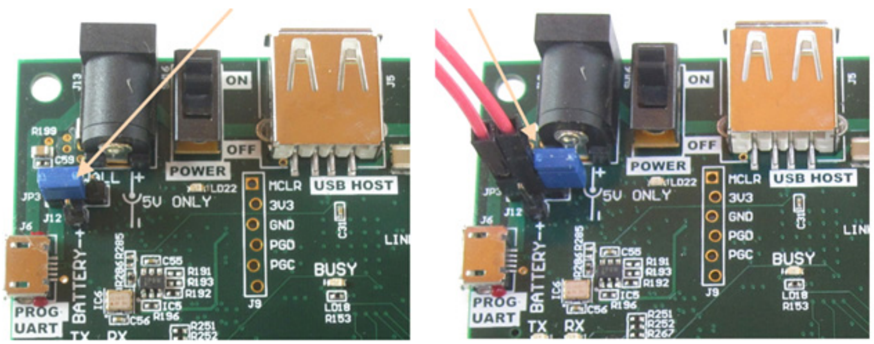
- 3 axis transmitted, only x,y axis data used
- 16 bit 2's complement format
- 9600 baud, lsb first

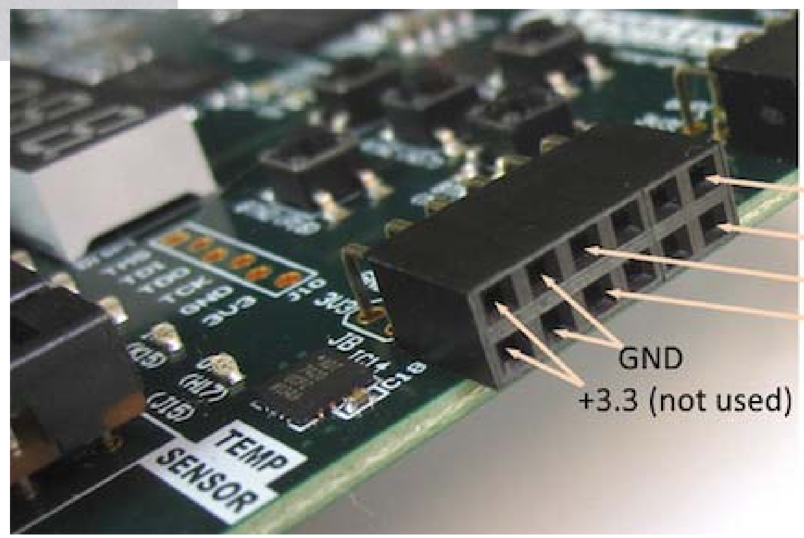
Lab 5c Interconnect



Color coded wires
provided for Nexys4

Move blue jumper two pins right
and attach 2 wires for +5V





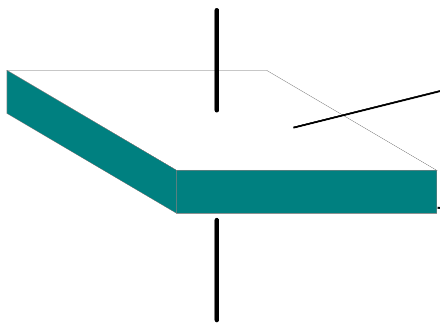
IMU Inertial Measurement Unit

- MEMS Accelerometer – MicroElectroMechanical Systems
- MEMS components generally 1-100 microns
- Silicon based – MEMS device fabricated on same silicon as circuits
- Circuits and digital processing key to MEMS

Movement sensing

- Accelerometers
  - Acceleration – movement from one point to another
  - Tilt sensing – measures inclination/angle with respect to gravity
- Gyroscopes
  - Rotation sensing – measures angular rate.

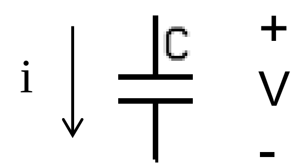
Capacitance



A = surface area of plates

d = distance between plates.

$$c = \frac{K\varepsilon_0 A}{d} \qquad i = C\frac{dv}{dt}$$



MEMS Capacitors*

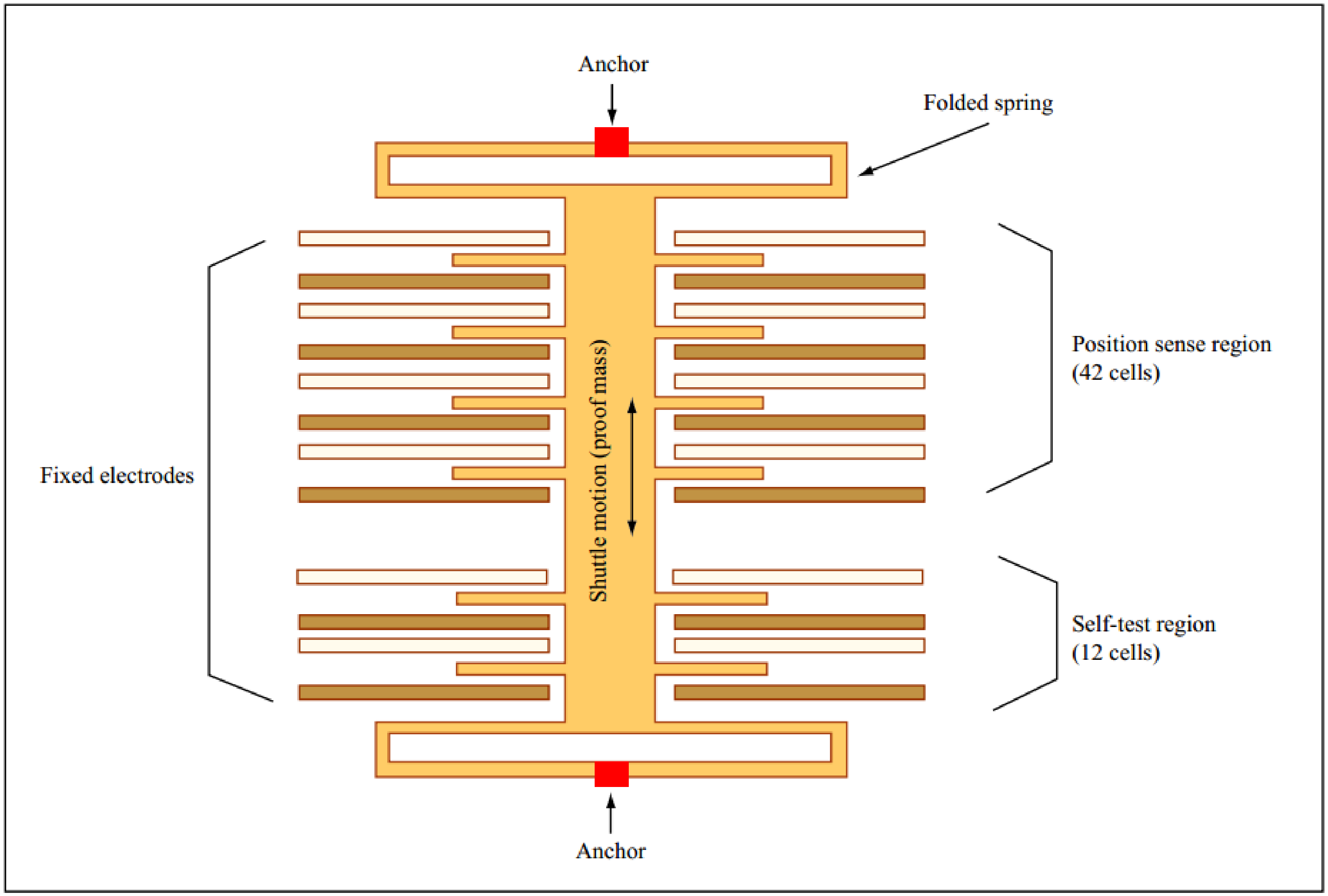
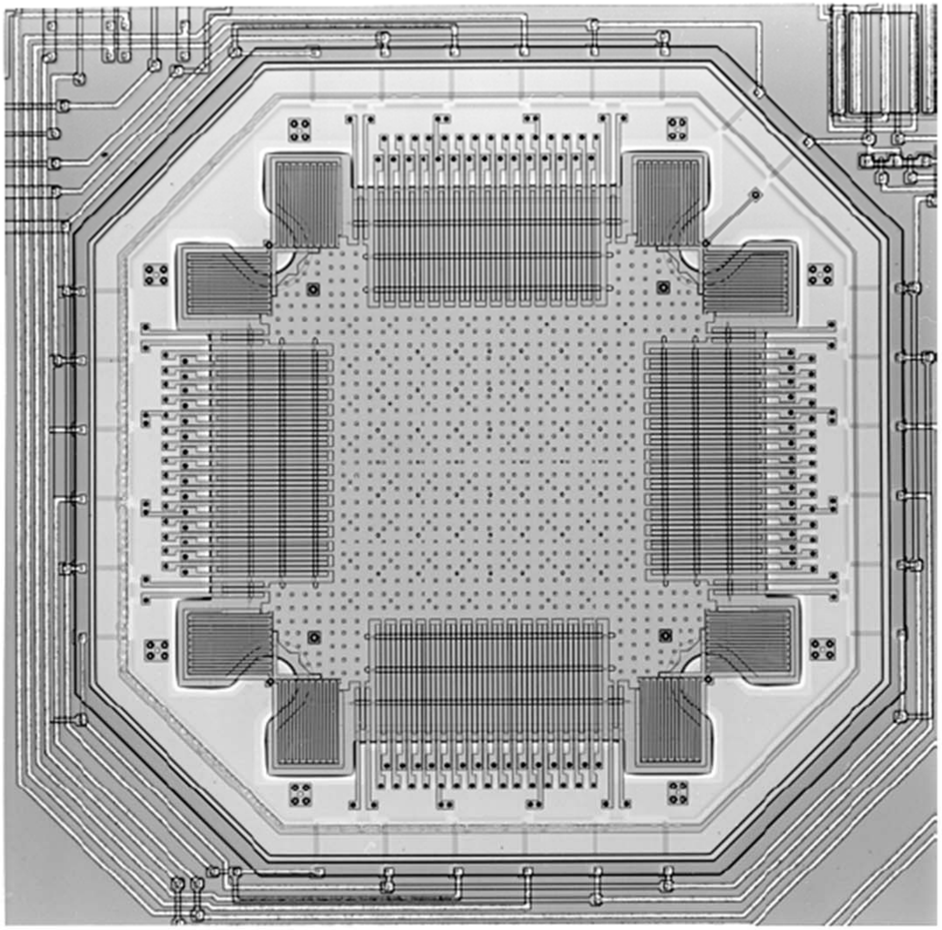


Image by MIT OpenCourseWare.

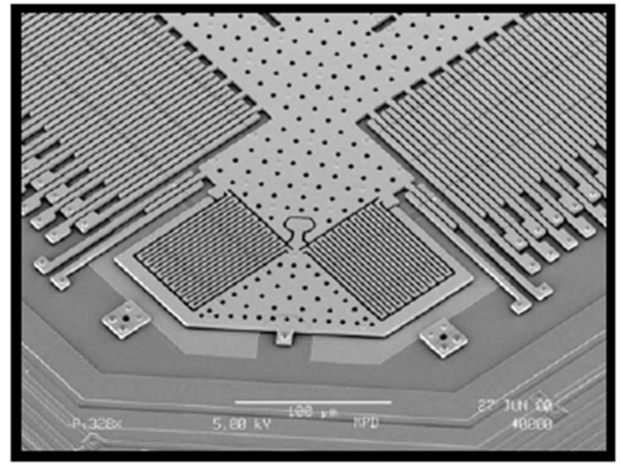
*6.777J OCW

2 Axis Accelerometer

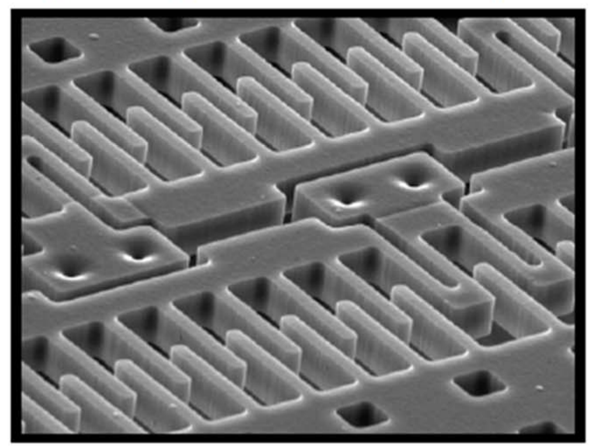


ADXL202 Sensor Structure

Courtesy of Analog Devices, Inc. Used with permission.



Courtesy of Analog Devices, Inc. Used with permission.



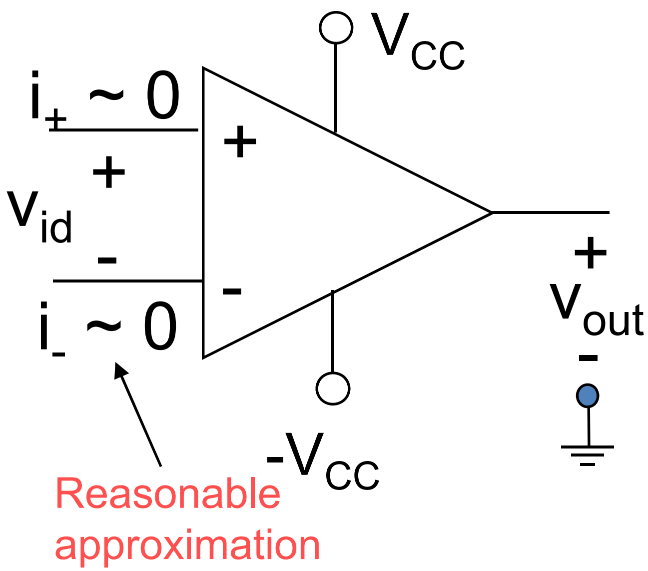
Courtesy of Analog Devices, Inc. Used with permission.

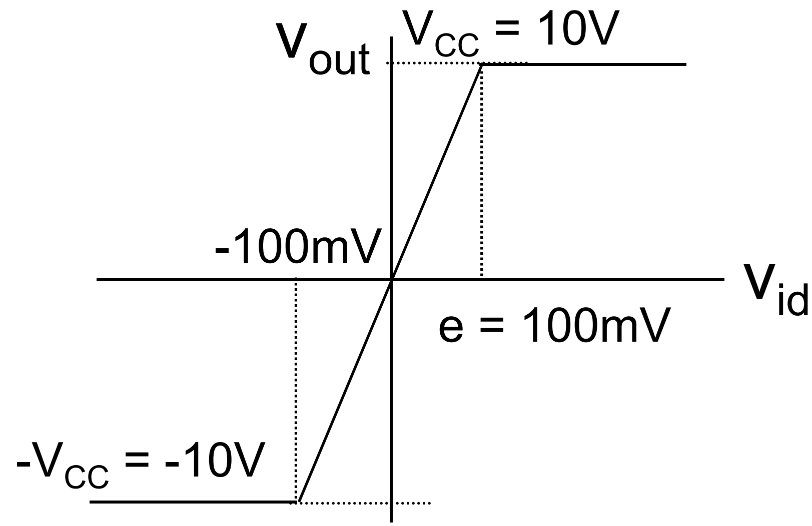
Giant “MEMS” Capacitor

Mems

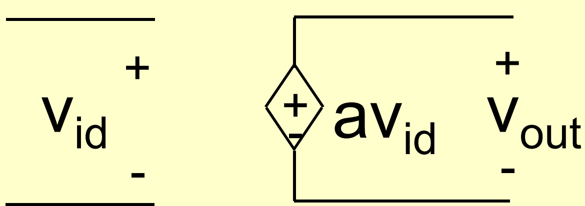
- Passenger sensor
- Tire pressure sensor
- Airbag deployment
- Phone rotation

Our Analog Building Block: OpAmp



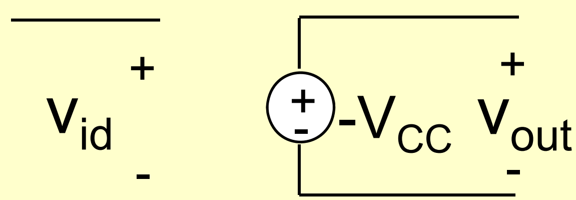


Linear Mode



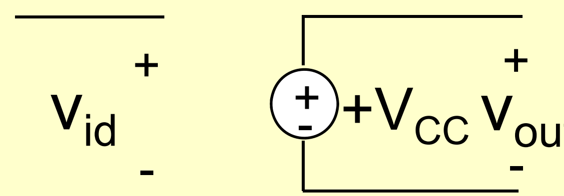
If $-V_{CC} < v_{out} < V_{CC}$

Negative Saturation



$v_{id} < -e$

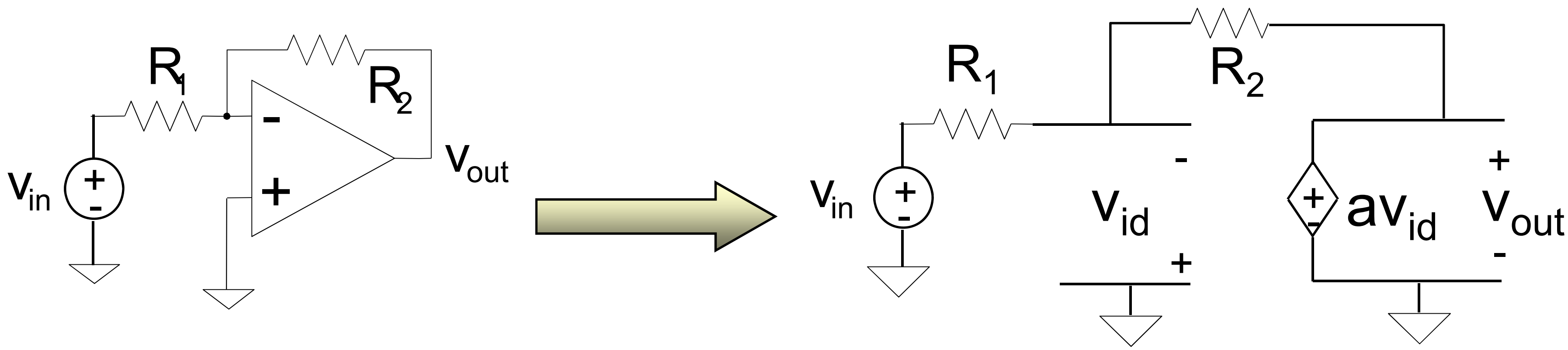
Positive Saturation



$v_{id} > e$

Very small input range for “open loop” configuration

The Power of (Negative) Feedback



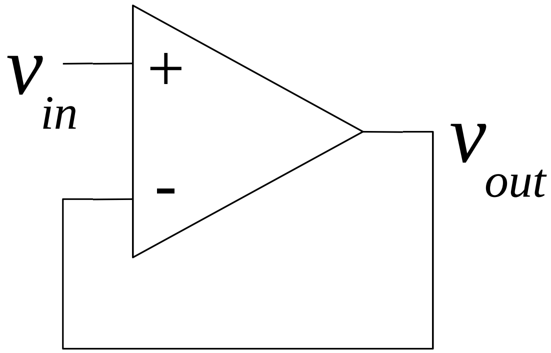
$$\frac{v_{in} + v_{id}}{R_1} + \frac{v_{out} + v_{id}}{R_2} = 0 \qquad v_{id} = \frac{v_{out}}{a} \qquad \frac{v_{in}}{R_1} = -\frac{v_{out}}{a}\left[\frac{1}{R_1} + \frac{a}{R_2} + \frac{1}{R_2}\right]$$

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

- 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

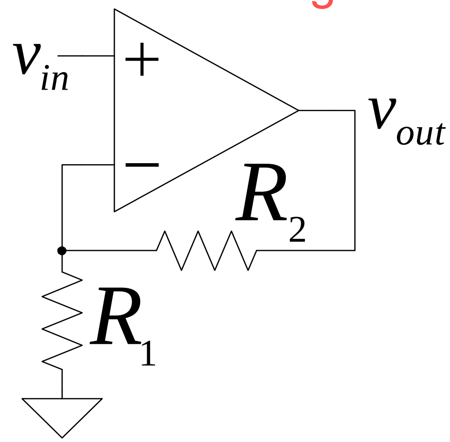
Basic OpAmp Circuits

Voltage Follower (buffer)



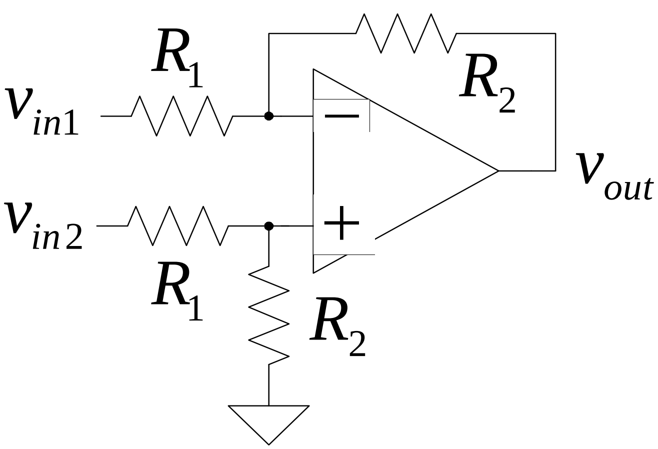
$$v_{out} \approx v_{in}$$

Non-inverting



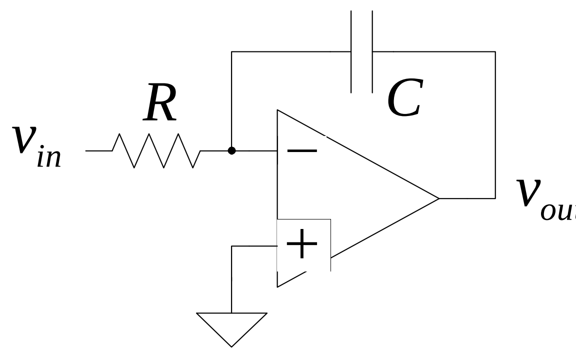
$$v_{out} \approx \frac{R_1+R_2}{R_1} v_{in}$$

Differential Input



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

Integrator

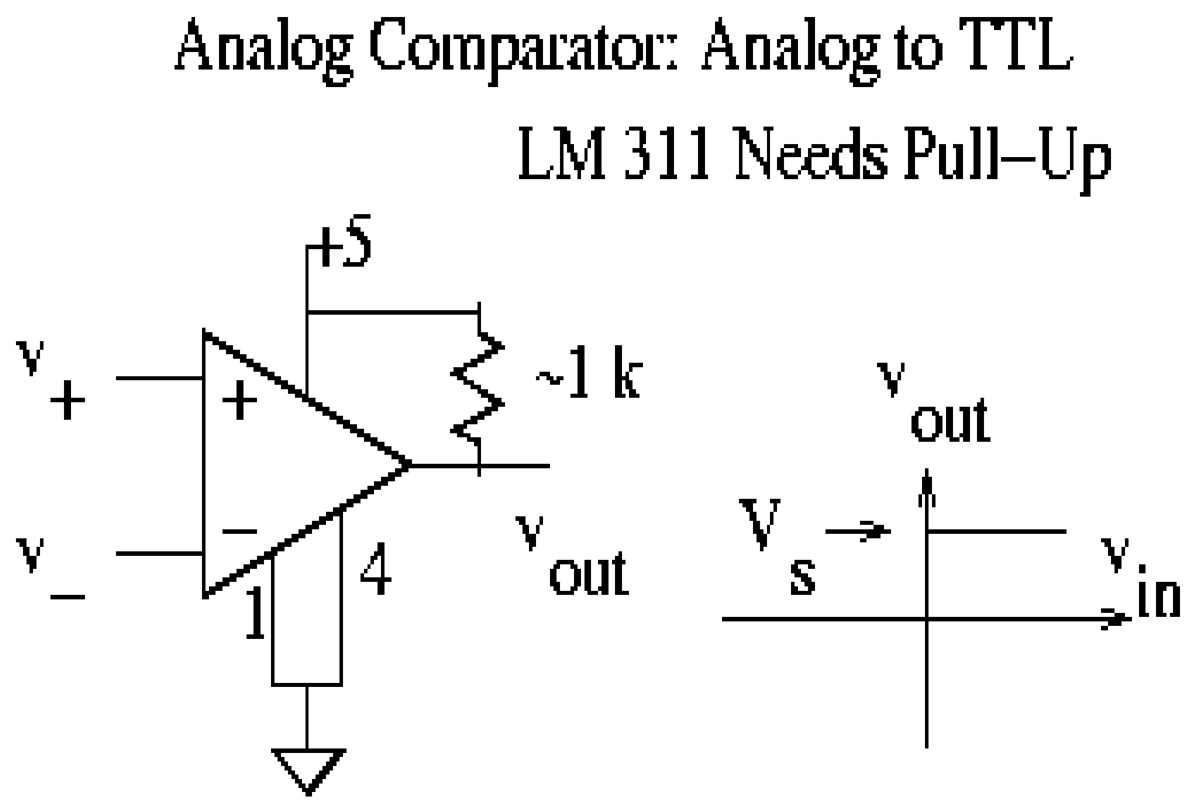


$$v_{out} \approx -\frac{1}{RC}\int_{-\infty}^{t} v_{in}\,dt$$

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

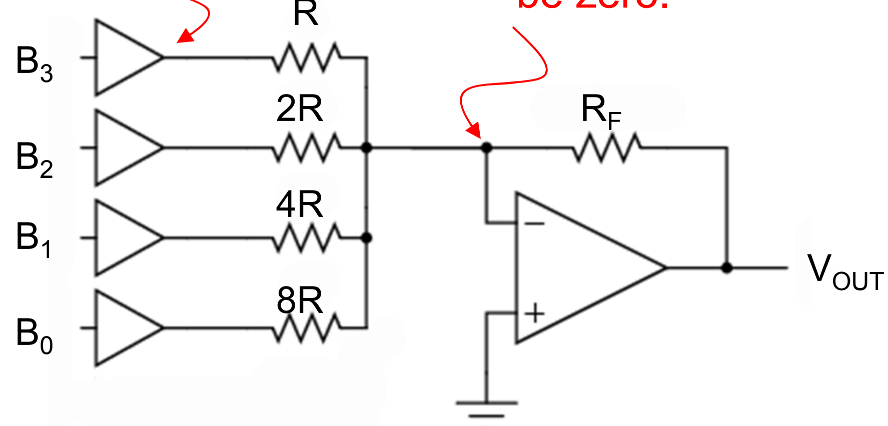
- Common applications:
  - Real world control (motors, lights)
  - Video signal generation
  - Audio / RF “direct digital synthesis”
  - Telecommunications (light modulation)
  - Scientific & Medical (ultrasound, …)

DAC: digital to analog converter

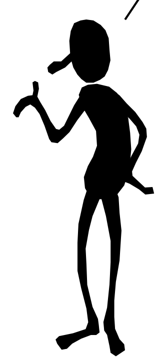
How can we convert a N-bit binary number to a voltage?

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.



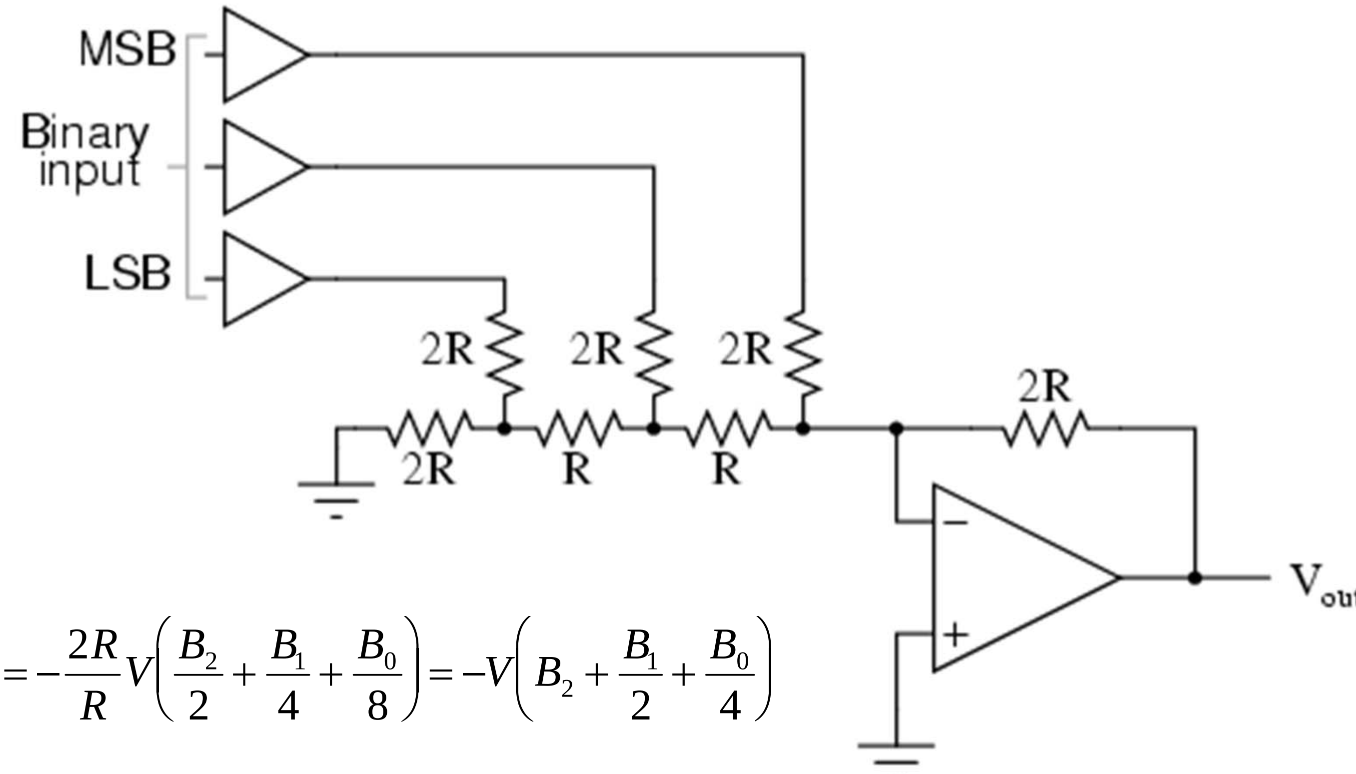
OKAY, this'll work, but the voltages produced by the drivers and various R's must be carefully matched in order to get equal steps.



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

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

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

Quantization*
A Graphical Example

How many bits are needed to represent 256 shades of gray (from white to black)?

Bits	Range		Bits	Range
1	2		5	32
2	4		6	64
3	8		7	128
4	16		8	256

* Acknowledgement: Quantization slides and photos by Prof Denny Freemen 6.003

Quantization: Images

Converting an image from a continuous representation to a discrete representation involves the same sort of issues.

This image has 280 × 280 pixels, with brightness quantized to 8 bits.

Quantizing Images

8 bit image

7 bit image

Quantizing Images

8 bit image

6 bit image

Quantizing Images

8 bit image

5 bit image

Quantizing Images

8 bit image

4 bit image

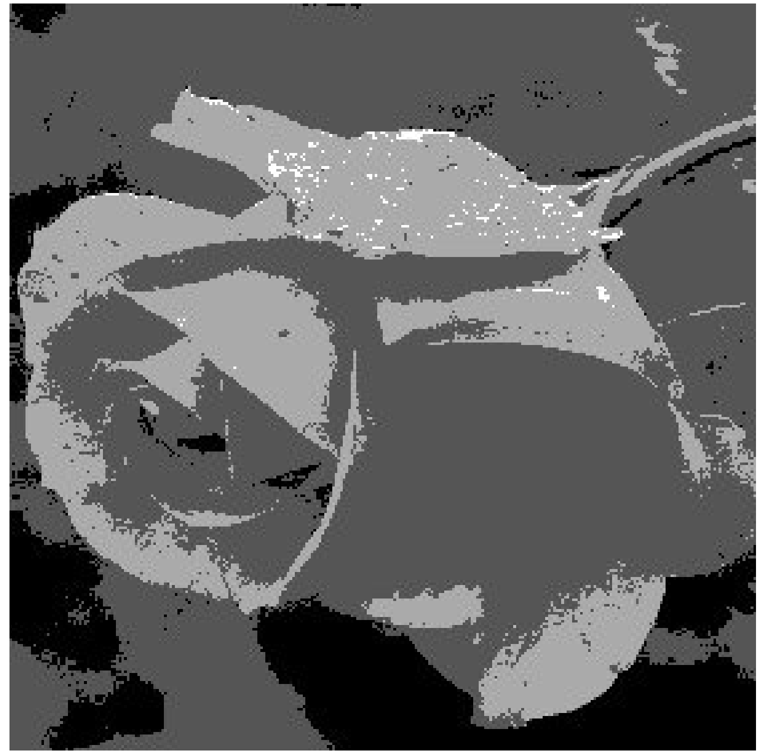
Quantizing Images

8 bit image

3 bit image

Quantizing Images

8 bit image



2 bit image

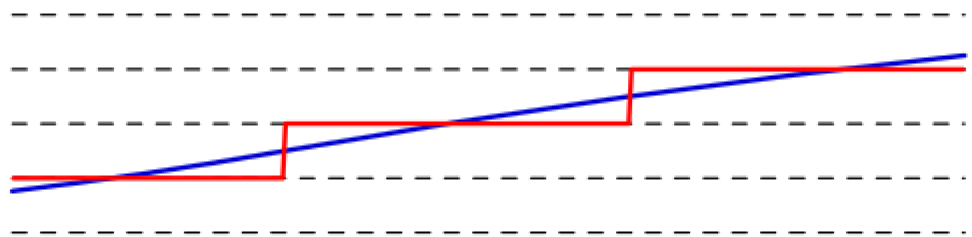
Quantizing Images

8 bit image

1 bit image

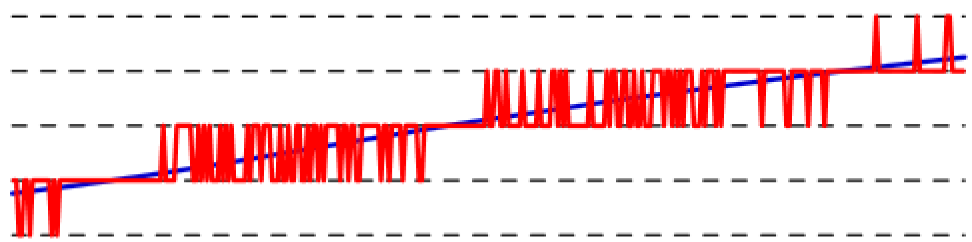
Quantization: $y = Q(x)$

Quantization: $y = Q(x)$

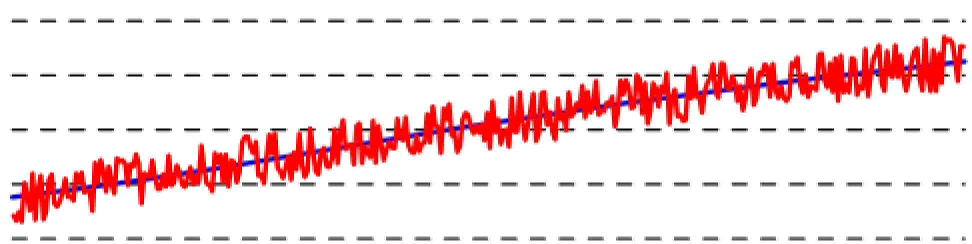


Quantization with dither: $y = Q(x + n)$

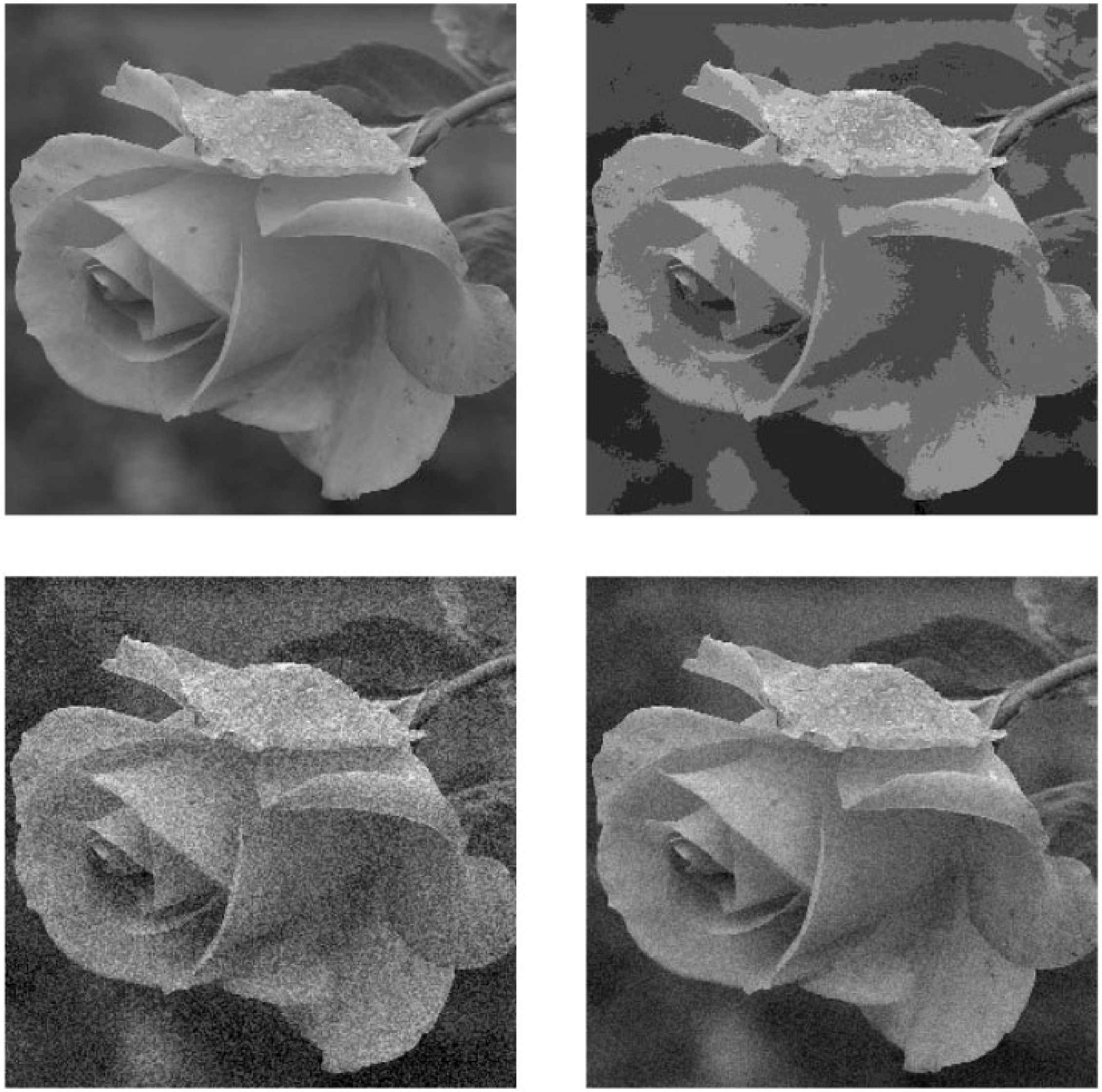
n = ±½ quantum



Quantization with Robert's technique: $y = Q(x + n) - n$



3 Bits Quantization



8 bits

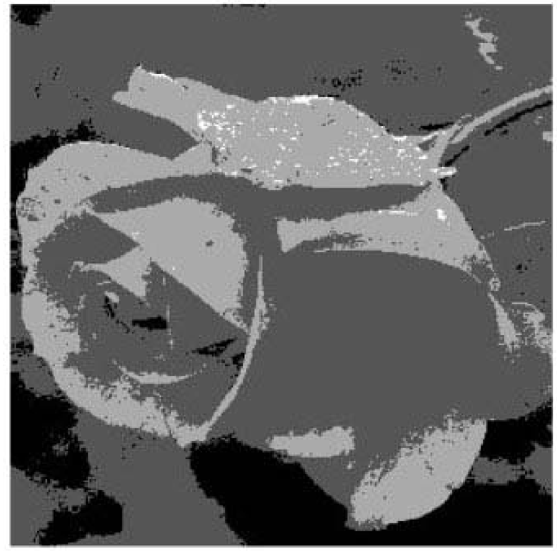
3 bits

dither

Robert's

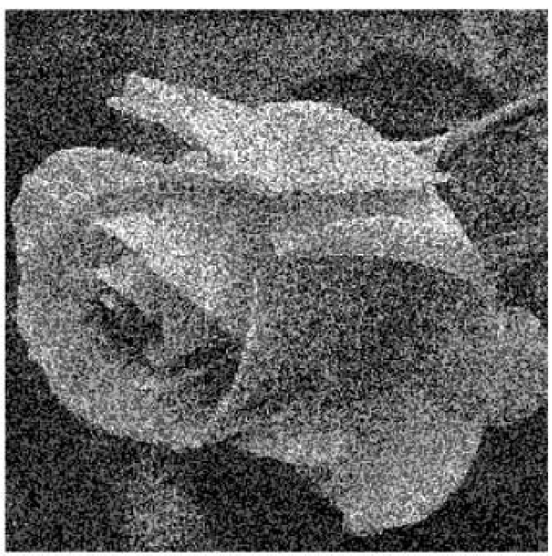
2 Bits Quantization + Noise

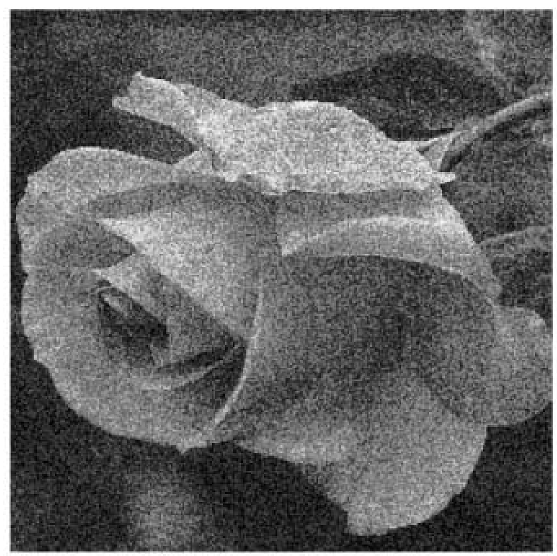
8 bits



2 bits

dither



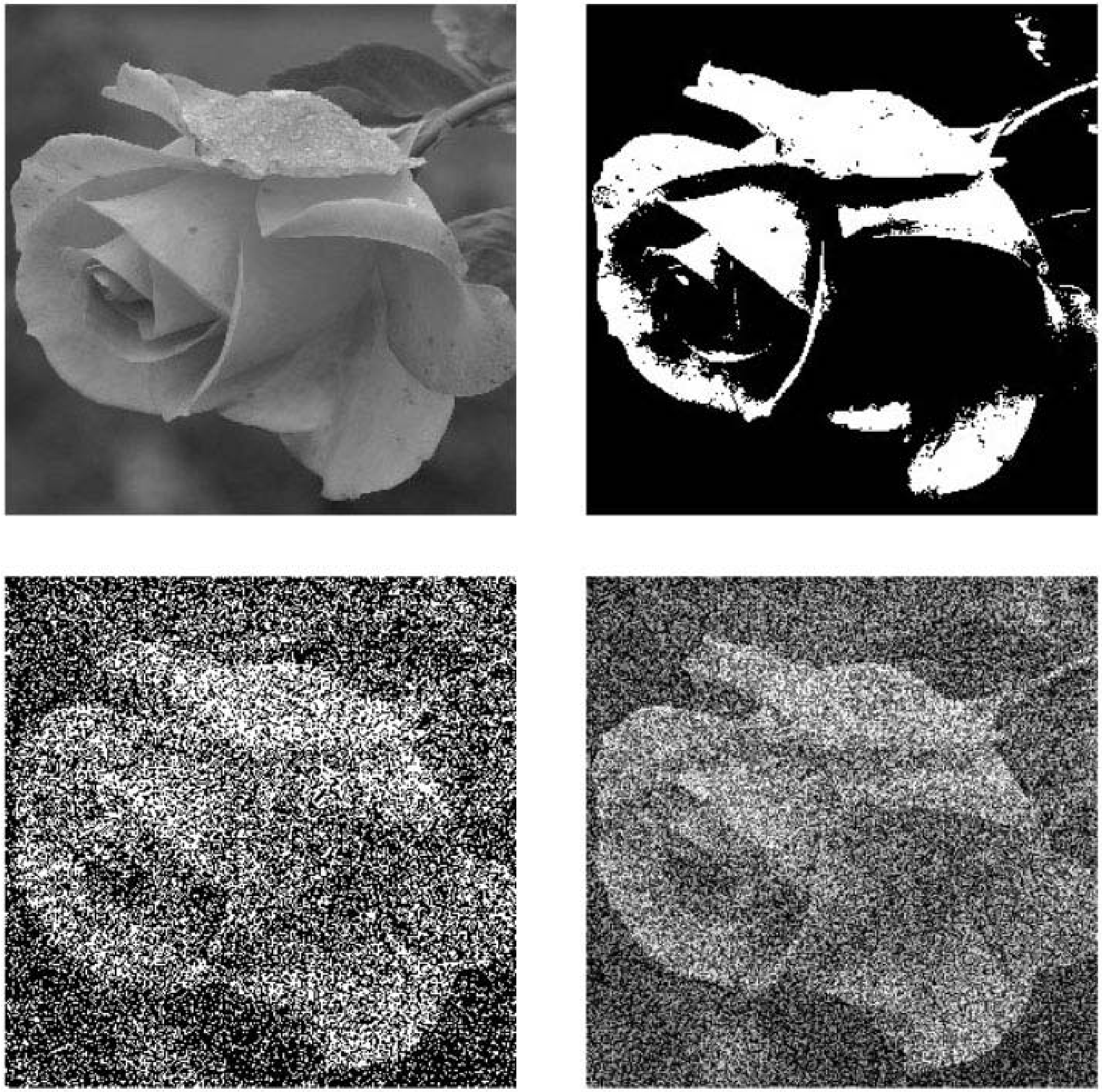


Robert's

1 Bit Quantization + Noise

8 bits

1 bit



dither

Robert's

Quantizing Colors

24 bit – 16M colors

8 bit – 256 colors

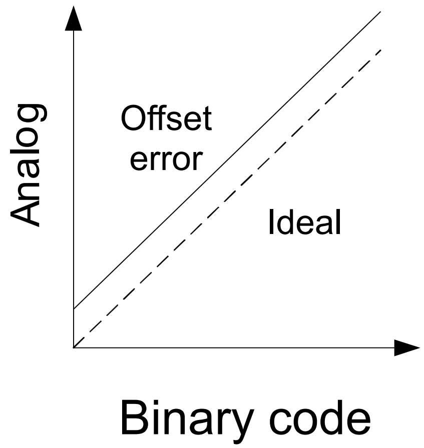
4 bit – 16 colors

Conclusions

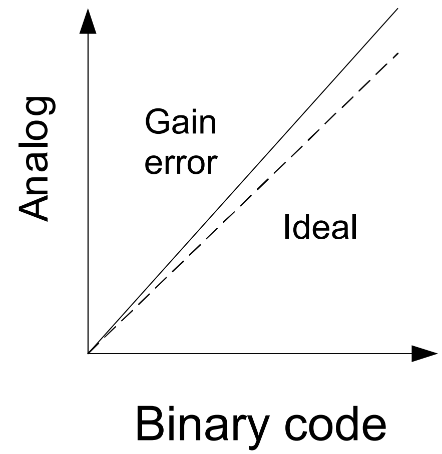
- For a given application, select the resolution that meets the design target and cost target.
- For bits means higher cost, higher power consumption
- Digital processing may help.

Non-idealities in Data Conversion

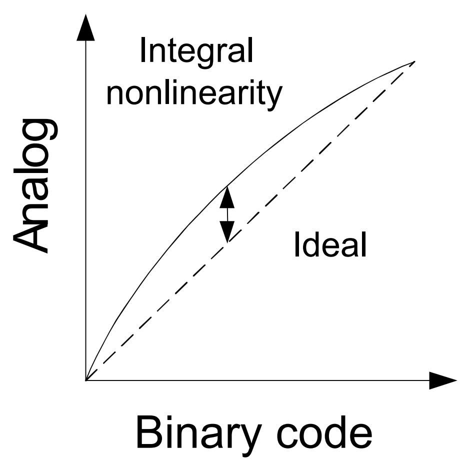
Offset – a constant voltage offset that appears at the output when the digital input is 0



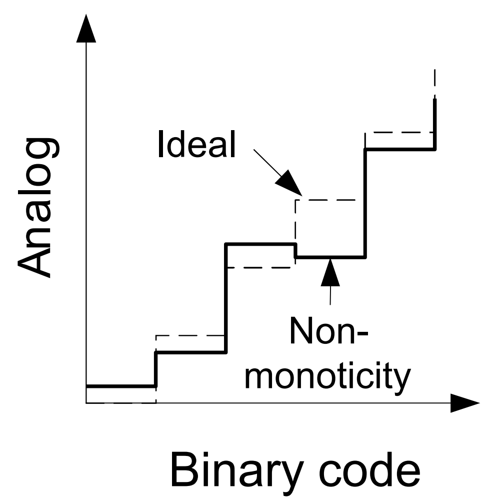
Gain error – deviation of slope from ideal value of 1



Integral Nonlinearity – maximum deviation from the ideal analog output voltage



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

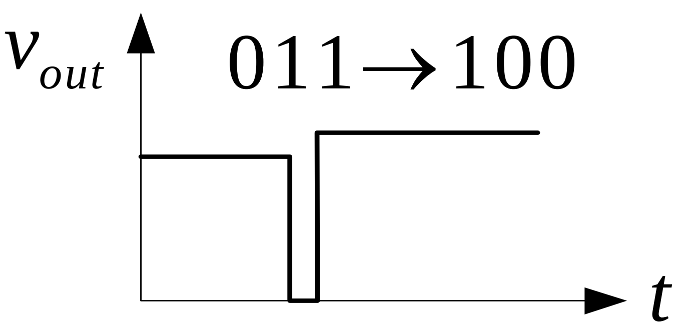


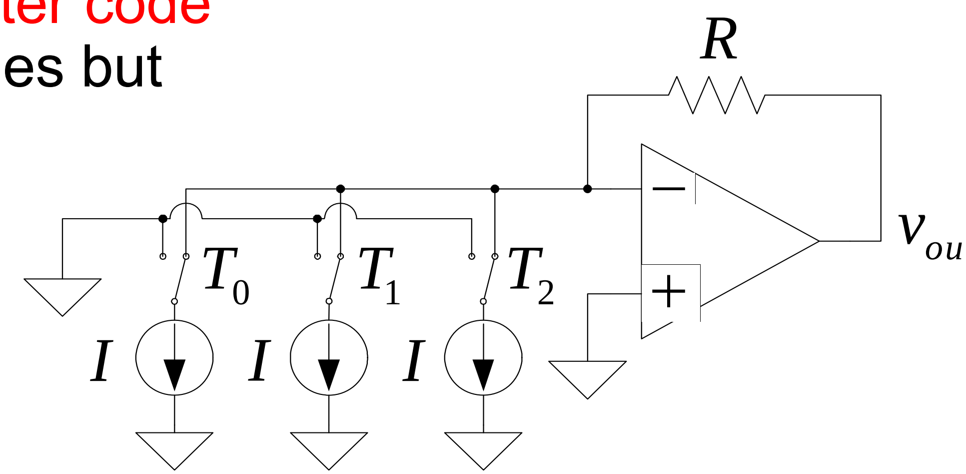
Glitching and Thermometer D/A

- 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

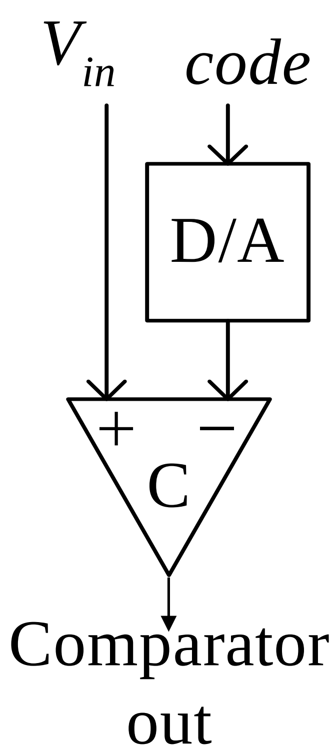


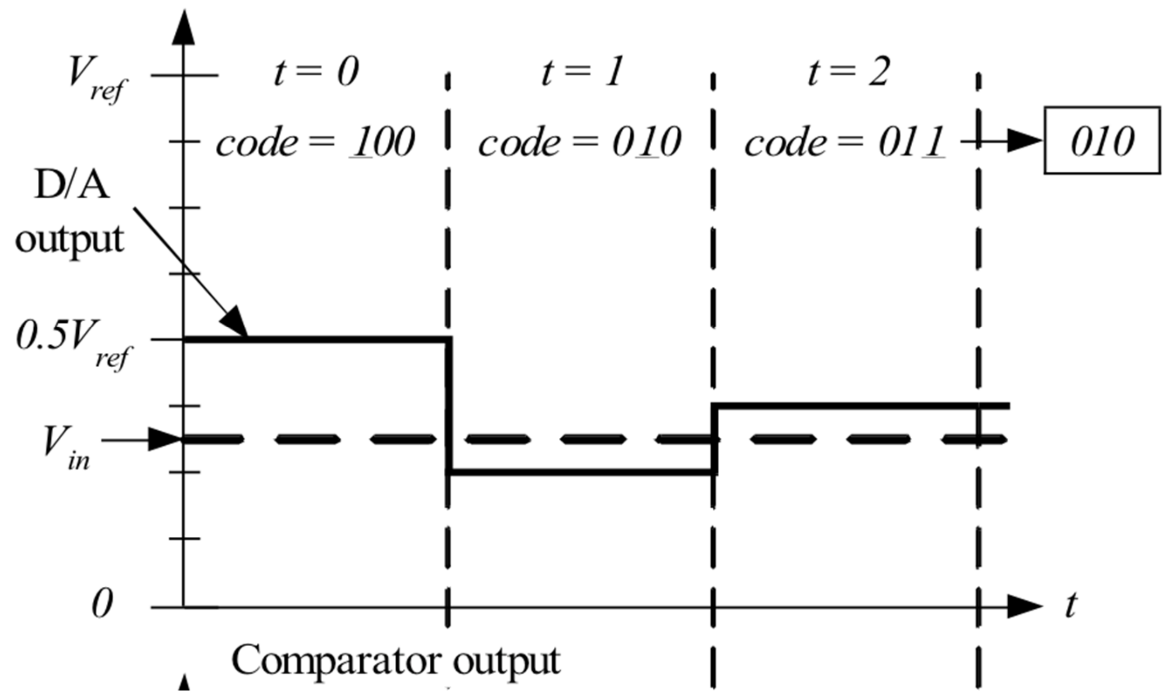


$$v_{out} = -IR(T_0 + T_1 + T_2)$$

Successive-Approximation A/D

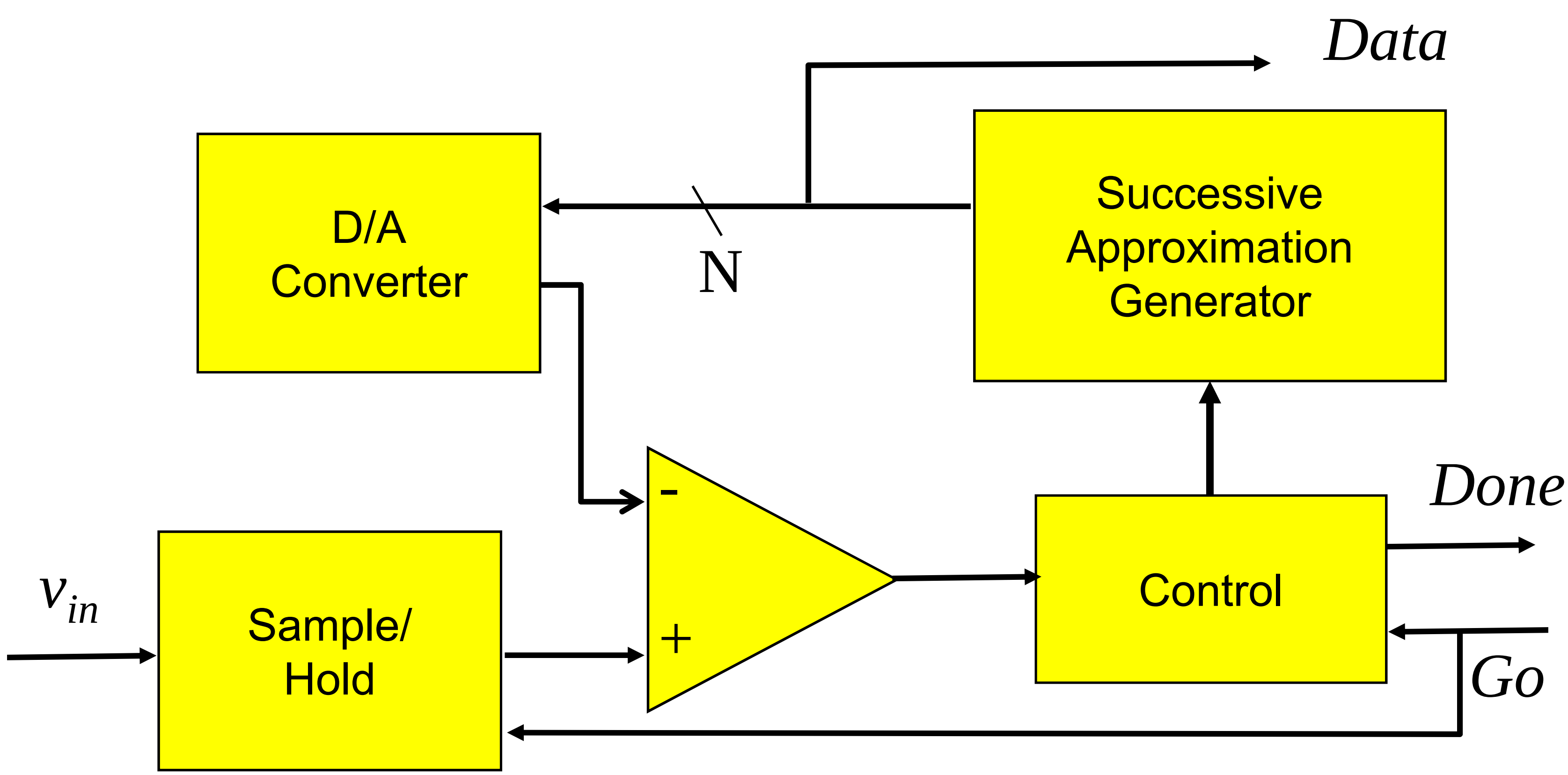
- D/A converters are typically compact and easier to design. Why not A/D convert using a D/A converter and a comparator?
- DAC generates analog voltage which is compared to the input voltage
- If DAC voltage > input voltage then set that bit; otherwise, reset that bit
- This type of ADC takes a fixed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





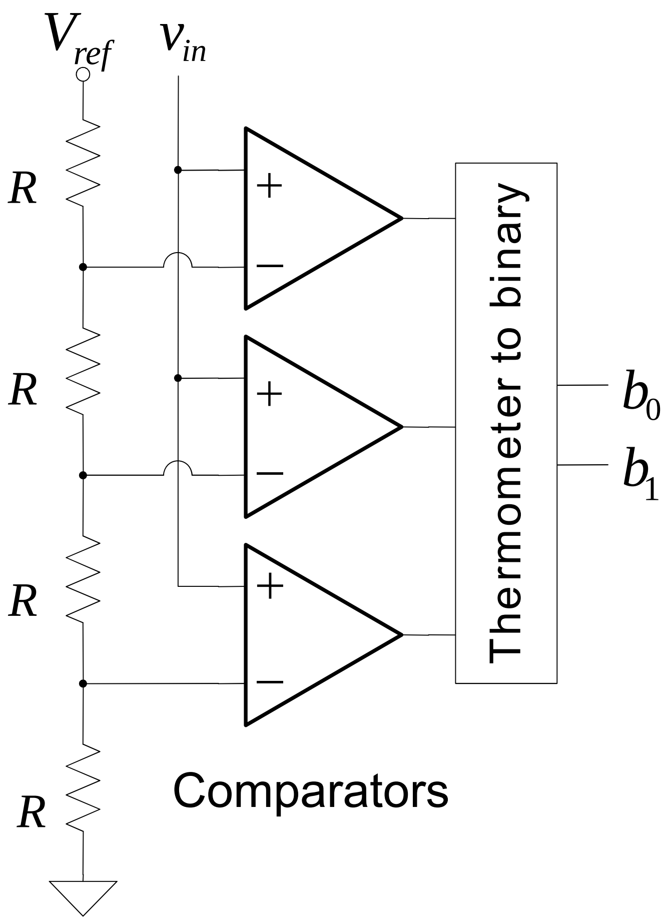
Example: 3-bit A/D conversion, 2 LSB < V_{in} < 3 LSB

Successive-Approximation A/D



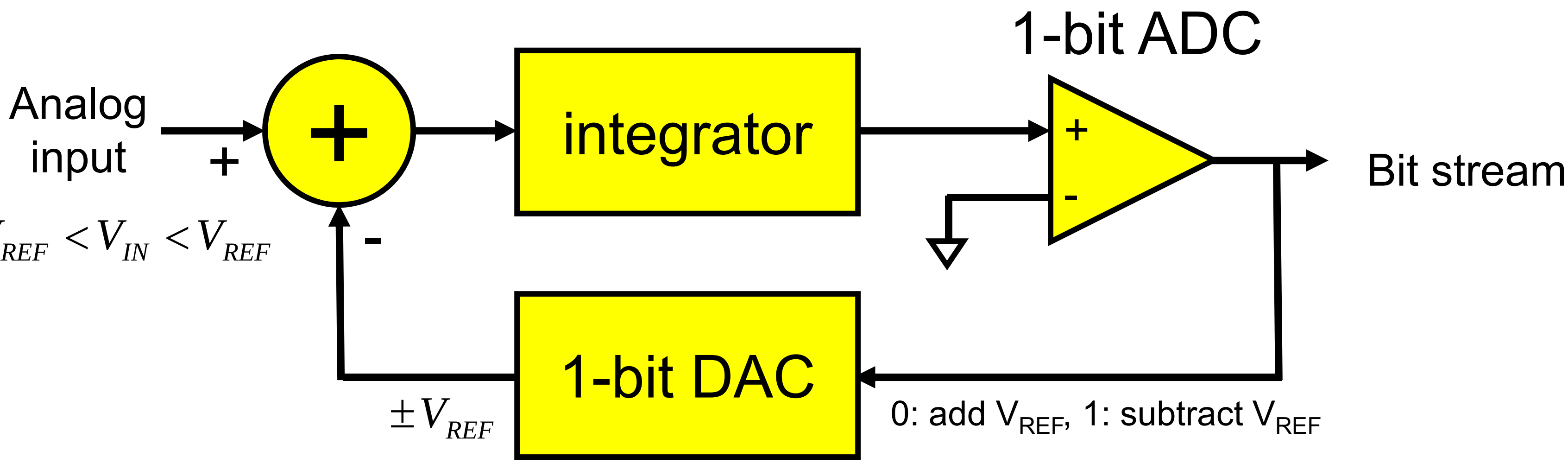
Serial conversion takes a time equal to N ($t_{D/A}$ + t_{comp})

Flash A/D Converter



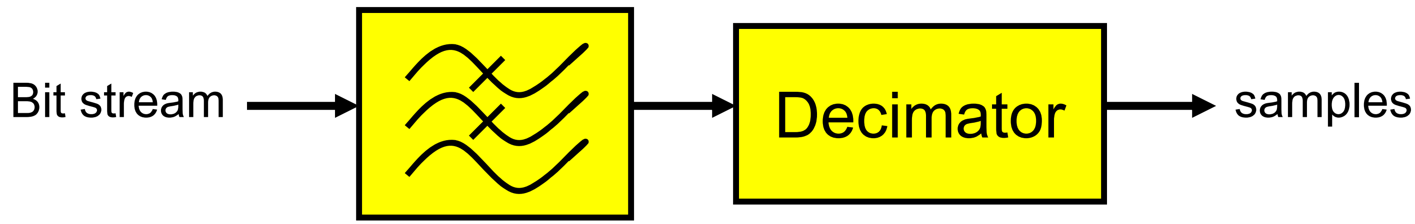
- Brute-force A/D conversion
- Simultaneously compare the analog value with every possible reference value
- Fastest method of A/D conversion
- Size scales exponentially with precision (requires 2^N comparators)

Sigma Delta ADC



$$-V_{REF} < V_{IN} < V_{REF}$$

$\pm V_{REF}$



Computes average, produces N-bit result

Only need to keep enough samples to meet Nyquist rate

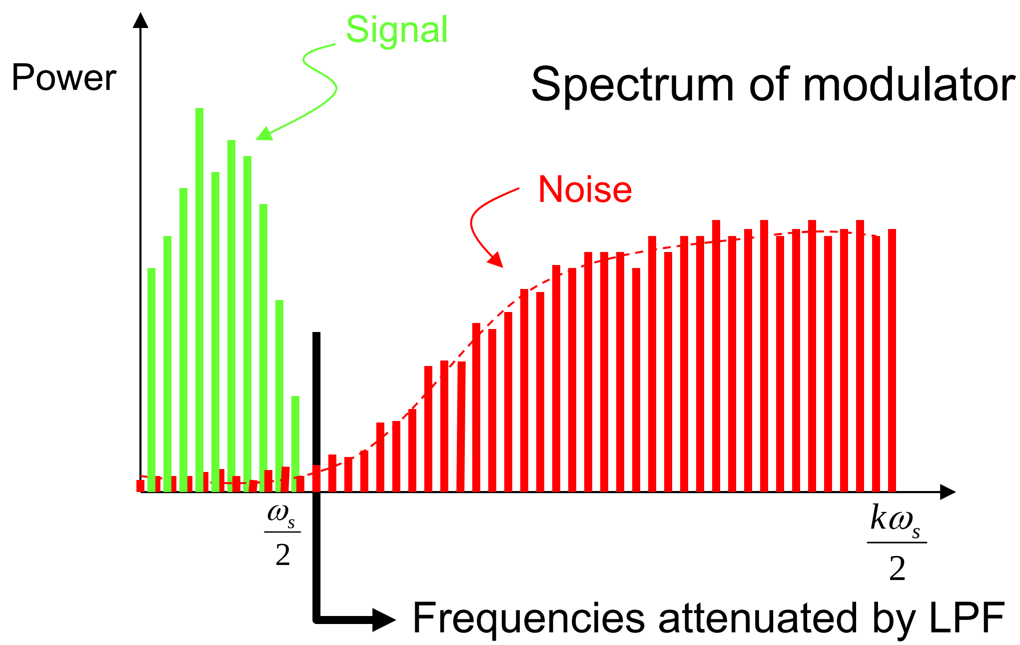
Average of bit stream (1=V_{REF}, 0=-V_{REF}) gives voltage

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

http://designtools.analog.com/dt/sdtutorial/sdtutorial.html#instructions

So, what's the big deal?

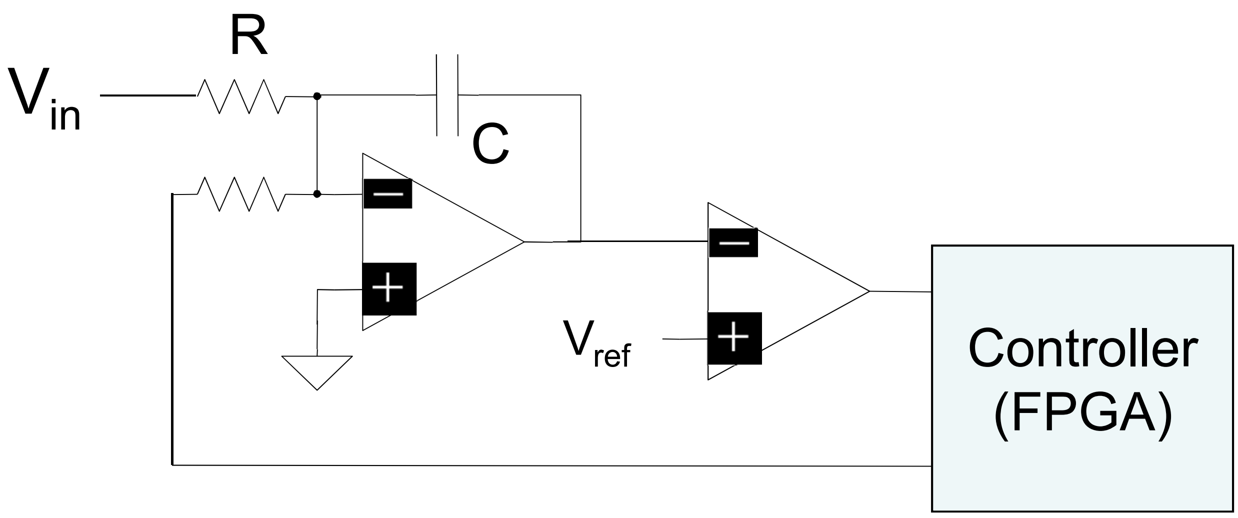
- Can be run at high sampling rates, oversampling by, say, 8 or 9 octaves for audio applications; low power implementations
- Feedback path through the integrator changes how the noise is spread across the sampling spectrum.



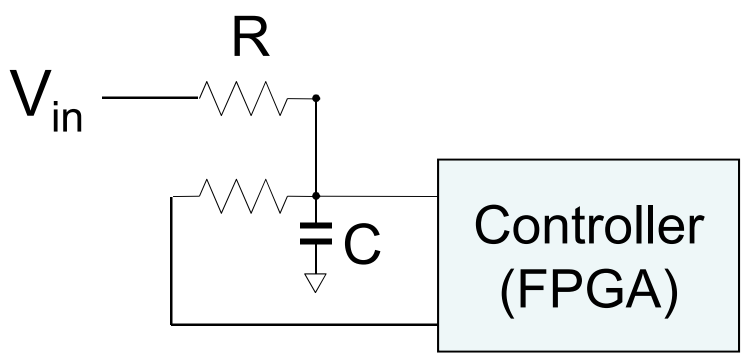
- Pushing noise power to higher frequencies means more noise is eliminated by LPF: N^{th} order ΣΔ SNR = (3+N*6)dB/octave

Sigma Delta ADC

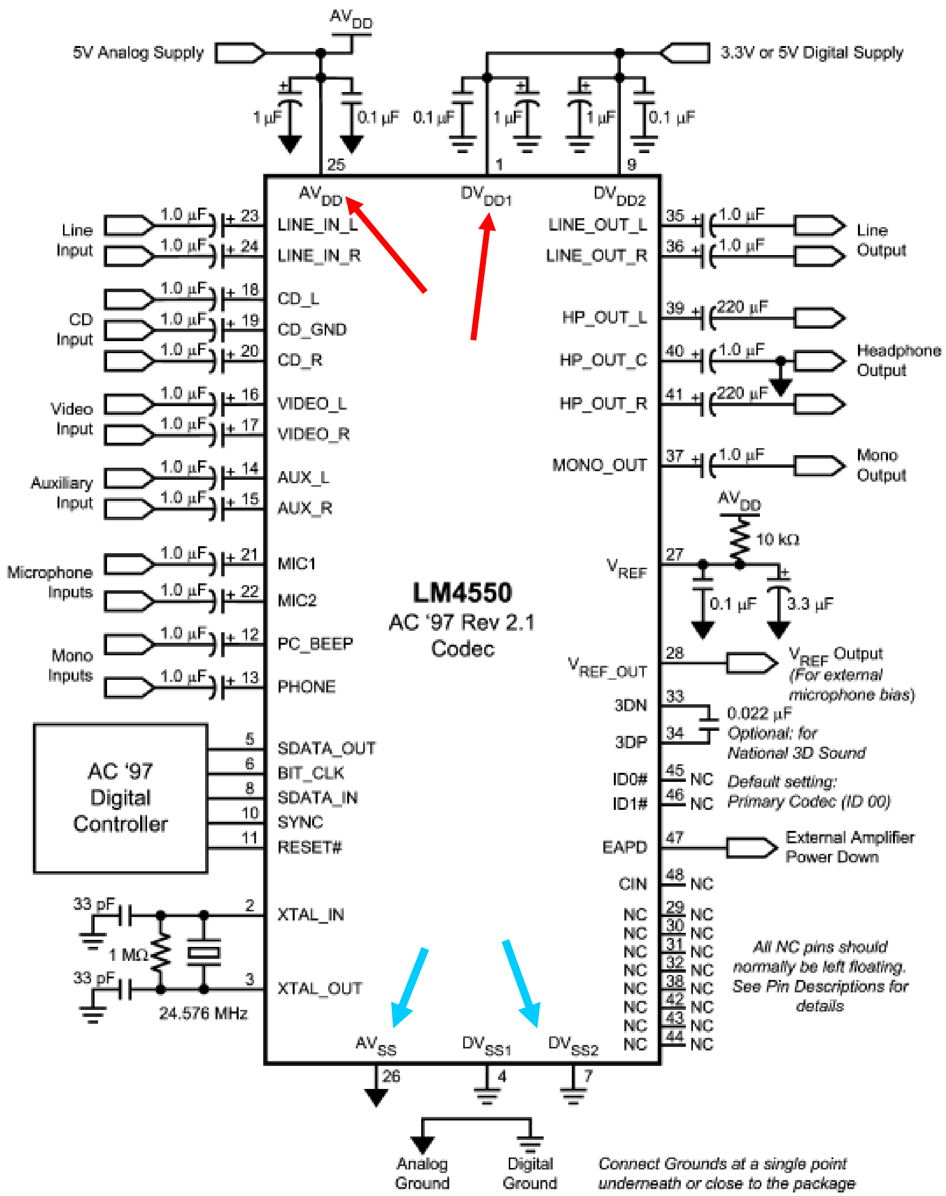
- A simple ADC:



- Poor Man's ADC:



AD Supply Voltages Consideration

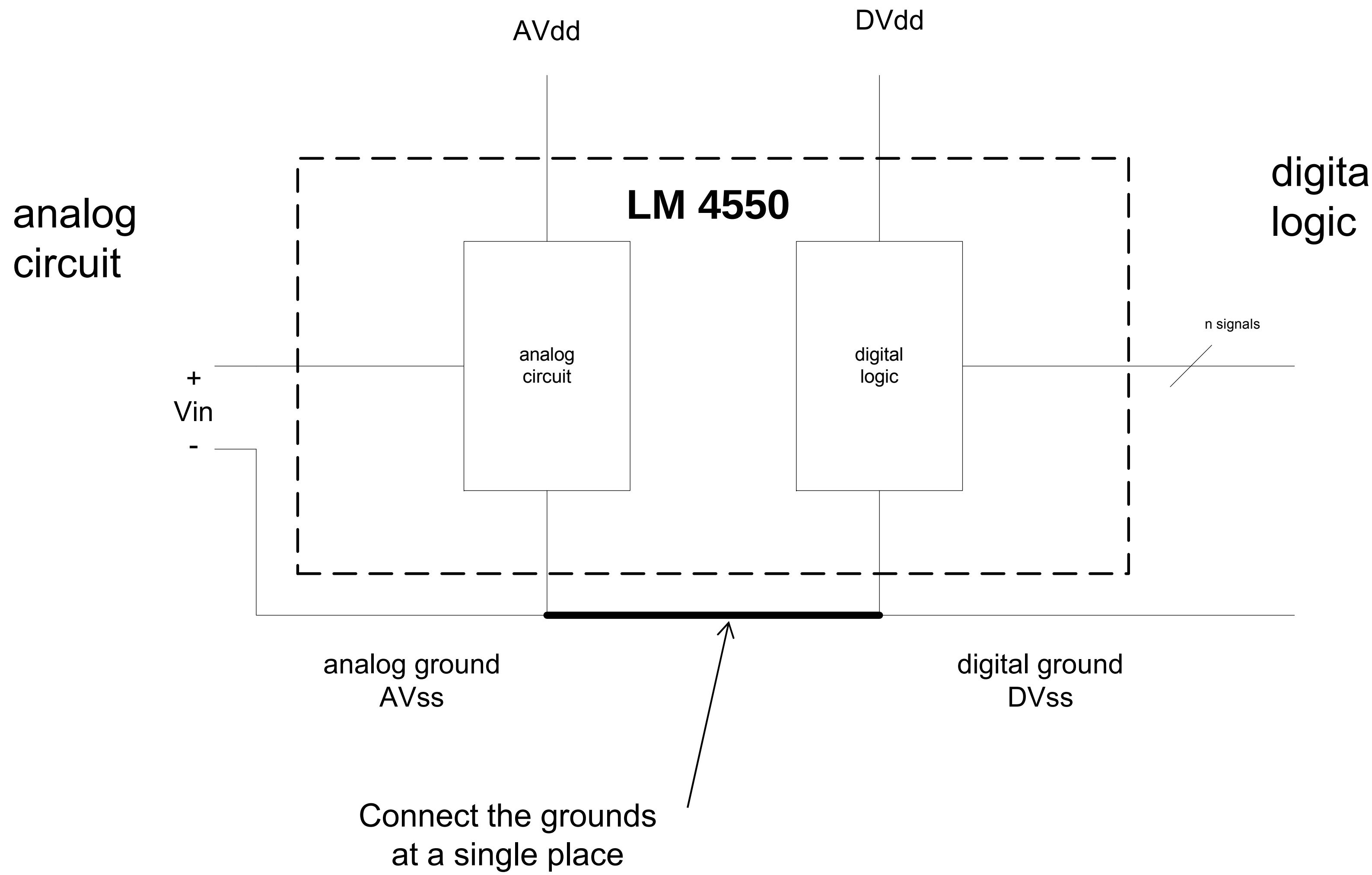


Noise caused by current spikes in fast switching digital circuits:

$$i_c = c\,\frac{dv}{dt}$$

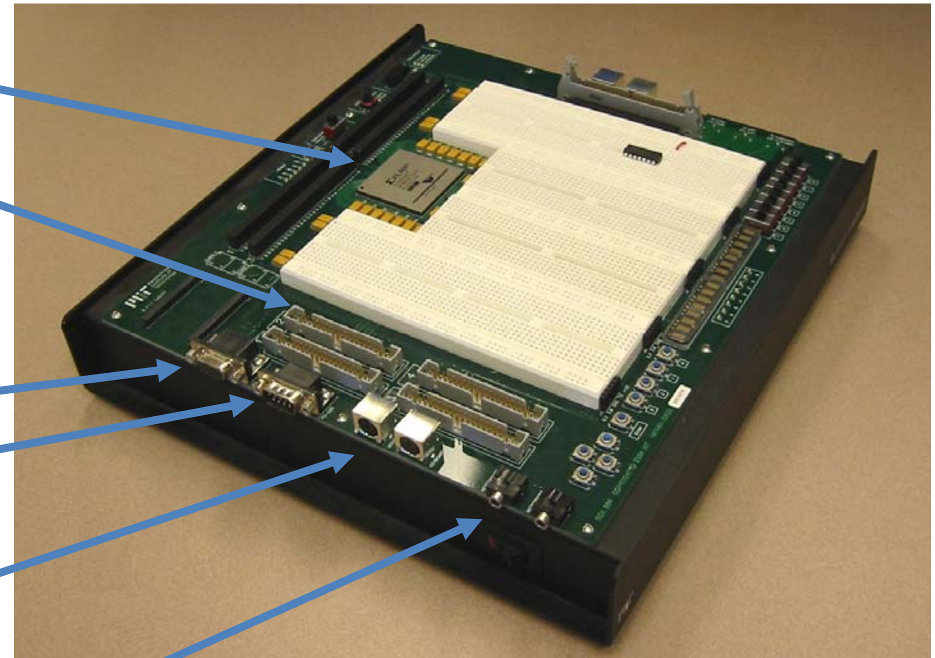
- AV$_{DD}$ Positive Analog Supply Voltage
- AV$_{SS}$ Analog Ground
- DV$_{DD}$ Positive Digital Supply Voltage
- DV$_{SS}$ Digital Ground

Digital/Analog Grounds

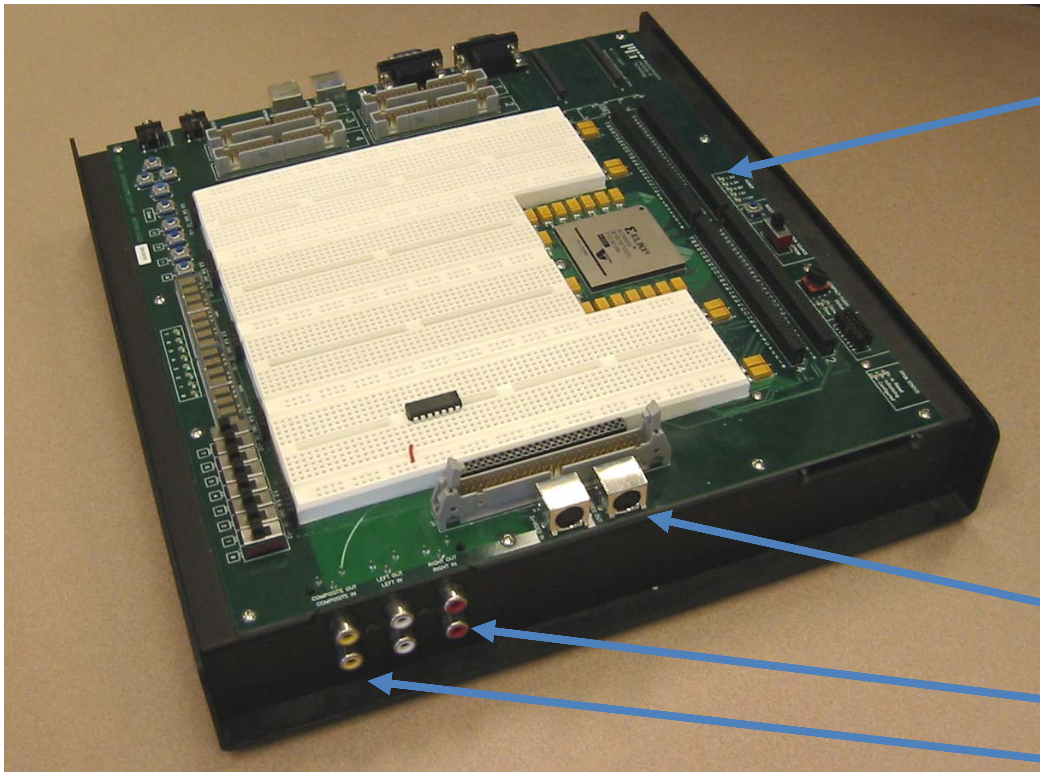


Labkit Hardware

- Xilinx FPGA
- Logic analyzer pods
  - 4 banks/pods of 16 data lines
  - (analyzerN_clock) and a 16-bit data bus (analyzerN_data[15:0]) N=1,2,3,4
- VGA video output
- RS-232 Serial IO
- PS/2 keyboard and mouse input
- AC97 audio input/output
  - Intel standard for PC audio systems
  - codec's ADCs and DACs operate at a 48kHz sample rate, with 18 bits of precision
- 128Mbits Flash memory, (2) 512k x 36 ZBT SRAM



Labkit Hardware



- Bidirectional user
  - general purpose I/O, such as connecting to devices on the breadboards
  - bidirectional (inout) signals user1[31:0] through user4[31:0]
- TV Video
  - S video input/output
  - Audio input/output
  - Composite video input/output