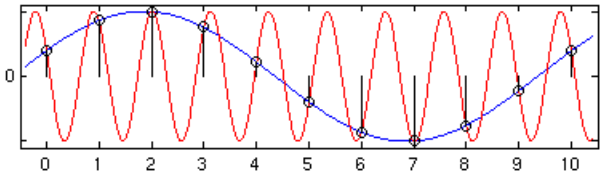
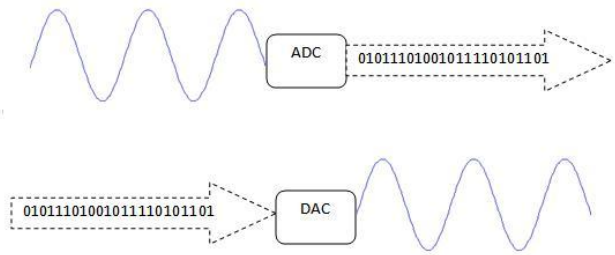
EECS 373
Design of Microprocessor-Based Systems



Ronald Dreslinski
University of Michigan



Sampling, ADCs, and DACs and more

Some slides adapted from Mark Brehob, Prabal Dutta, Jonathan Hui & Steve Reinhardt

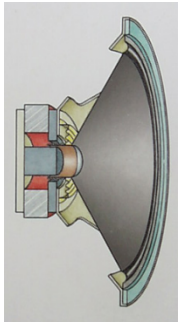
Outline

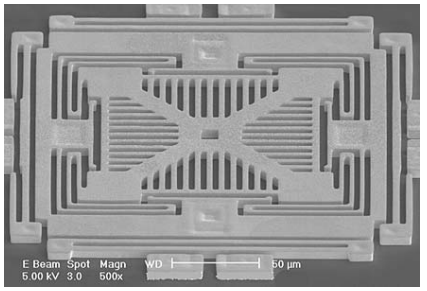
- Sampling
- ADC
- DAC

We live in an analog world

- Everything in the physical world is an analog signal
  - Sound, light, temperature, pressure

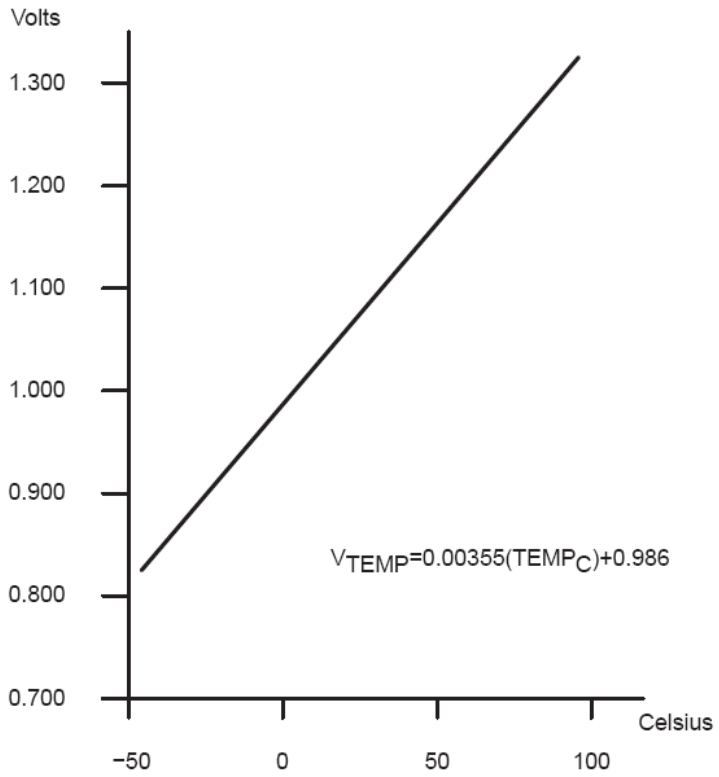
- Need to convert into electrical signals
  - Transducers: converts one type of energy to another
    - Electro-mechanical, Photonic, Electrical, …
  - Examples
    - Microphone/speaker
    - Thermocouples
    - Accelerometers





Transducers convert one form of energy into another

- Transducers
  - Allow us to convert physical phenomena to a voltage potential in a well-defined way.

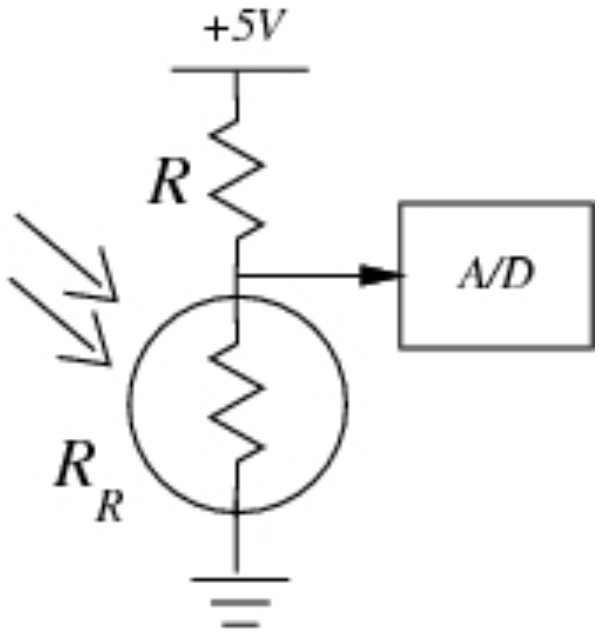


A transducer is a device that converts one type of energy to another. The conversion can be to/from electrical, electro-mechanical, electromagnetic, photonic, photovoltaic, or any other form of energy. While the term transducer commonly implies use as a sensor/detector, any device which converts energy can be considered a transducer. – *Wikipedia.*

Convert light to voltage with a CdS photocell

$V_{signal} = (+5V)\ R_R/(R + R_R)$

- Choose $R=R_R$ at median of intended range
- Cadmium Sulfide (CdS)
- Cheap, low current
- $t_{RC} = (R+R_R)*C_l$
  - Typically R~50-200kΩ
  - C~20pF
  - So, t_{RC}~20-80uS
  - f_{RC} ~ 10-50kHz



Source: Forrest Brewer

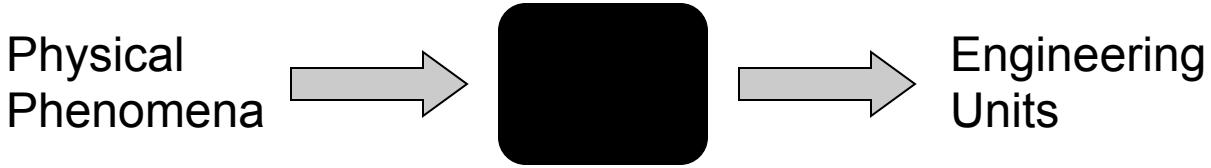
Many other common sensors (some digital)

- Force
  - strain gauges - foil, conductive ink
  - conductive rubber
  - rheostatic fluids
    - Piezorestive (needs bridge)
  - piezoelectric films
  - capacitive force
    - Charge source
- Sound
  - Microphones
    - Both current and charge versions
  - Sonar
    - Usually Piezoelectric
- Position
  - microswitches
  - shaft encoders
  - gyros
- Acceleration
  - MEMS
  - Pendulum
- Monitoring
  - Battery-level
    - voltage
  - Motor current
    - Stall/velocity
  - Temperature
    - Voltage/Current Source
- Field
  - Antenna
  - Magnetic
    - Hall effect
    - Flux Gate
- Location
  - Permittivity
  - Dielectric

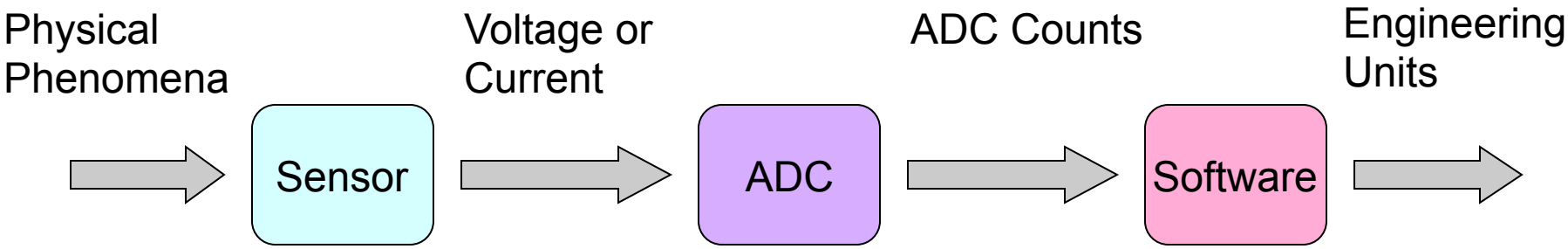
Source: Forrest Brewer

Going from analog to digital

- What we want

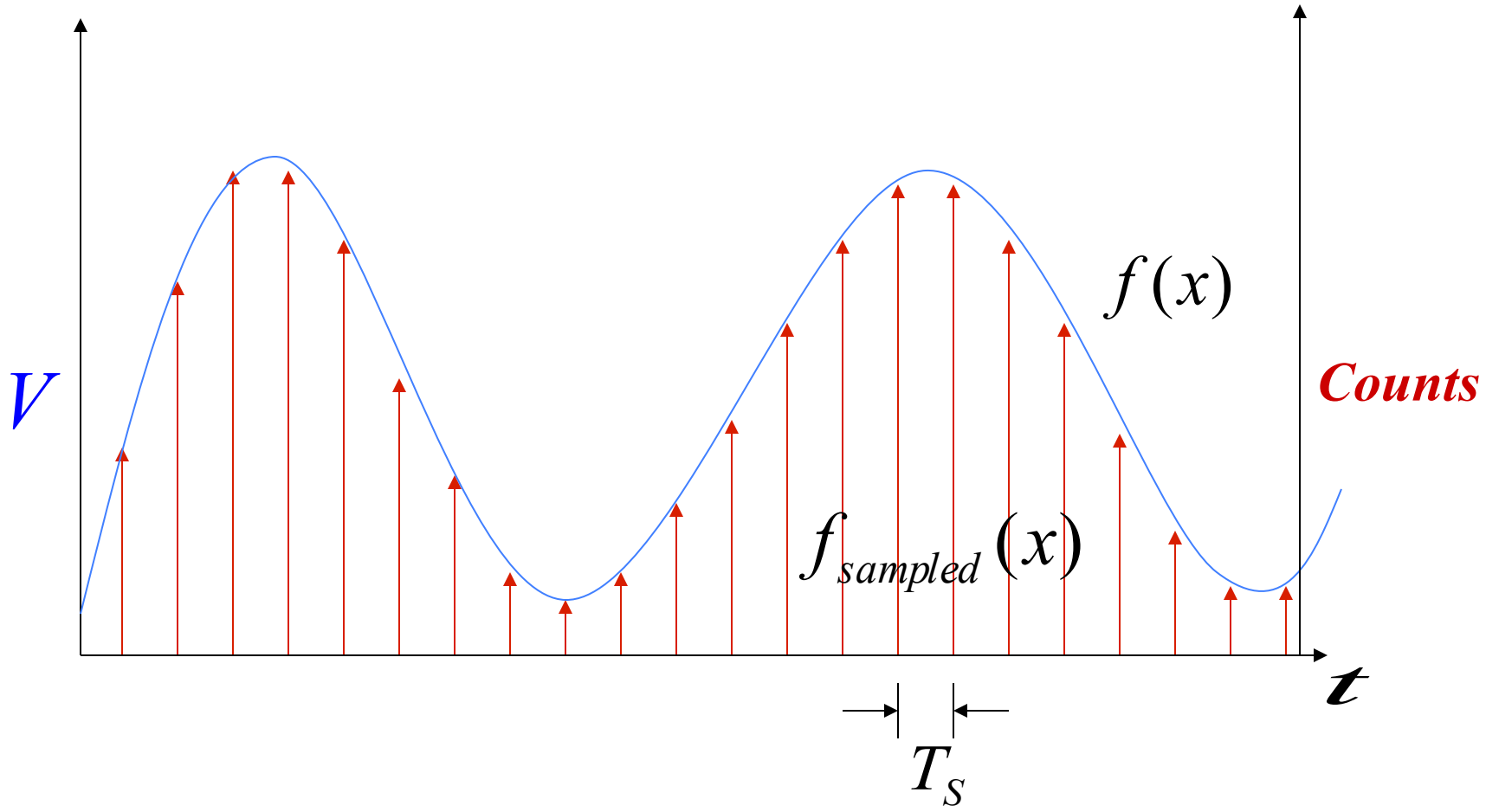


- How we have to get there

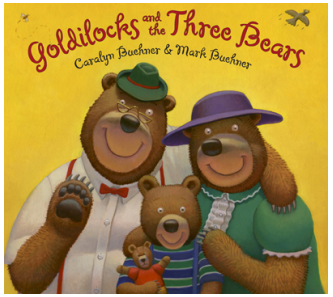


Representing an analog signal digitally

- How do we represent an analog signal?
  - As a time series of discrete values
    → On MCU: read the ADC data register periodically

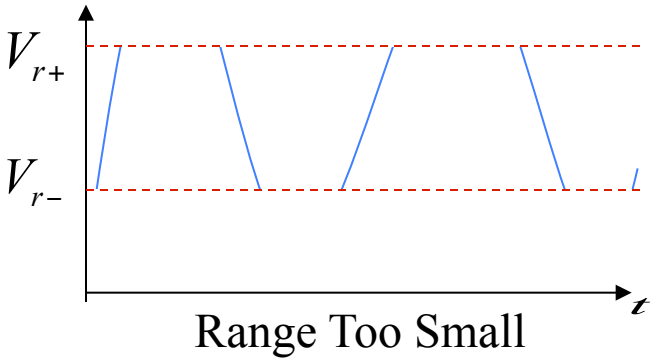


Choosing the horizontal range

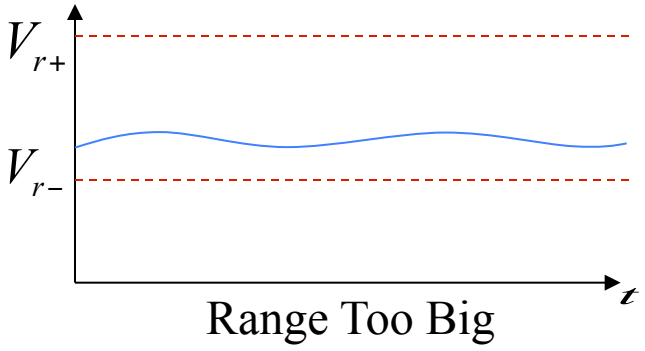


- What do the sample values represent?
  - Some fraction within the range of values

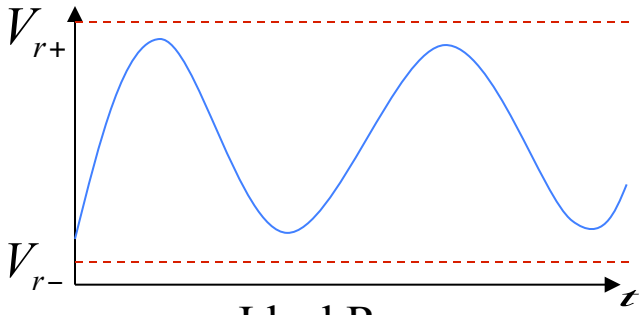
    → ***What range to use?***



Range Too Small



Range Too Big



Ideal Range

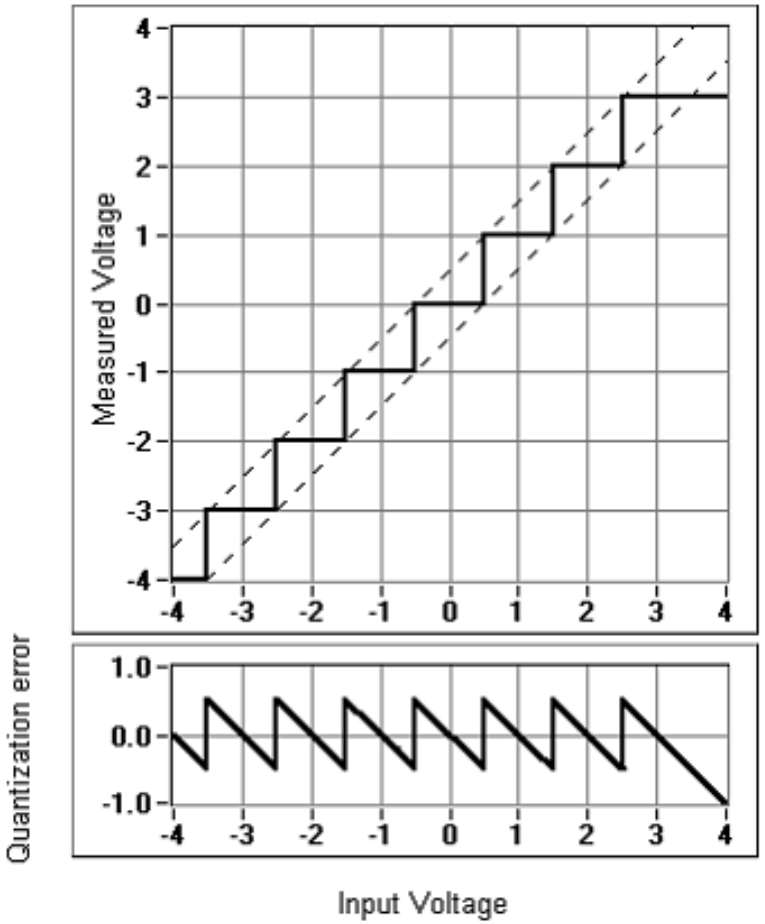
Choosing the horizontal granularity

- Resolution
  - Number of discrete values that represent a range of analog values
  - MSP430: 12-bit ADC
    - 4096 values
    - Range / 4096 = Step

  Larger range ➔ less information

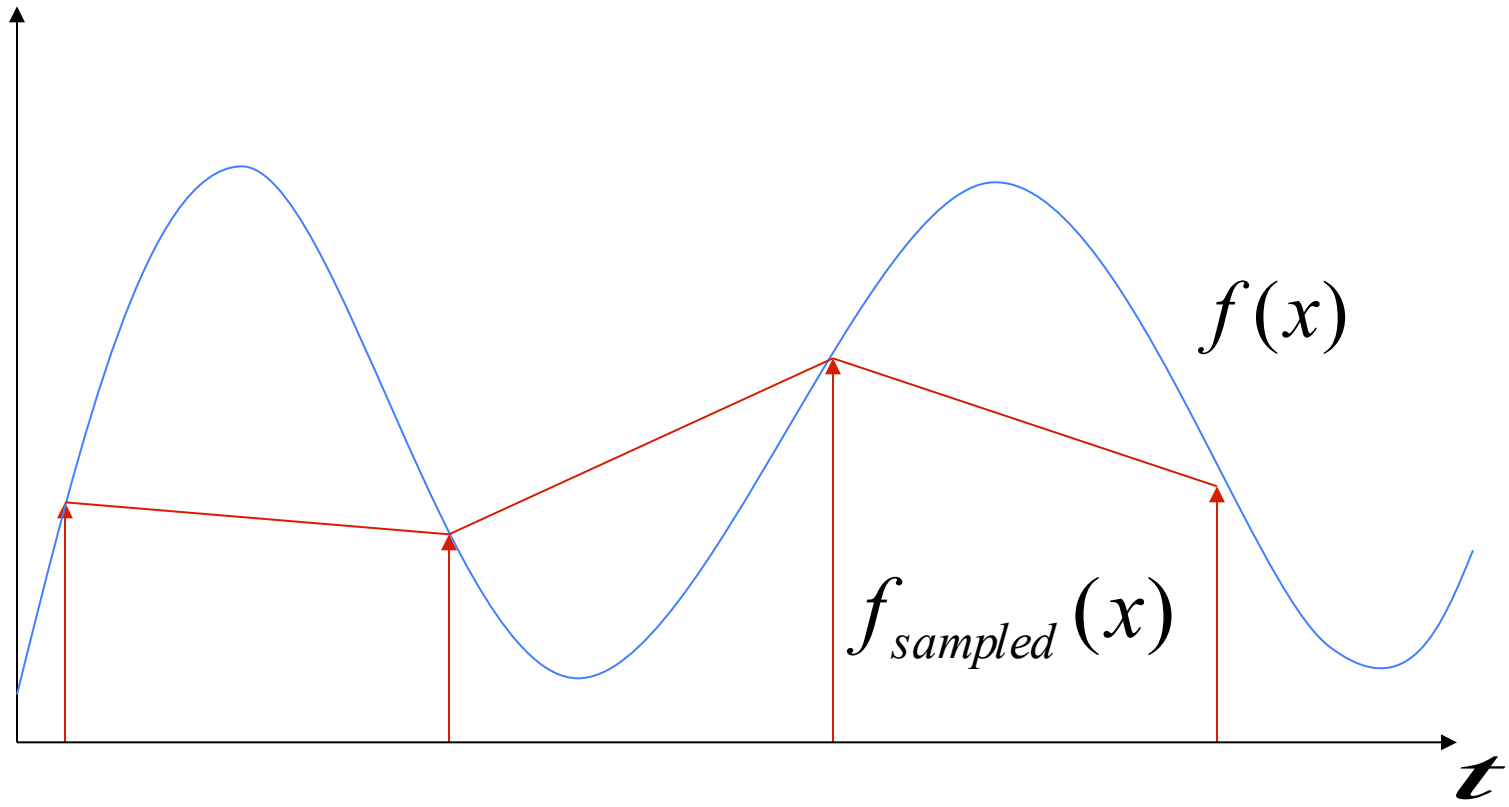
- Quantization Error
  - How far off discrete value is from actual
  - ½ LSB → Range / 8192

  Larger range ➔ larger error



Choosing the sample rate

- What sample rate do we need?
  - Too little: we can't reconstruct the signal we care about
  - Too much: waste computation, energy, resources



Shannon-Nyquist sampling theorem

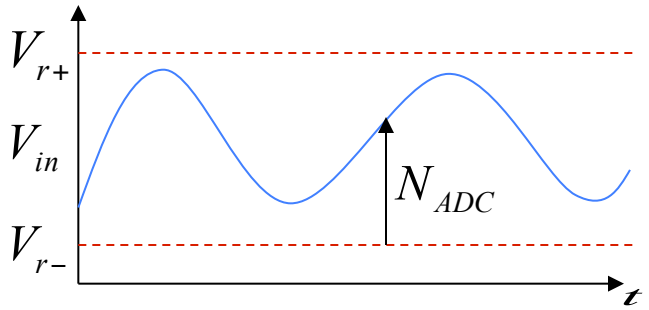
- *If a continuous-time signal $f(x)$ contains no frequencies higher than $f_{\max}$, it can be completely determined by discrete samples taken at a rate:*

$$f_{\text{samples}} > 2 f_{\max}$$

- Example:
  - Humans can process audio signals 20 Hz – 20 KHz
  - Audio CDs: sampled at 44.1 KHz

Converting between voltages, ADC counts, and engineering units

- Converting: ADC counts ⇔ Voltage



$$N_{ADC} = 4095 \times \frac{V_{in} - V_{r-}}{V_{r+} - V_{r-}}$$

$$V_{in} = N_{ADC} \times \frac{V_{r+} - V_{r-}}{4095}$$

- Converting: Voltage ⇔ Engineering Units

$$V_{\text{TEMP}} = 0.00355(\text{TEMP}_{\text{C}}) + 0.986$$

$$\text{TEMP}_{\text{C}} = \frac{V_{\text{TEMP}} - 0.986}{0.00355}$$

A note about sampling and arithmetic*

- Converting values in fixed-point MCUs

$$V_{\text{TEMP}} = N_{ADC} \times \frac{V_{r+} - V_{r-}}{4095} \qquad \text{TEMP}_{\text{C}} = \frac{V_{\text{TEMP}} - 0.986}{0.00355}$$

```
float vtemp = adccount/4095 * 1.5;
float tempc = (vtemp-0.986)/0.00355;
```

→ vtemp = 0! Not what you intended, even when vtemp is a float!

→ tempc = -277 C

- Fixed point operations
  - Need to worry about underflow and overflow

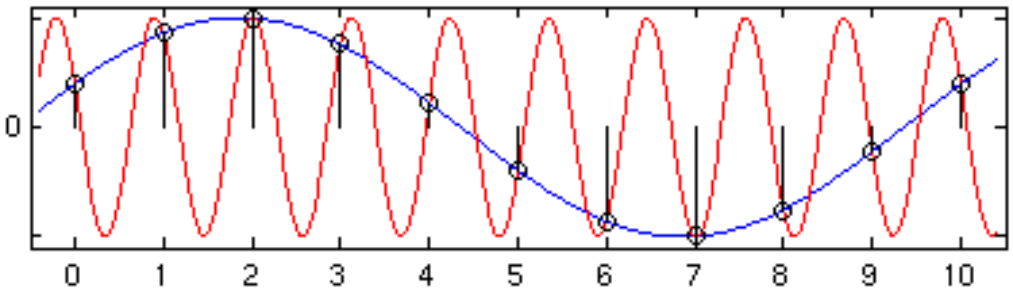
- Floating point operations
  - They can be costly on the node

Try it out for yourself…

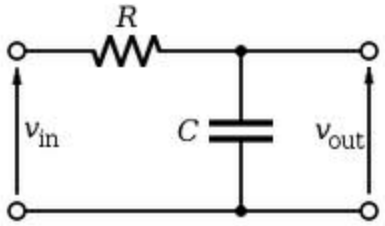

```
$ cat arithmetic.c
#include <stdio.h>

int main() {

  int adccount = 2048;
  float vtemp;
  float tempc;

  vtemp = adccount/4095 * 1.5;
  tempc = (vtemp-0.986)/0.00355;

  printf("vtemp: %f\n", vtemp);
  printf("tempc: %f\n", tempc);
}

$ gcc arithmetic.c

$ ./a.out
vtemp: 0.000000
tempc: -277.746490
```

Use anti-aliasing filters on ADC inputs to ensure that Shannon-Nyquist is satisfied

- Aliasing
  - Different frequencies are indistinguishable when they are sampled.



- Condition the input signal using a low-pass filter
  - Removes high-frequency components
  - (a.k.a. anti-aliasing filter)

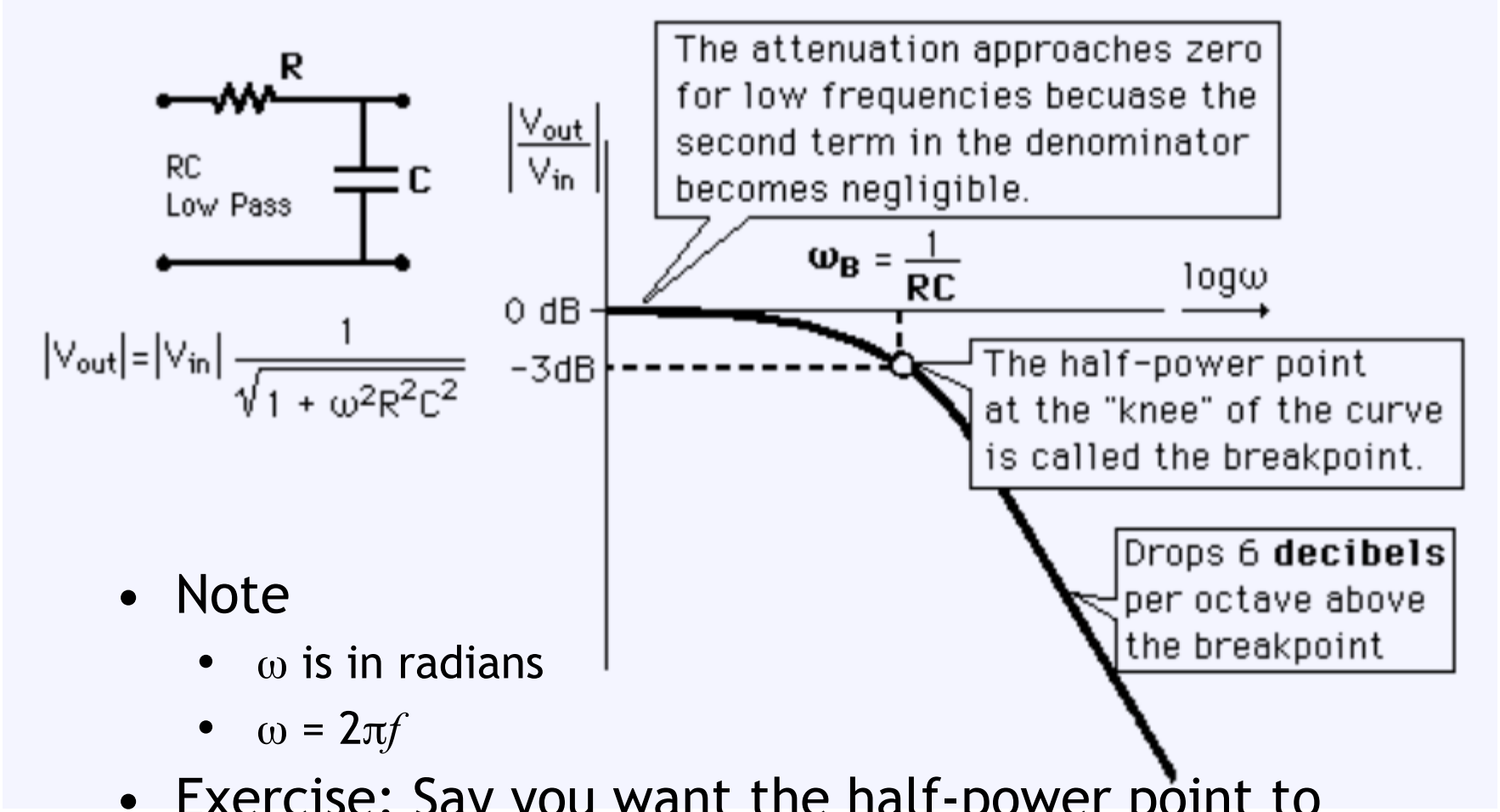


Do I really need to condition my input signal?

- Short answer: Yes.

- Longer answer: Yes, but sometimes it's already done for you.
  - Many (most?) ADCs have a pretty good analog filter built in.
  - Those filters typically have a cut-off frequency just above ½ their ***maximum*** sampling rate.
    - Which is great if you are using the maximum sampling rate, less useful if you are sampling at a slower rate.

Designing the anti-aliasing filter

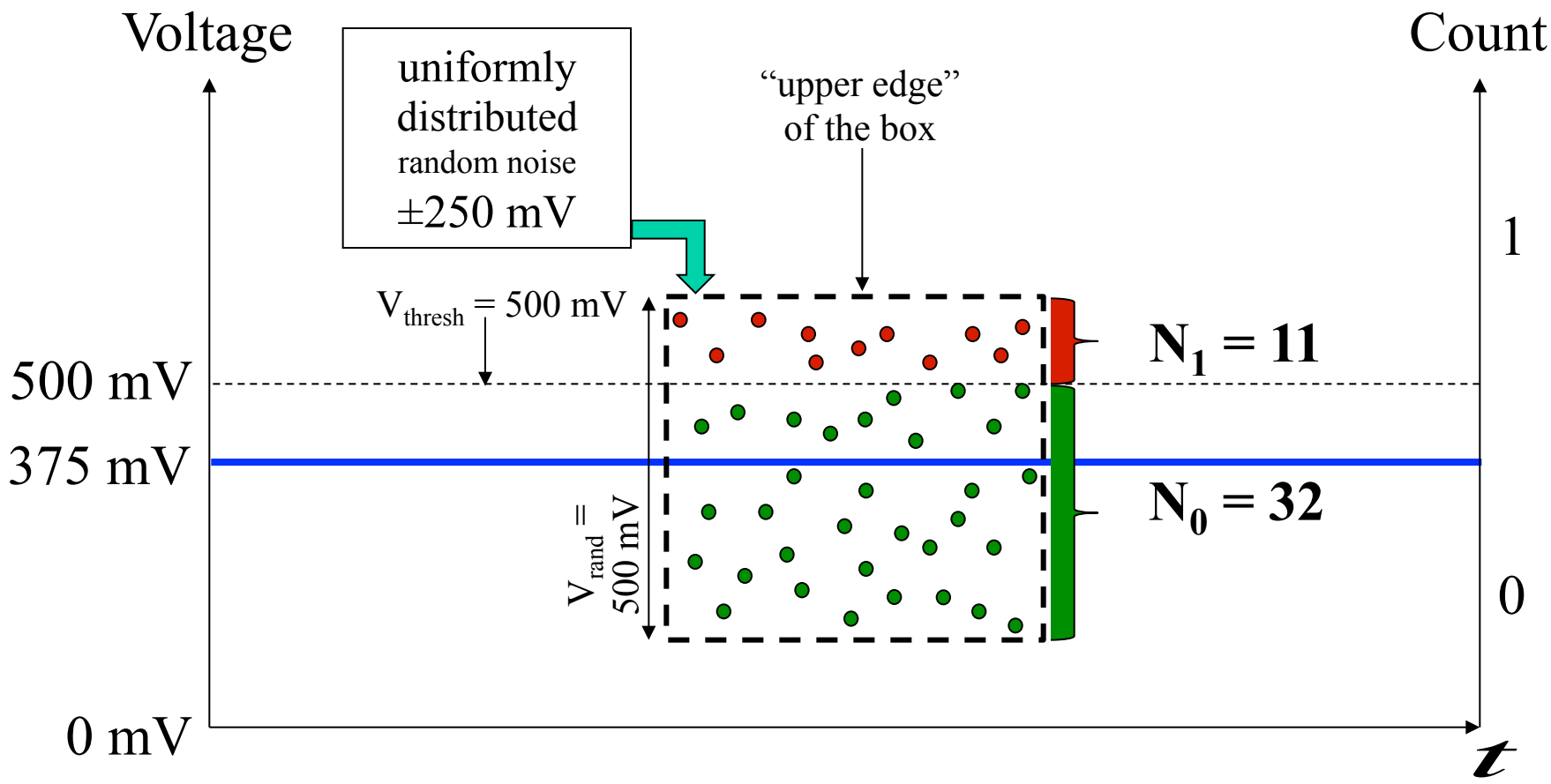


- Note
  - ω is in radians
  - $\omega = 2\pi f$
- Exercise: Say you want the half-power point to be at 30Hz and you have a 0.1 μF capacitor. How big of a resistor should you use?

Oversampling

- One interesting trick is that you can use oversampling to help reduce the impact of quantization error.
  - Let's look at an example of oversampling plus dithering to get a 1-bit converter to do a much better job…

Oversampling a 1-bit ADC w/ noise & dithering (cont)



Note:

N_1 is the # of ADC counts that = 1 over the sampling window

N_0 is the # of ADC counts that = 0 over the sampling window

Oversampling a 1-bit ADC w/ noise & dithering (cont)

- How to get more than 1-bit out of a 1-bit ADC?
- Add some noise to the input
- Do some math with the output
- Example
  - 1-bit ADC with 500 mV threshold
  - Vin = 375 mV → ADC count = 0
  - Add ±250 mV uniformly distributed random noise to Vin
  - Now, roughly
    - 25% of samples (N_1) ≥ 500 mV → ADC count = 1
    - 75% of samples (N_0) < 500 mV → ADC count = 0
  - So, the "upper edge" of the box equals
    - V_{thresh} + N1/(N1+N0) * V_{rand} = 0.5 + 11/(11+32)*0.5 = 0.628 V
  - Middle of box (where our "signal" of 375 mV sits) equals
    - 0.628 V – V_{rand}/2 = 0.628 V – 0.25 = <u>0.378 V</u>
  - Real value is 0.375 V, so our estimate has < 1% error!

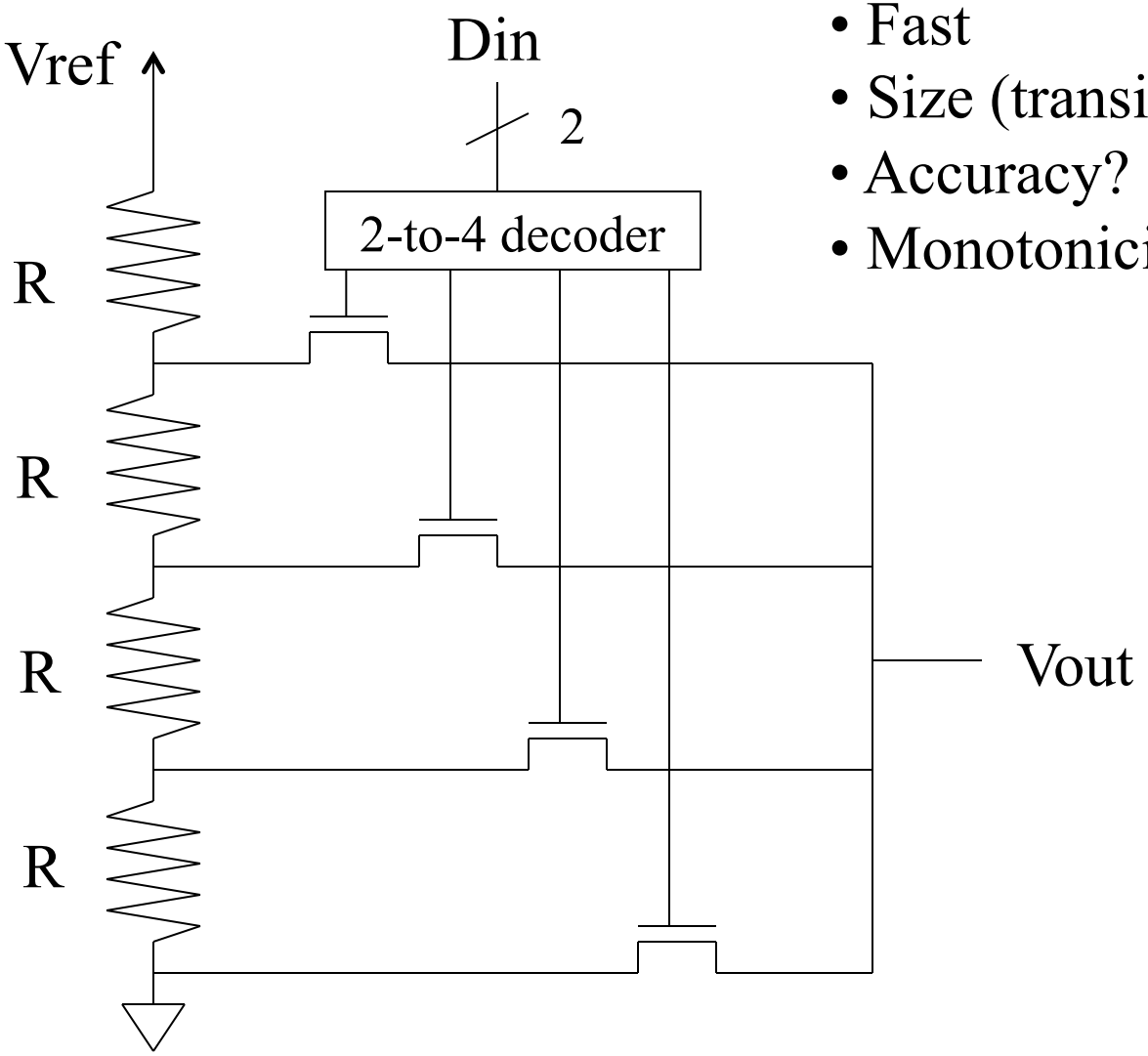
Lots of other issues

- Might need anti-imaging filter
- Cost and power play a role
- Might be able to avoid analog all together
  - Think PWM when dealing with motors…

How do ADCs and DACs work?

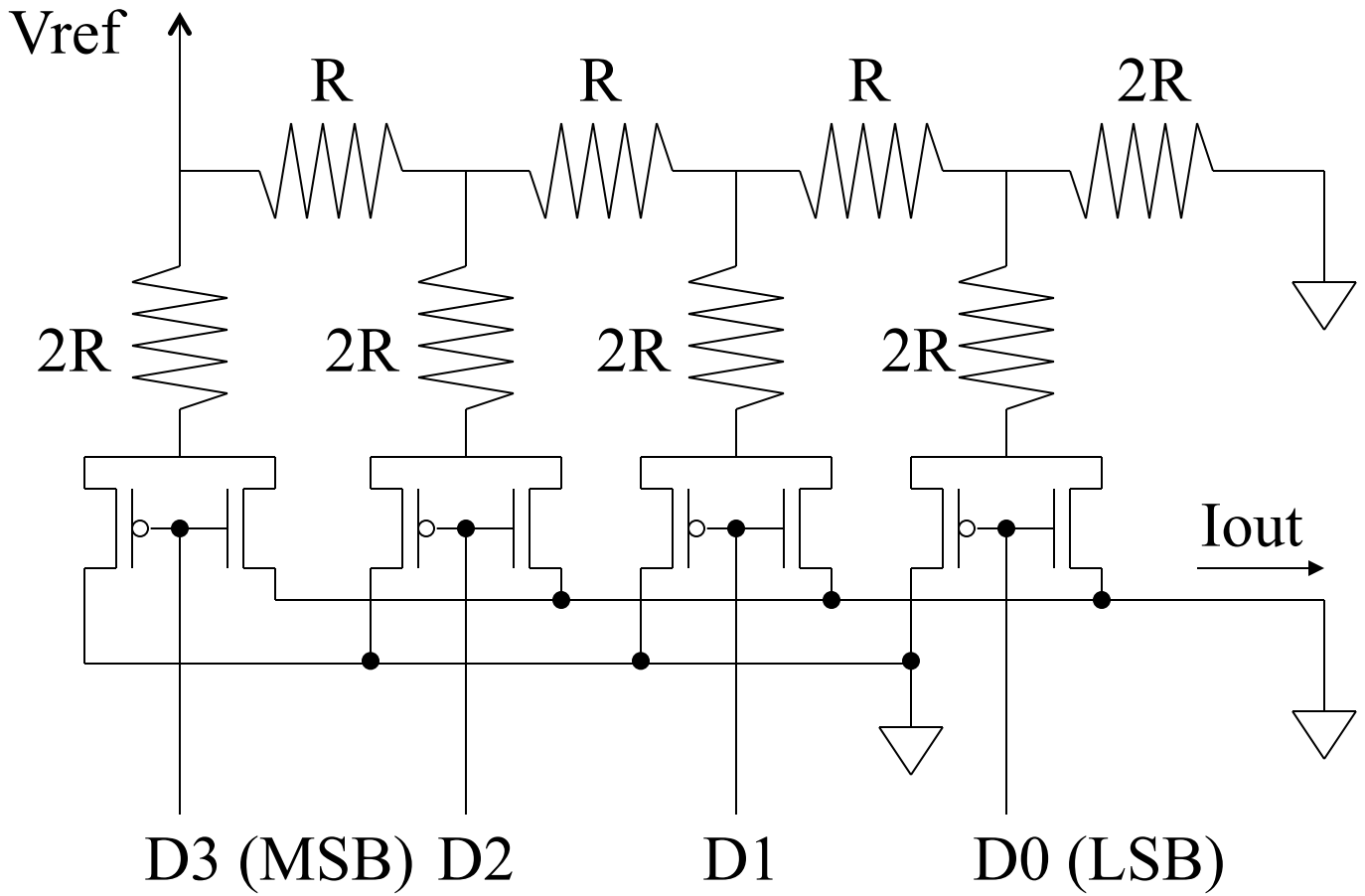
- Next Time…

DAC #1: Voltage Divider



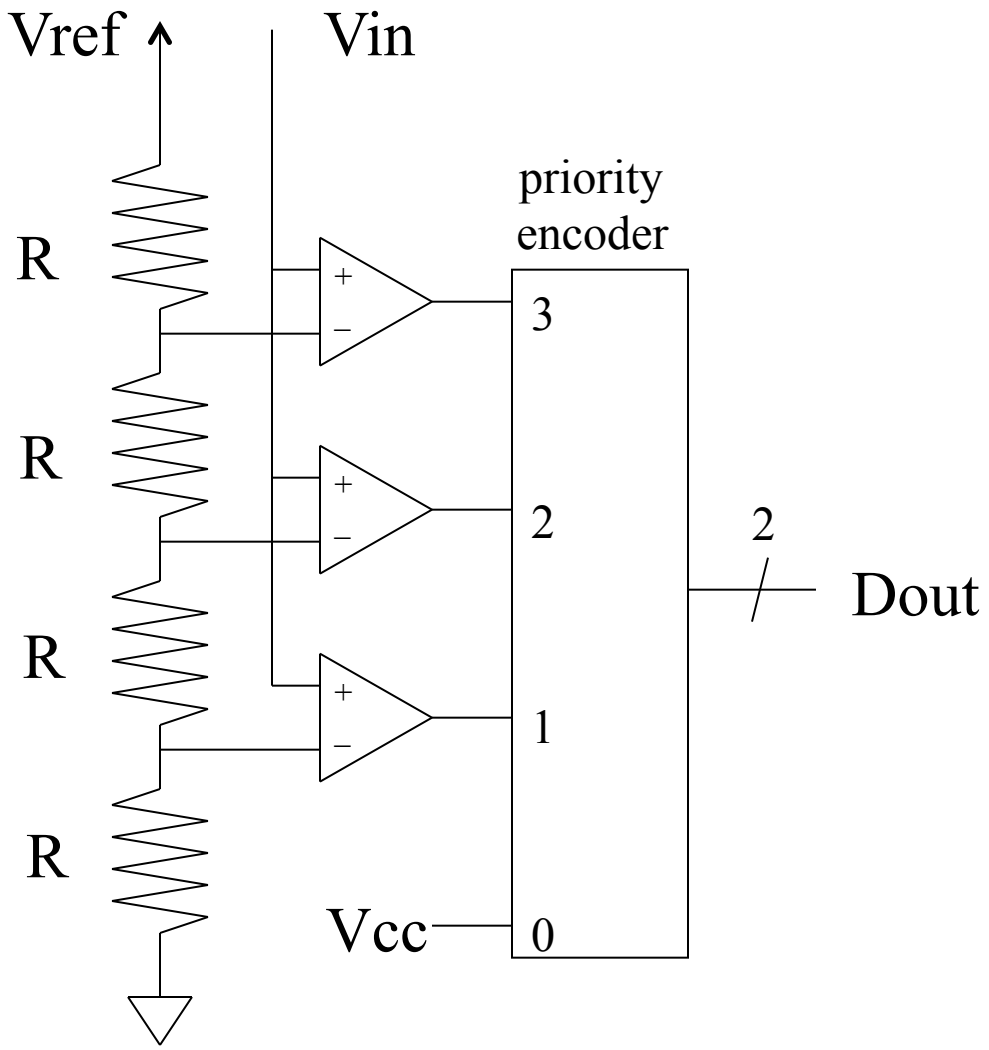
- Fast
- Size (transistors, switches)?
- Accuracy?
- Monotonicity?

DAC #2: R/2R Ladder

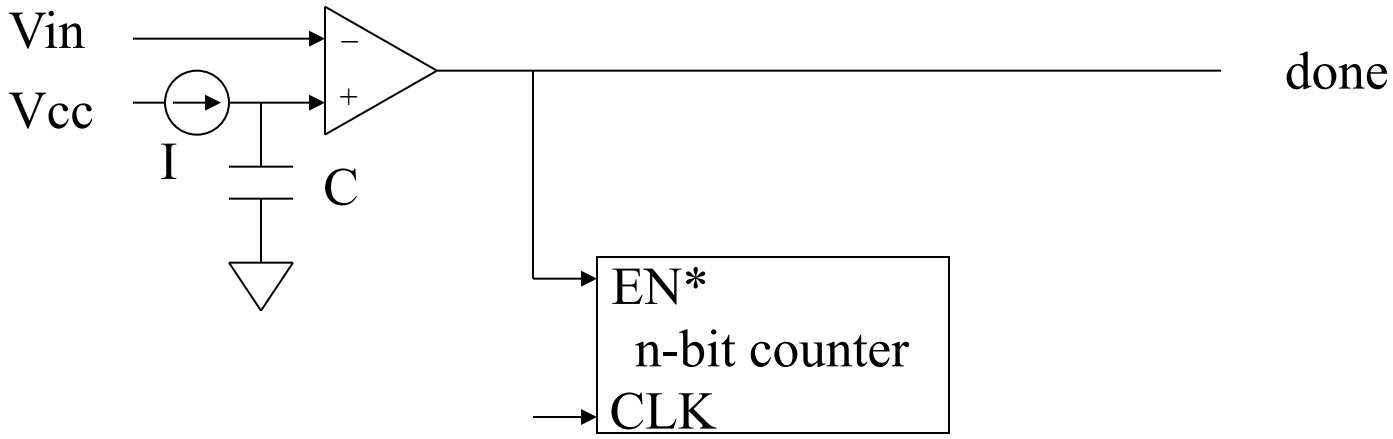


- Size?
- Accuracy?
- Monotonicity? (Consider 0111 -> 1000)

ADC #1: Flash

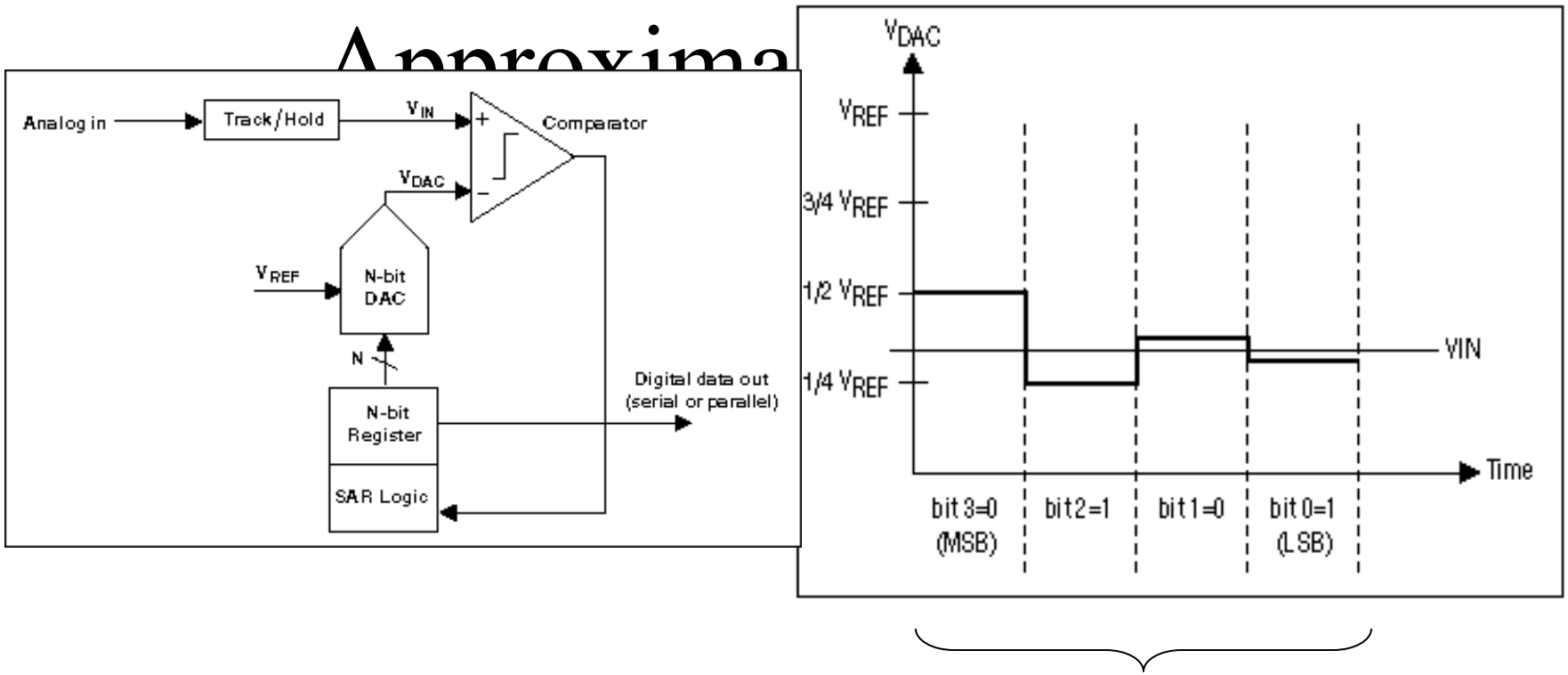


ADC #2: Single-Slope Integration



- Start: Reset counter, discharge C.
- Charge C at fixed current I until Vc > Vin . How should C, I, n, and CLK be related?
- Final counter value is Dout.
- Conversion may take several milliseconds.
- Good differential linearity.
- Absolute linearity depends on precision of C, I, and clock.

ADC #3: Successive Approxima

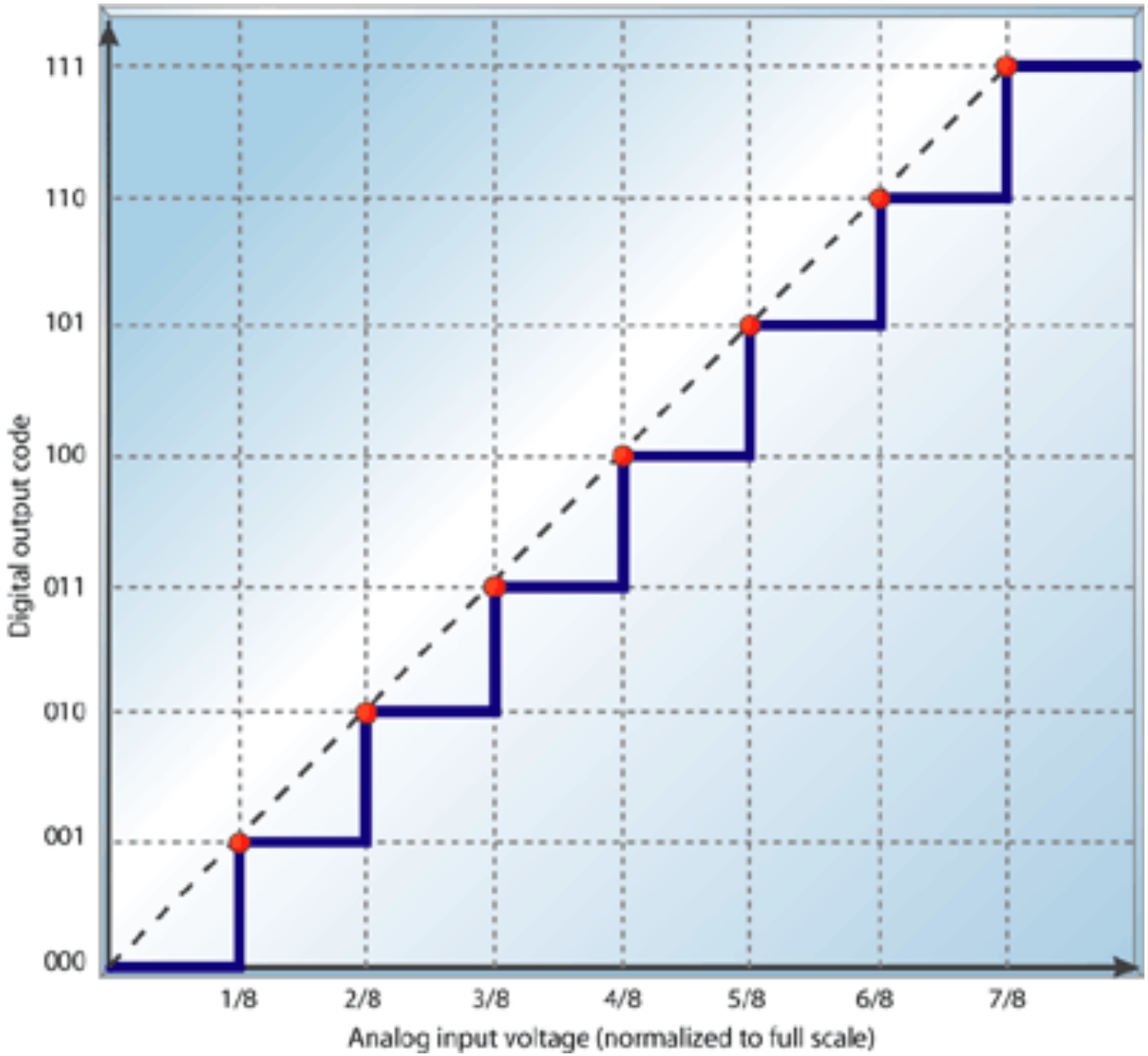


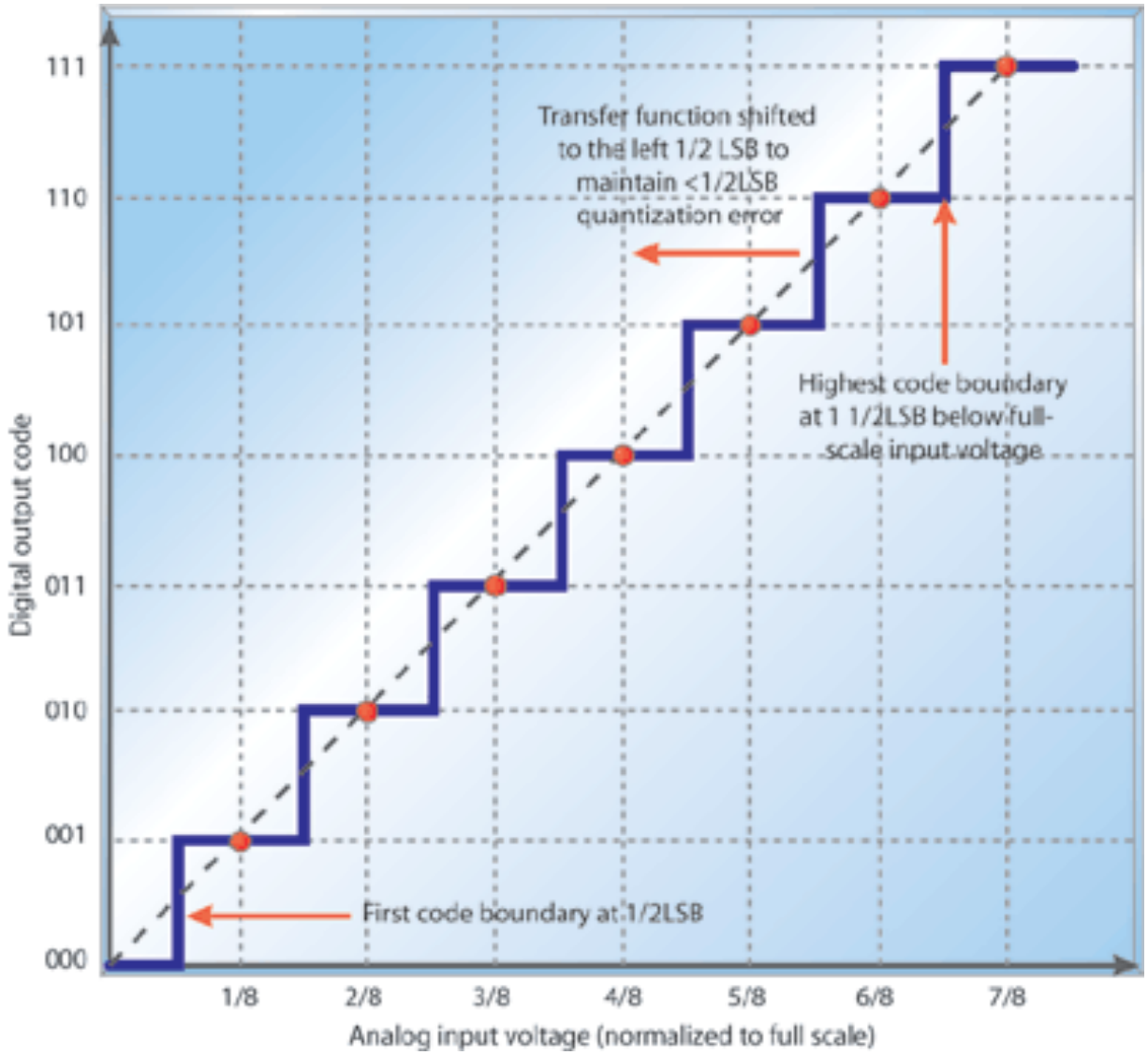
1 Sample → Multiple cycles

- Requires N-cycles per sample where N is # of bits
- Goes from MSB to LSB
- Not good for high-speed ADCs

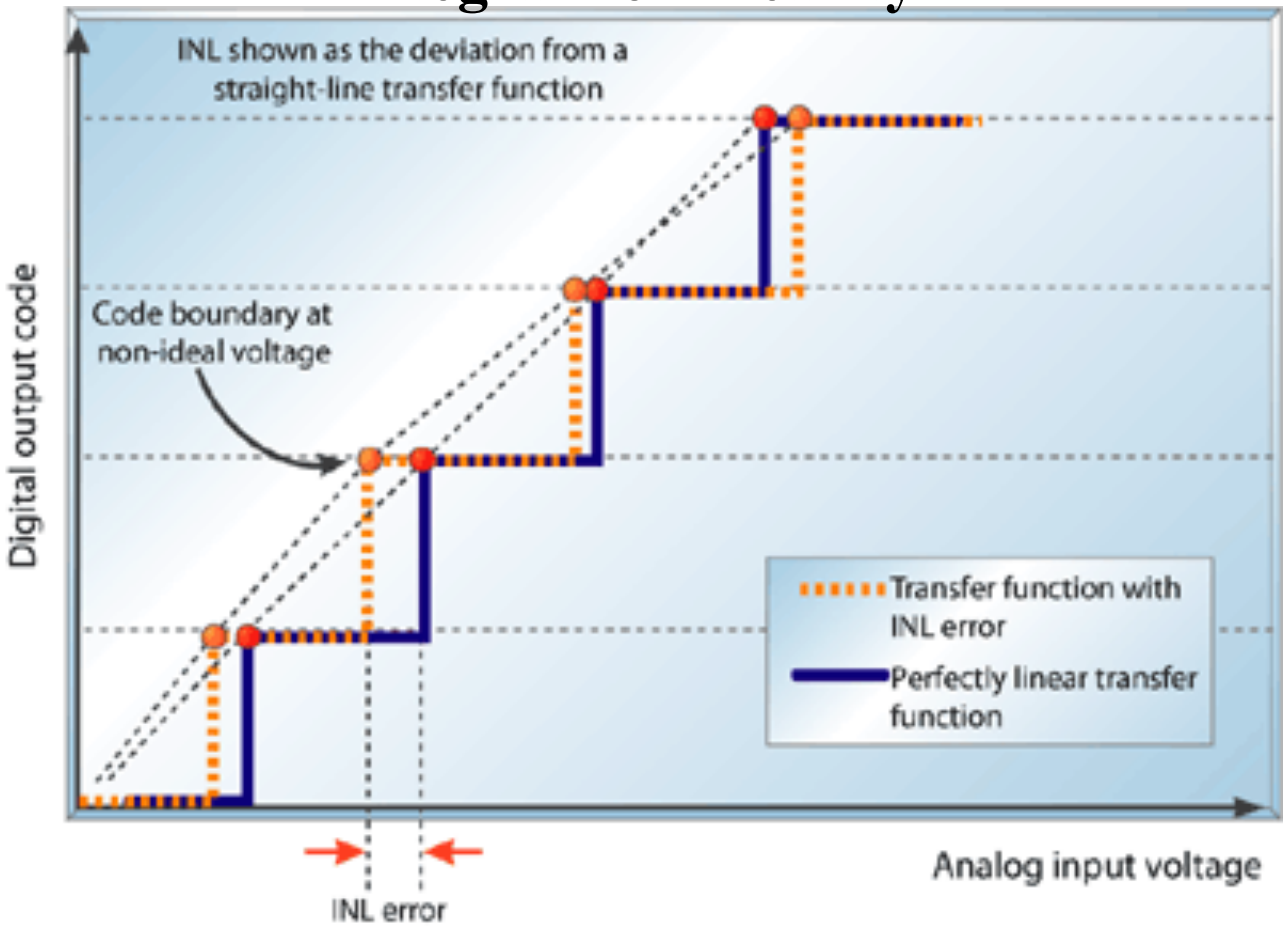
Errors and ADCs

- Figures and some text from:
  - Understanding analog to digital converter specifications. By Len Staller
  - http://www.embedded.com/showArticle.jhtml?articleID=60403334
- Key concept here is that the specification provides *worst case* values.


111
110
101
100
011
010
001
000
Digital output code
1/8
2/8
3/8
4/8
5/8
6/8
7/8
Analog input voltage (normalized to full scale)

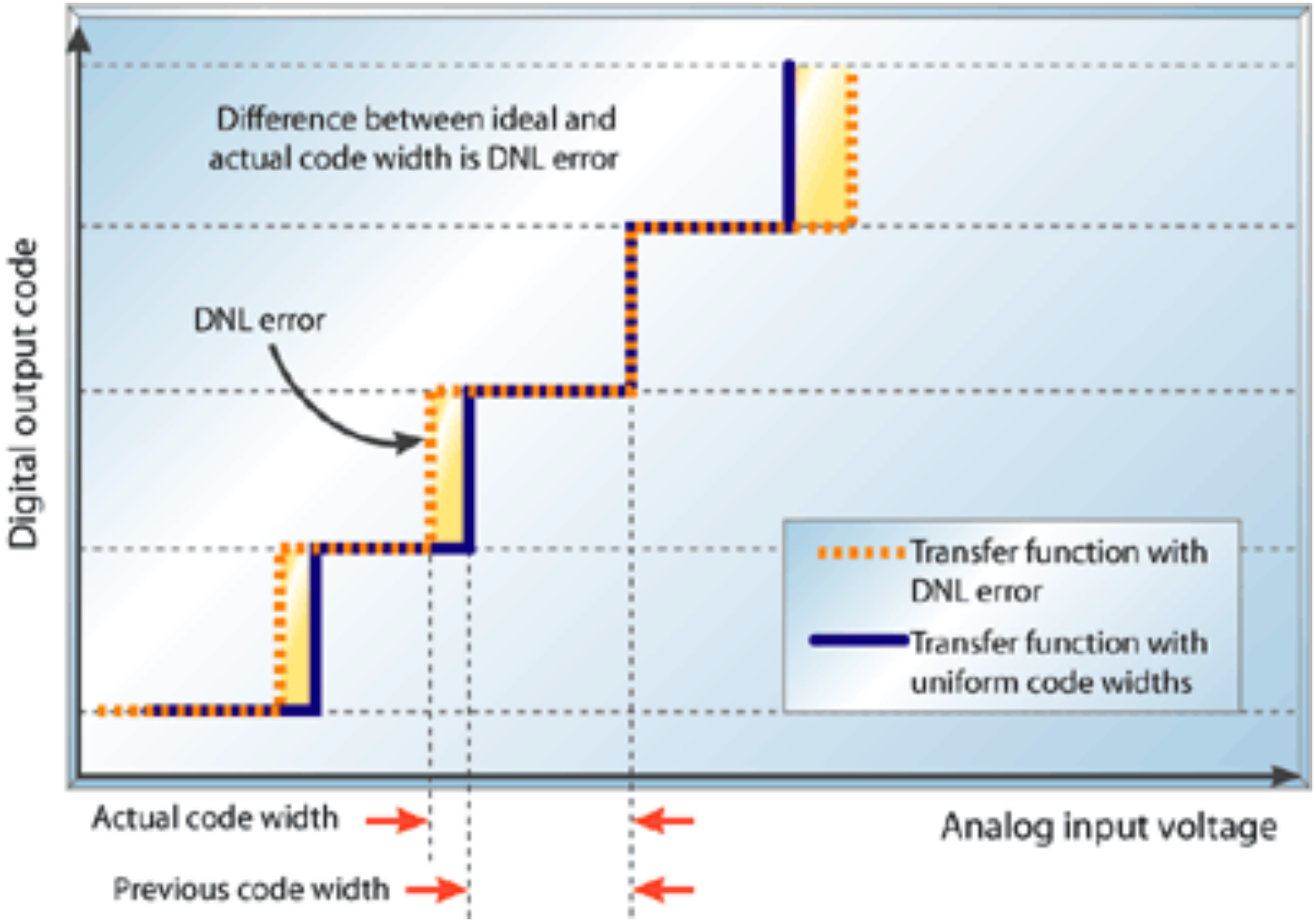

111
110
101
100
011
010
001
000
Digital output code
Transfer function shifted to the left 1/2 LSB to maintain <1/2LSB quantization error
Highest code boundary at 1 1/2LSB below full-scale input voltage
First code boundary at 1/2LSB
1/8
2/8
3/8
4/8
5/8
6/8
7/8
Analog input voltage (normalized to full scale)

Integral nonlinearity



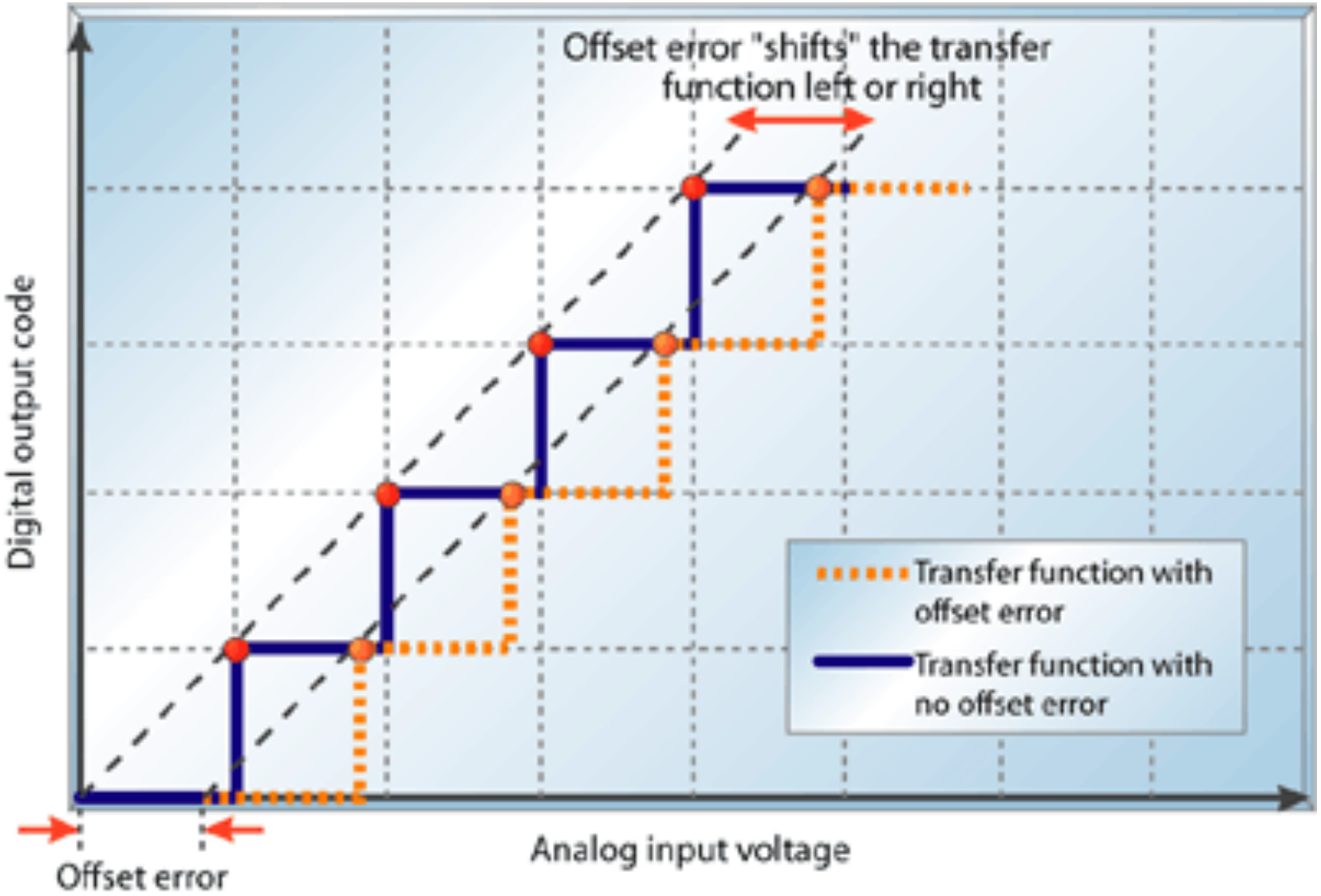
The *integral nonlinearity (INL) is the deviation of an ADC's transfer function from a* ***straight line***. This line is often a best-fit line among the points in the plot but can also be a line that connects the highest and lowest data points, or endpoints. INL is determined by measuring the voltage at which all code transitions occur and comparing them to the ideal. The difference between the ideal voltage levels at which code transitions occur and the actual voltage is the INL error, expressed in LSBs. INL error at any given point in an ADC's transfer function is the accumulation of all DNL errors of all previous (or lower) ADC codes, hence it's called *integral* nonlinearity.

Differential nonlinearity



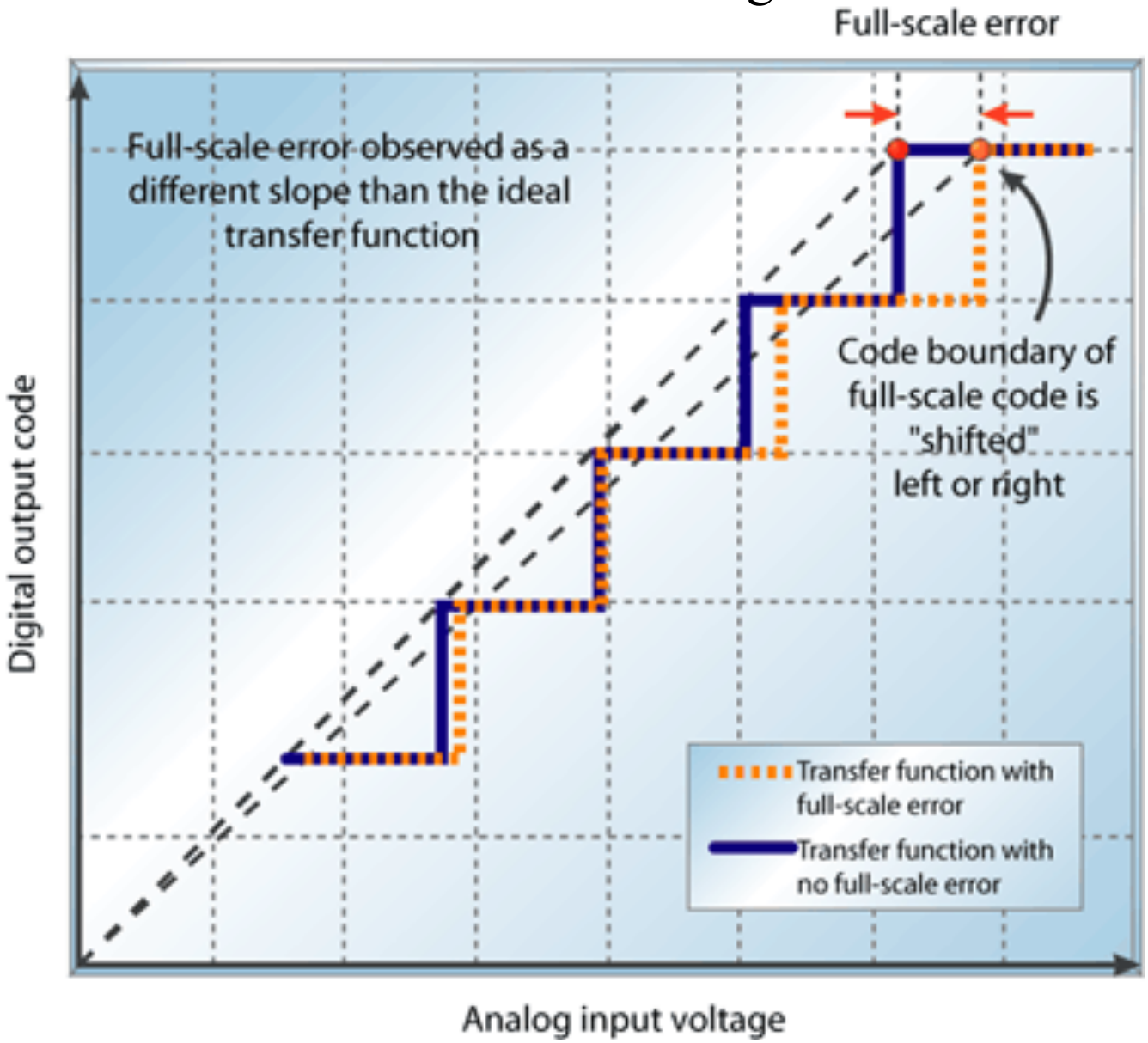
DNL is the worst cases variation of actual step size vs. ideal step size.

It's a promise it won't be worse than X.



Sometimes the intentional ½ LSB shift is included here!

Full-scale error is also sometimes called “gain error”



full-scale error is the difference between the ideal code transition to the highest output code and the actual transition to the output code when the offset error is zero.

Errors

- Once again: Errors in a specification are worst case.
  - So if you have an INL of ±.25 LSB, you “know” that the device will never have more than .25 LSB error from its ideal value.
  - That of course assumes you are opperating within the specification
    - Temperature, input voltage, input current available, etc.
- INL and DNL are the ones I expect you to work with
  - Should know what full-scale error is