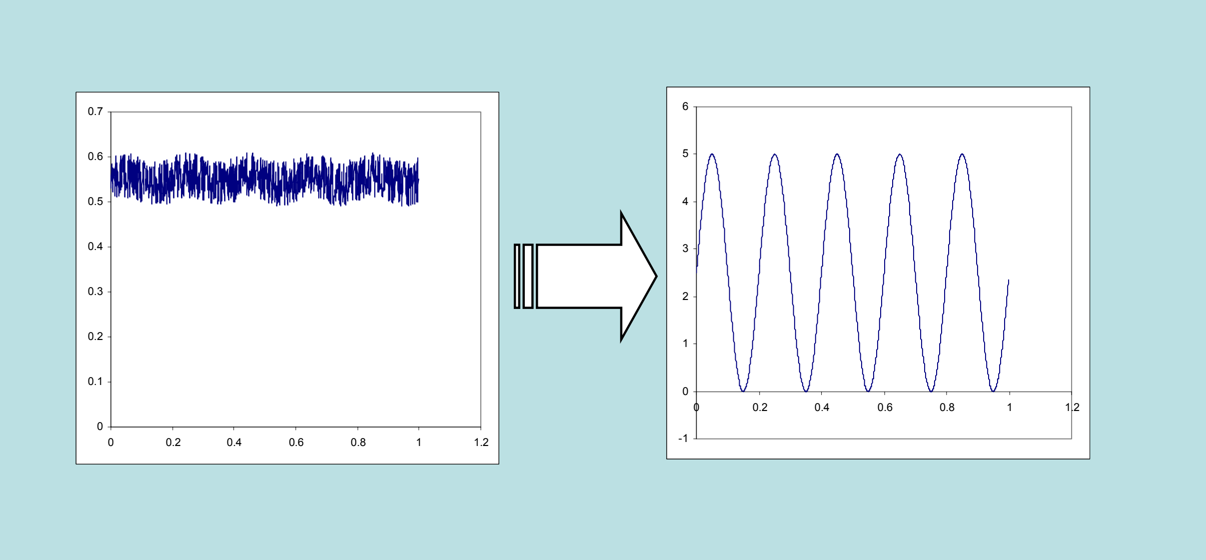
Using the Anadigm® FPAA to Interface With Sensors – Technical Considerations

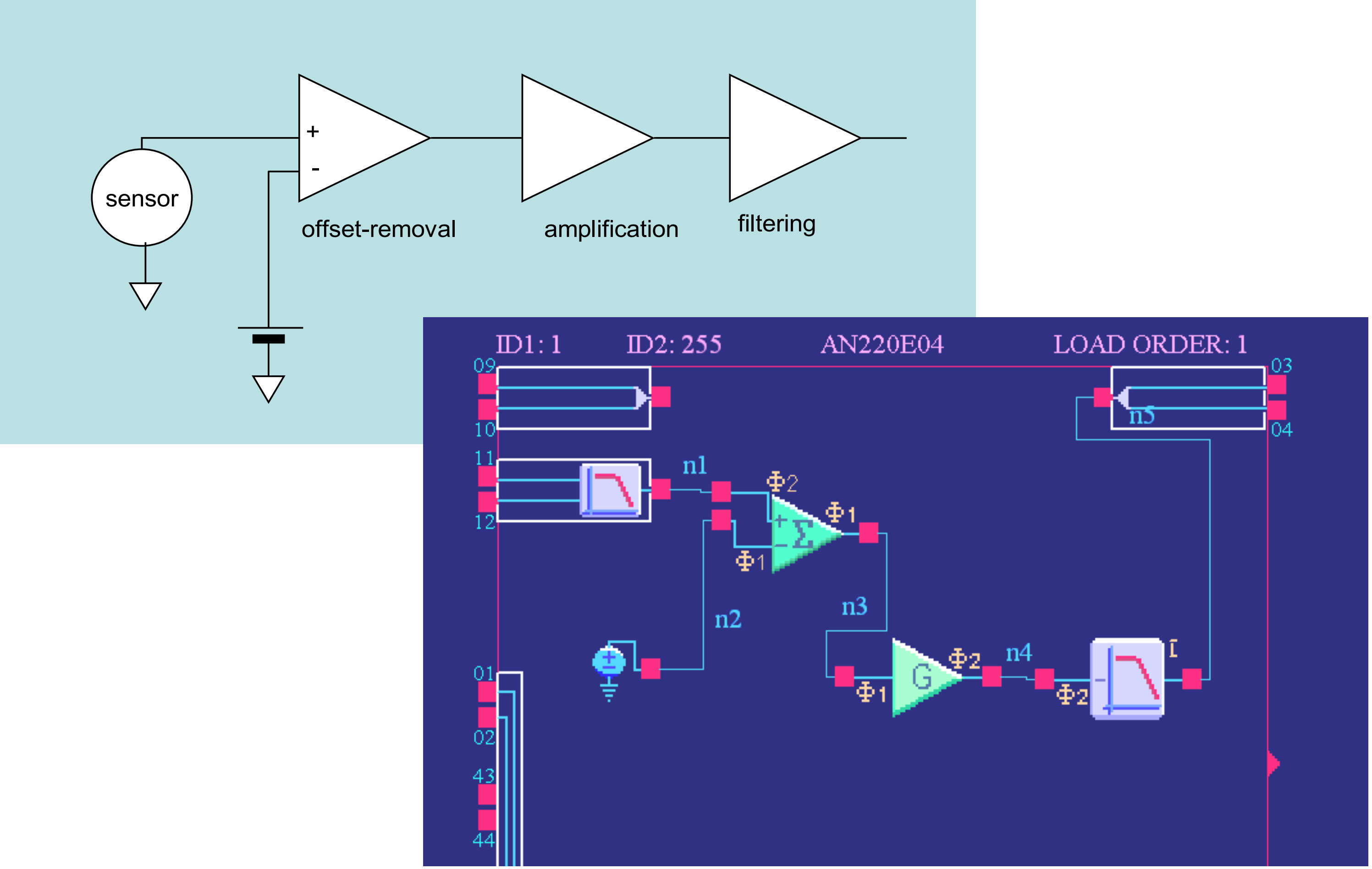
Sensor Signal Conditioning Needs

- **Common signal conditioning tasks:**
  - Amplification
  - Offset removal
  - Rectification
  - Filtering



Sensor Signal Conditioning Needs

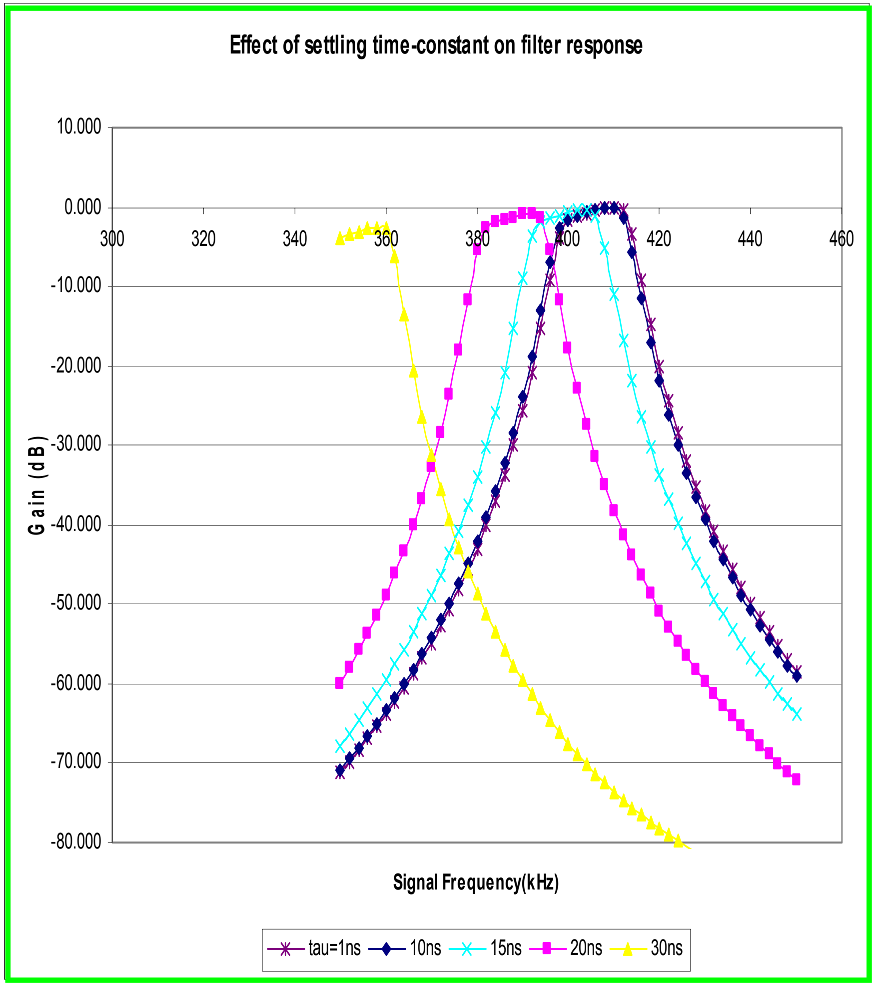
- Common conditioning tasks



Operating Limits (1):

High Clock Frequencies

- **Finite OpAmp bandwidth**
  - Capacitors must charge or discharge within each half clock-cycle
  - OpAmp speed affects settling-time
- **Non-zero switch resistance**
  - Similar effect on settling-time



Operating Limits (2):

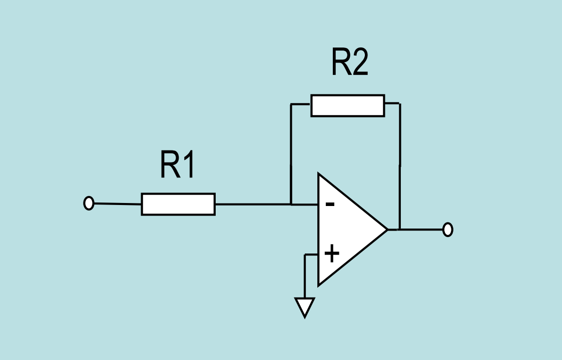
High Signal Frequencies

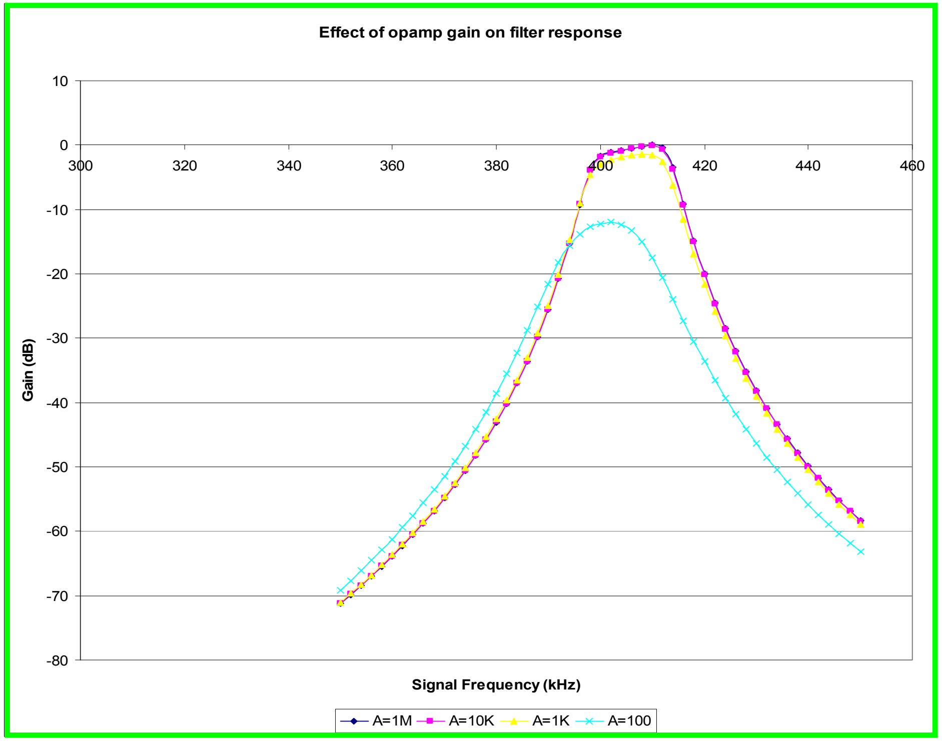
- **Finite OpAmp Gain**
  - Provided opamp has high *open-loop* gain and high input-resistance then:

$$\frac{1}{G} = -\frac{R1}{R2}$$

  - If open-loop gain is not infinite then:

$$\frac{1}{G} = -\frac{R1}{R2}\left(1+\frac{1}{A}\right)-\frac{1}{A}$$





Operating Limits (3):

Low Clock Frequencies

- **"Droop"**
  - The charge that should be stored on a node is:
    $Q = C.V$
  - This charge will be changed (+ or -) by leakage:
    $\Delta Q_L = I_L.T$
  - So, the error will be
    $Error = I_L.T / C.V$
  - For a specified max error:
    $$T < \frac{(C.V)_{min}.Error}{(I_L)_{max}}$$

So, using some approximate values:

- $C \cong 10^{-11}$ (farad)
- $V \cong 10^{-3}$ to 1 (volt)
- So, $CV \cong 10^{-11}$ to 10^{-14} (coulomb)
- $I_L \cong 10^{-15}$ to 10^{-18} (amperes)

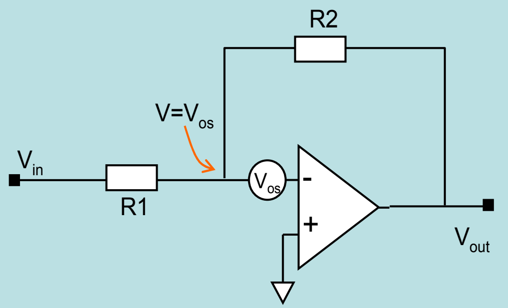
For a leakage errors of 0.1% or less:

$$T < (C.V)_{min}.Error / (I_L)_{max} \cong 10^{-14}.10^{-3} / 10^{-15} \cong 10^{-2} \text{ (secs)}$$

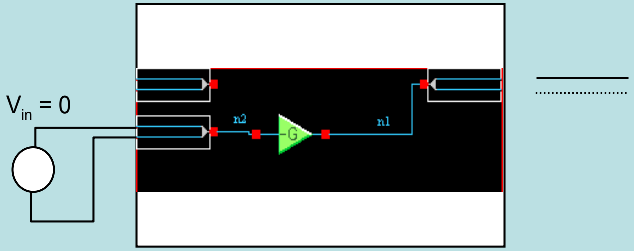
We can obtain leakage errors of 0.1% or less for millivolt signals, for sampling frequencies in the low kHz range.

Offset Voltages

- **Offset errors**
  - Slight mismatches inside the opamp require balancing with a small input voltage
  - Output voltage will have a DC error

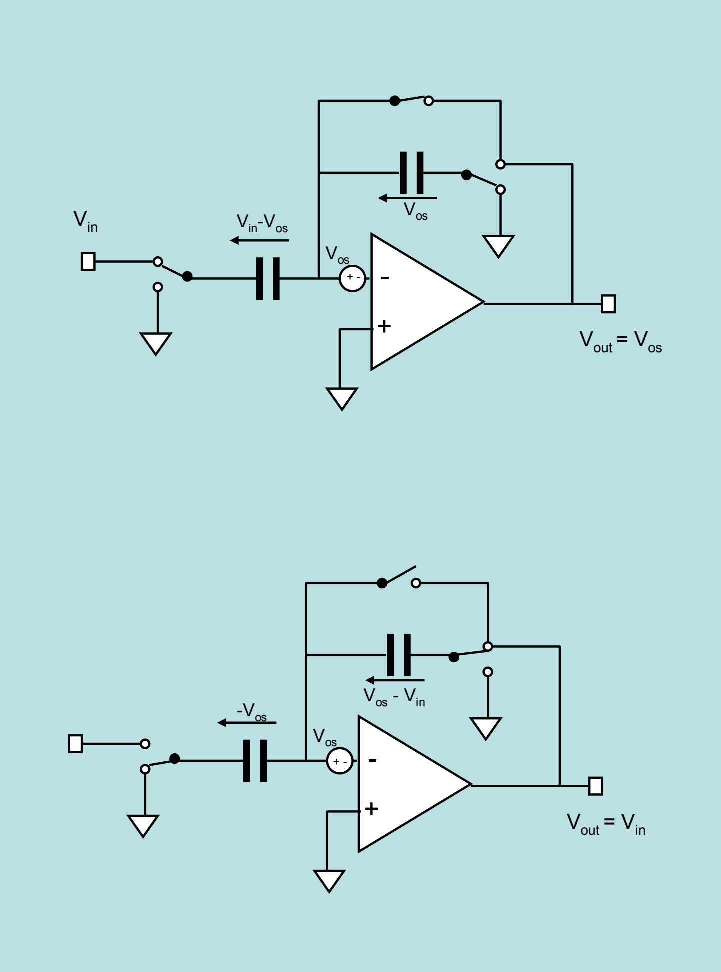


$$V_{out} = -V_{in} \cdot \frac{R2}{R1} + V_{os}\left(\frac{R2}{R1} + 1\right)$$



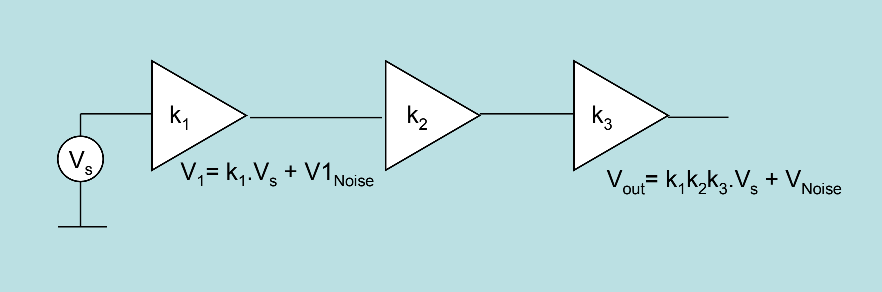
Offset Compensation

- **“Half-cycle” building blocks allow cancellation of offset voltage**
  - GainHalf
  - GainHold
  - RectifierHalf
  - RectifierHold
  - SumDiff
- **“Offset-compensated” in one clock phase only (return-to-zero or hold-with-offset in other phase)**
- **Gain-Bandwidth Product (GBP) and slew-rate are not infinite!**
  - Selection of high-gain will limit max clock frequency (and vice versa) of RTZ CAMs



Achieving High-gain

- **Cascade stages – high-gain at front-end for low-noise**

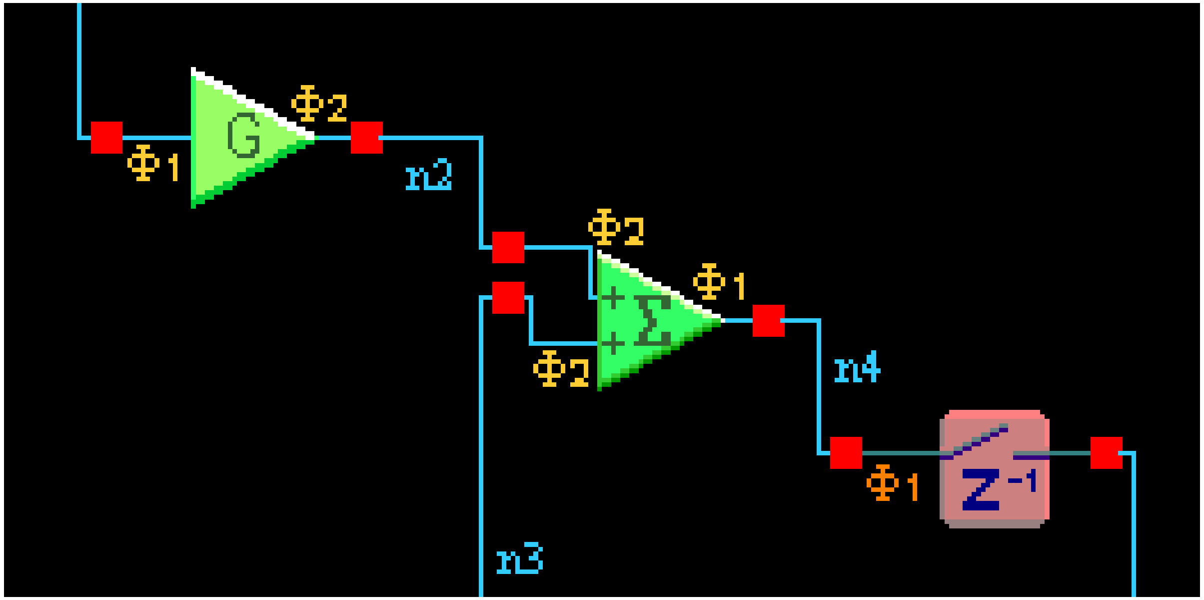


$$V_{noise} = K.\left(\frac{V1_{noise}}{k_1} + \frac{V2_{noise}}{k_1k_2} + \frac{V3_{noise}}{k_1k_2k_3}\right)$$

- **Use offset-compensated CAMs when implementing large gains**
  - Remember GBP limitations – select appropriate clock frequency (but watch anti-aliasing requirements)

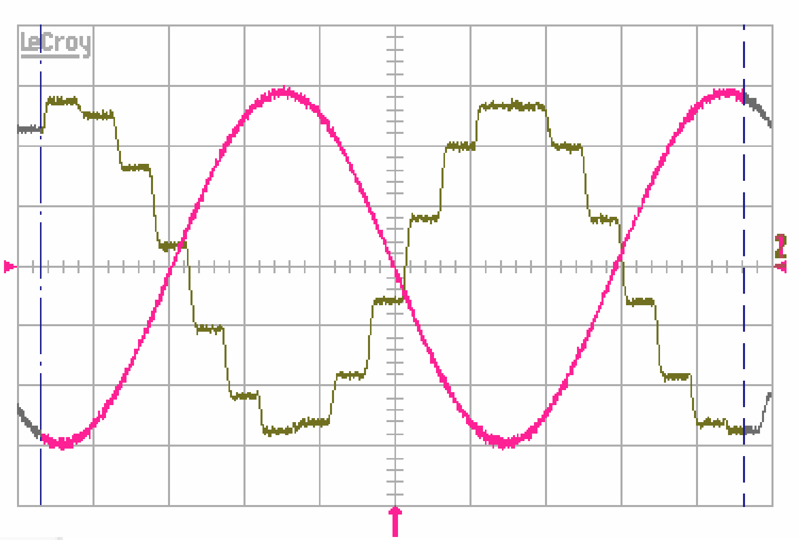
Phase Selection

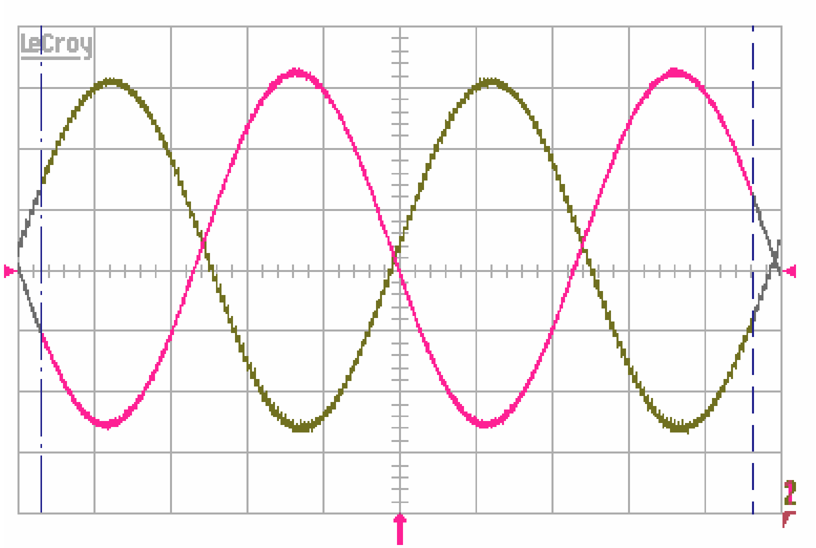
- Some CAMs have outputs which are only valid (or offset-compensated) in one phase
- Make sure input-sampling and output-valid phases match!



Output Waveforms (1)

- **Staircase outputs:**

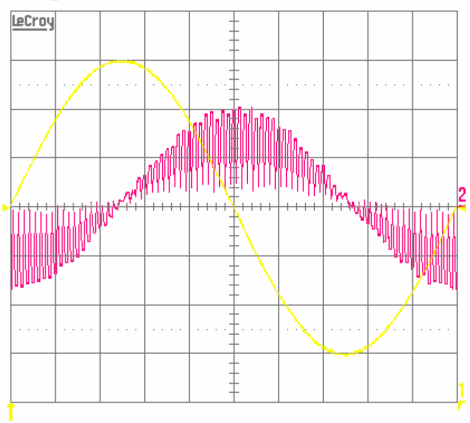


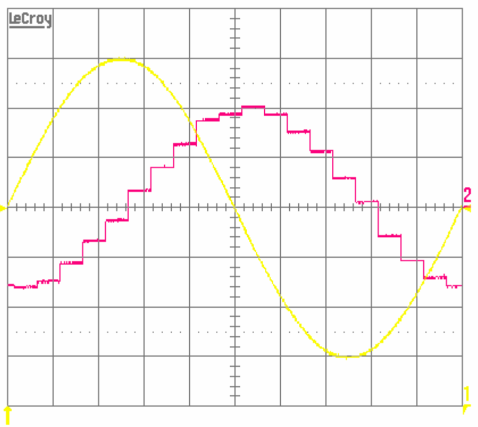


- ❑ Use high f_{clock}/f_{signal} ratio to reduce step size and clock noise
- ❑ Keep $f_{corner}/f_{signal} > 30$ if using output cell filter
- ❑ If output is being sampled (eg by ADC), steps may be desirable !

Output Waveforms (2)

- **“Final-value” outputs**
  - Differentiator (and transimpedance amplifier) outputs only reach their final value at the end of a clock phase
  - Final-value CAM outputs need capturing by CAMs which sample in one-phase and produce a valid output in the next

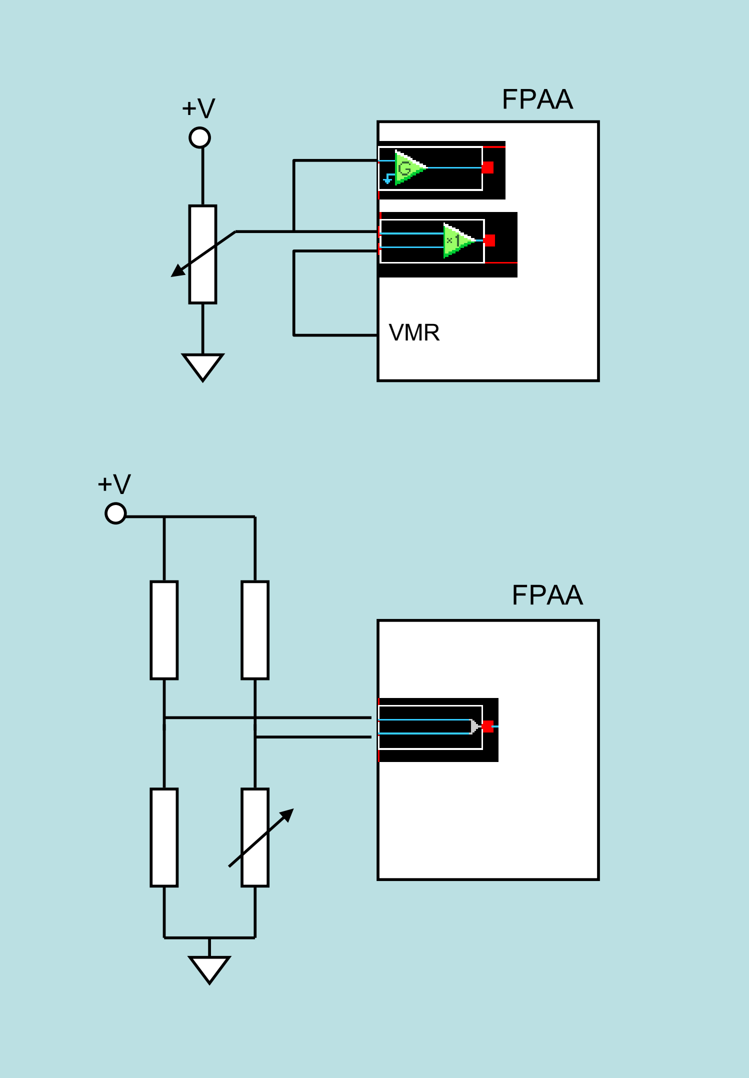




- **Hold**
- **Non-inverting GainHalf**
- **Non-inverting SumDiff**
- **Non-inverting SumFilter**

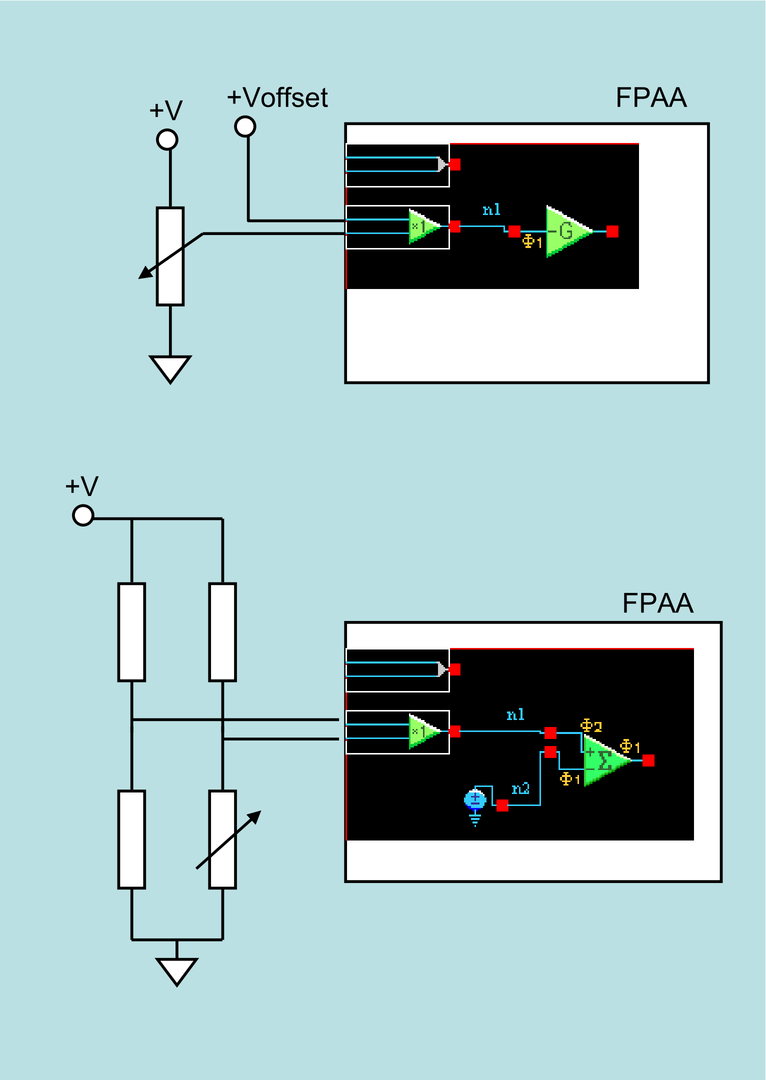
Input Levels

- **Internal signal ground is 2 volts (VMR)**
  - ❍ Center inputs on 2V if possible (use Wheatstone bridge?)
  - ❍ Input range 0 – 4V
  - ❍ Convert external single-ended signals to internal differential signals for max dynamic range



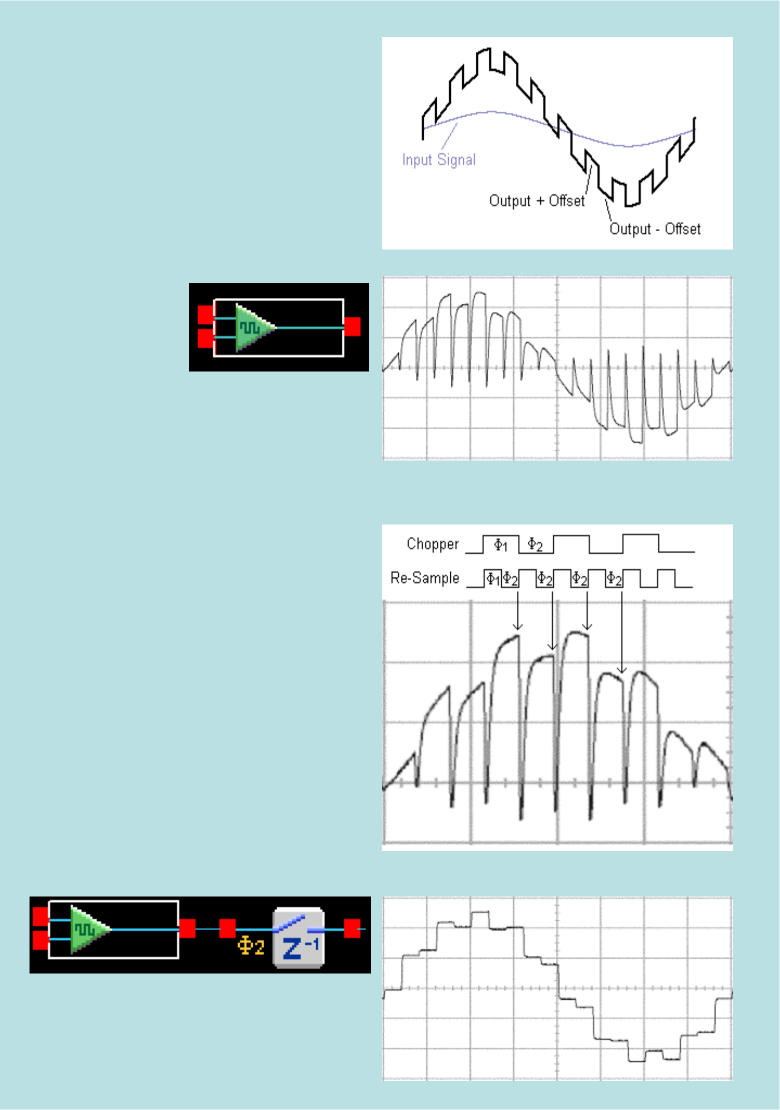
Offset Removal

- **Single-ended signals – apply offset to 2nd input pin**
- **Differential signals – apply offset internally**
- **Using differential signals gives better CM noise immunity**



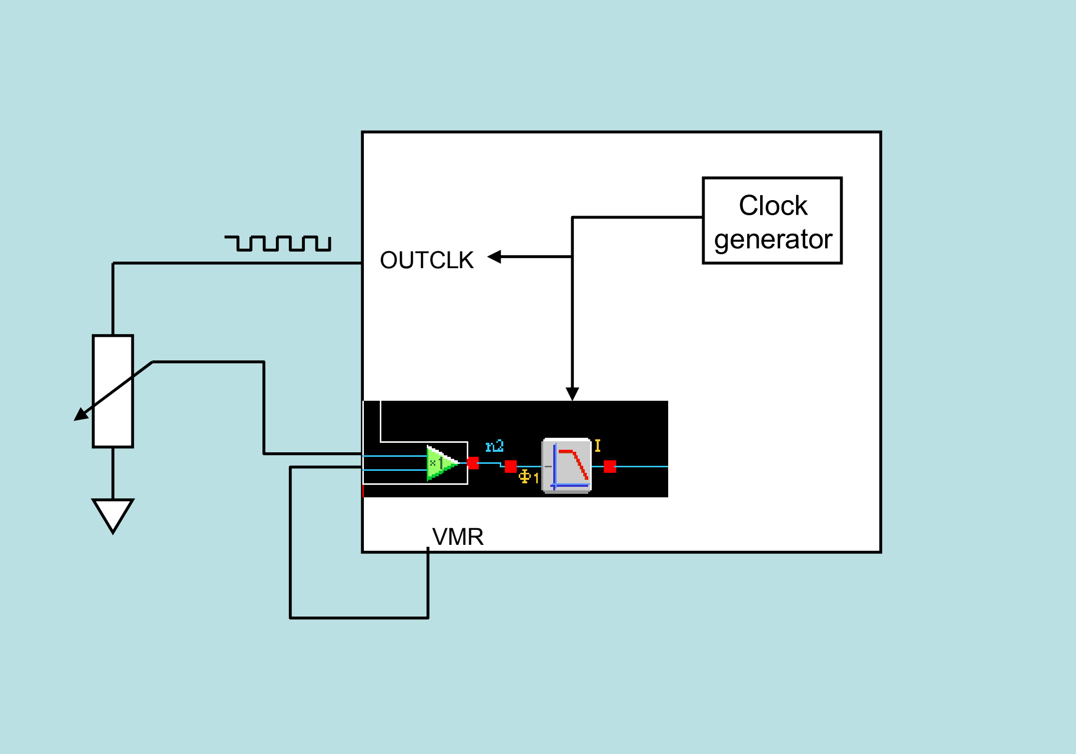
Using the Chopper Input

- **"Chopping" modulates the input signal with a square wave**
  - Low freq noise and DC offsets are pushed out to the clock freq (for easy filtering)
- **Slewing and settling limitations mean the chopper output is not "ideal"**
  - To recover the "ideal" output, sample in phase2 *at twice the chopper frequency*
- **To boost the gain, use offset-compensated CAMs, *sampling in phase2 at twice the chopper frequency***



Noisy Environments

- Use differential signals where noise can be made common-mode
- Use modulated sensor stimulus and synchronous demodulation (lock-in detection)



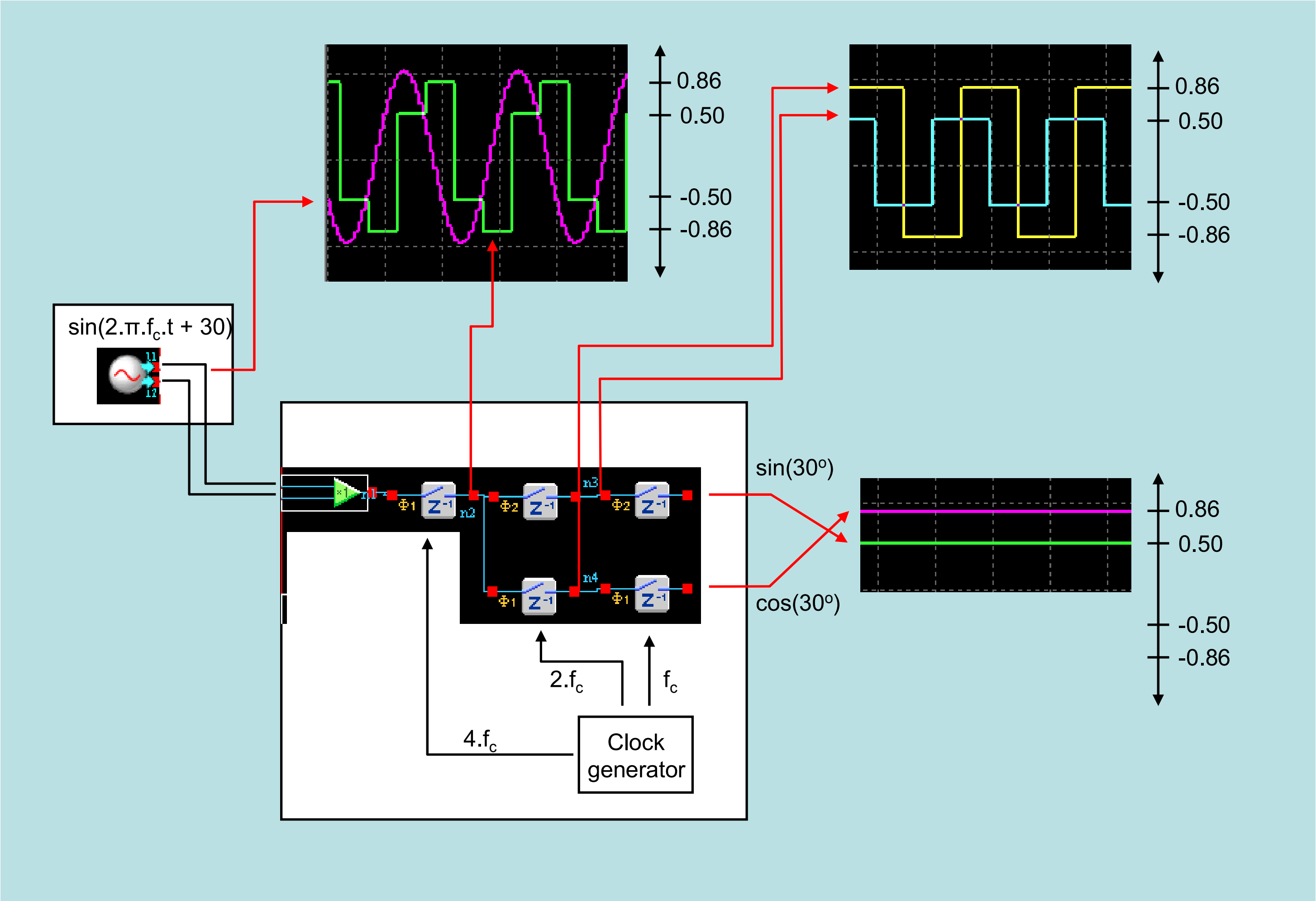
Clock generator

OUTCLK

VMR

Phase Detection

- **Synchronous demodulation**



Anti-aliasing and Clock Selection

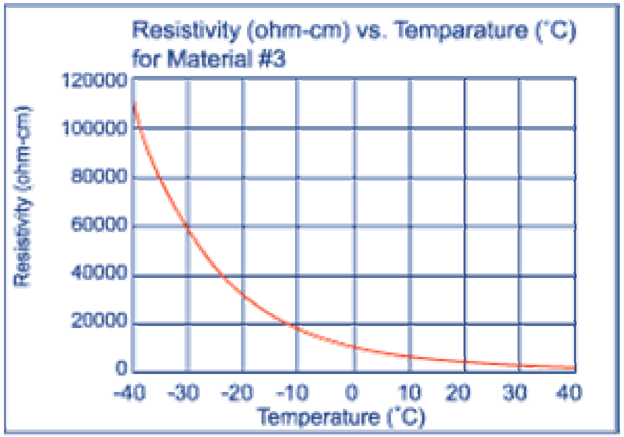
- **Input signal should be band-limited to << $f_{clock}/2$**
  - prevents hf noise adding to lf noise
  - Use a clock *at least* 5–10 times faster than the maximum signal

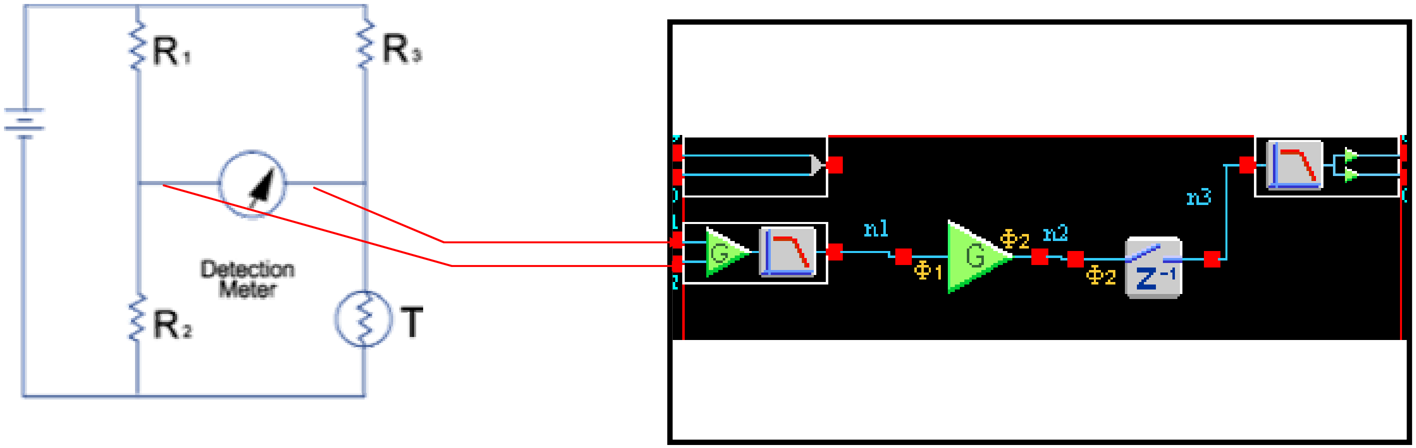
- **Use continuous-time not SC filtering**
  - attenuation at $f_{clock}/2$ should attenuate any hf signals by the required SNR
  - Keep $f_{corner}/f_{signal} > 30$ if using input cell filter

- **Lock-in detection averages out noise & signals except at harmonics of the modulation frequency**
  - Anti-aliasing not necessary (but will improve SNR if used)

Temperature-sensing (1)

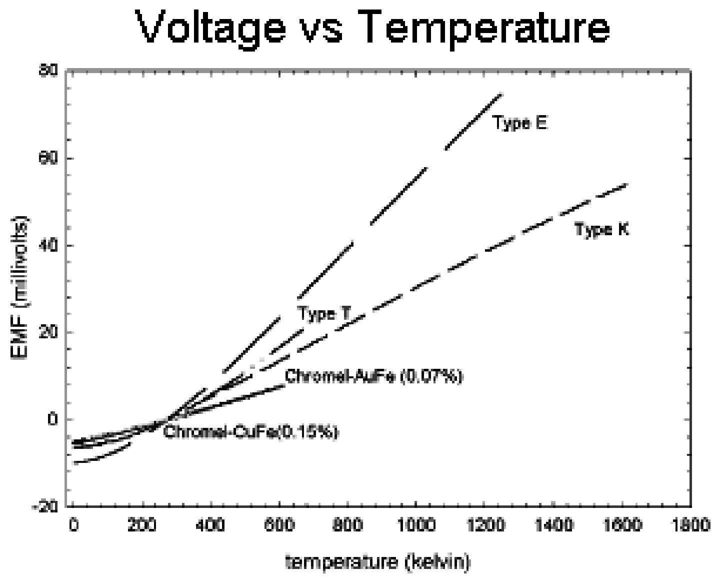
- Thermistors
    - Resistances range from 100 to 1Mohm
    - Typical current < 100uA to avoid self-heating

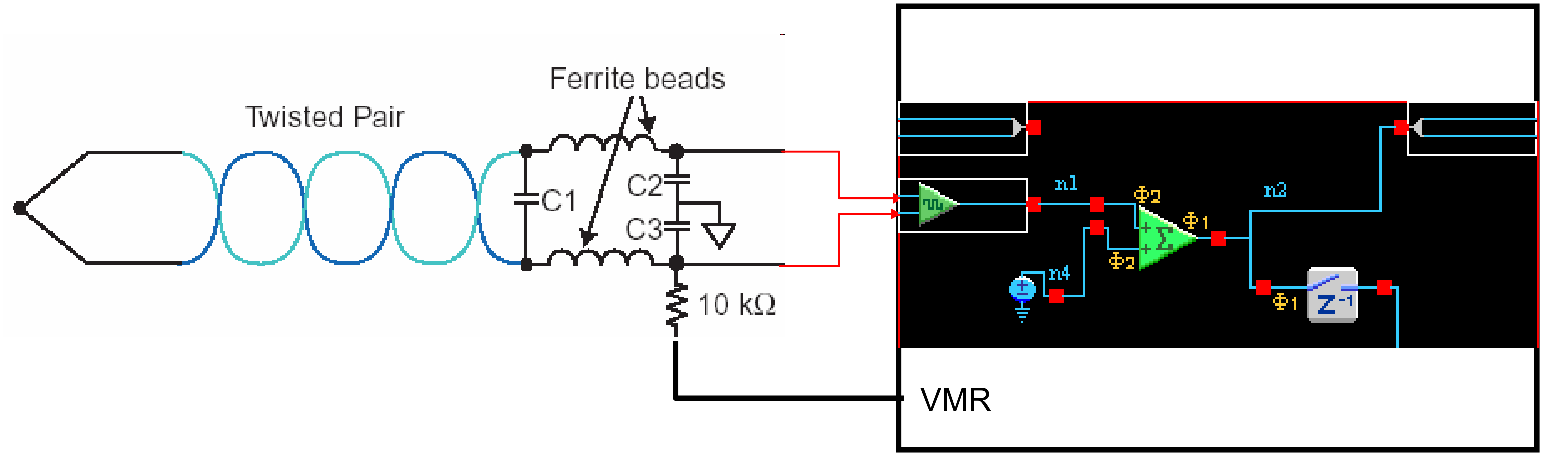




Temperature-sensing (2)

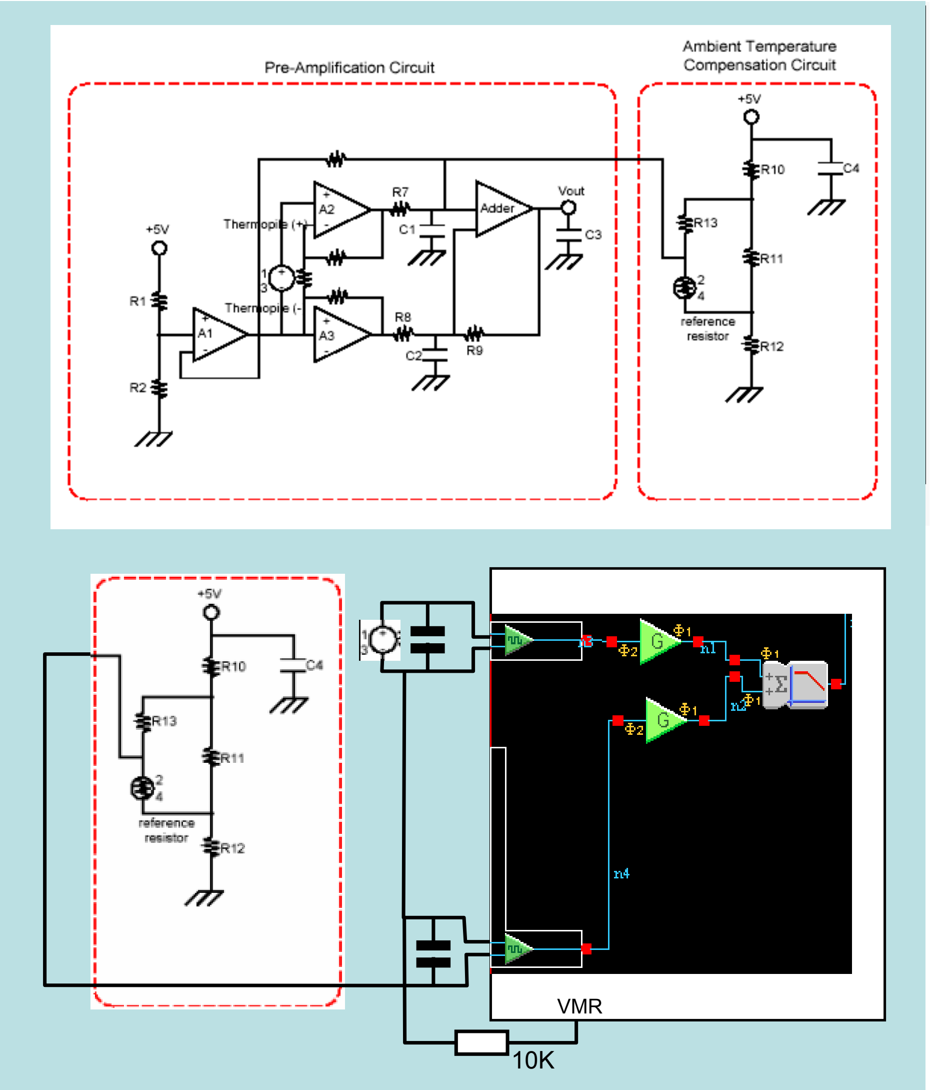
- **Thermocouples**
  - Small voltage/degree C
  - Prone to high levels of CM noise

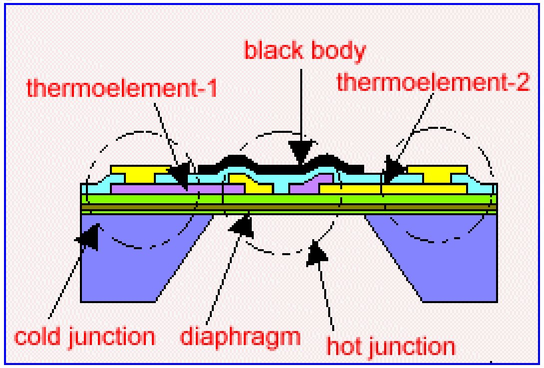




Temperature-sensing (3)

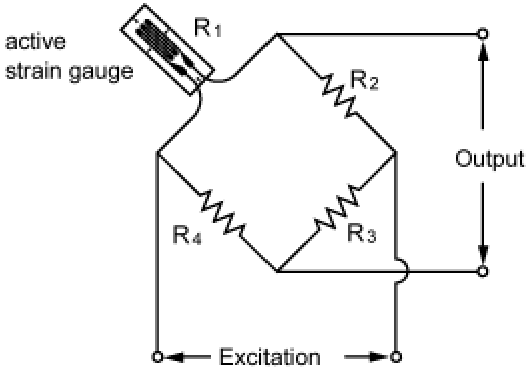
- **Thermopiles**
  - Serially-interconnected thermocouples with “hot” and “cold” junctions
  - “Hot” junction sensitive to received IR radiation
  - Close proximity of junctions gives low sensitivity to ambient temperature

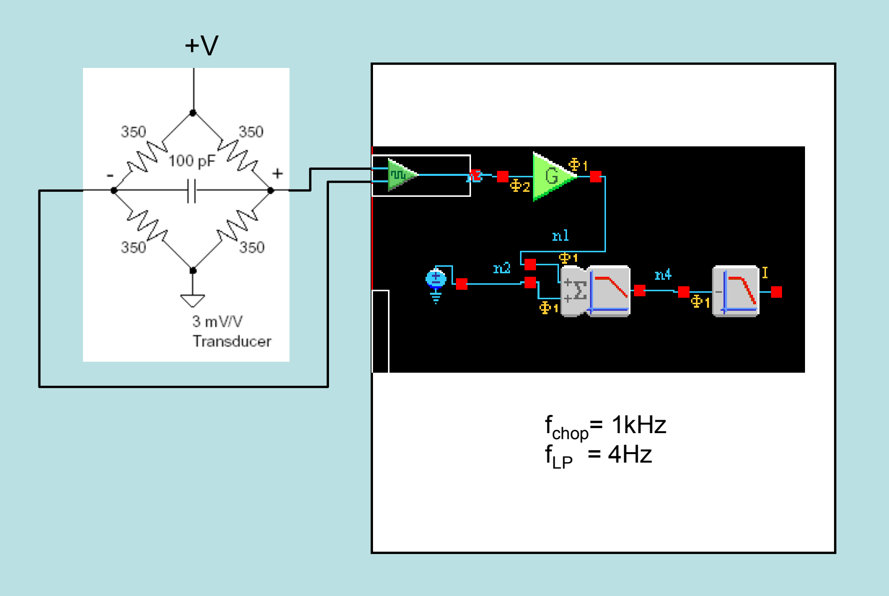




Position-sensing (1)

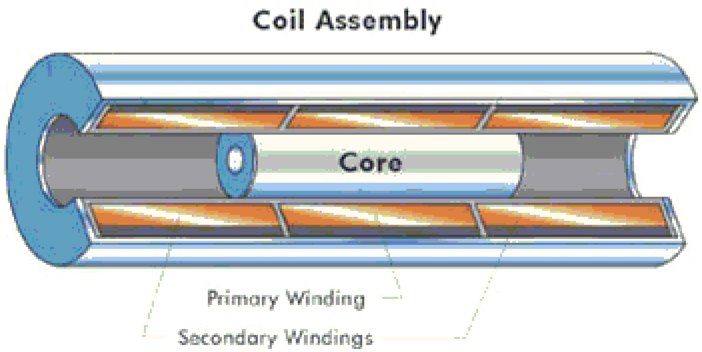
- **Strain gauges**
  - Typical output < 10 mV/V
  - “Half bridge” has strain gauges in two arms:
    - doubles the output and compensates for thermal effects
  - “Full bridge” has strain gauges in four arms:
    - re-doubles output and compensates for thermal effects

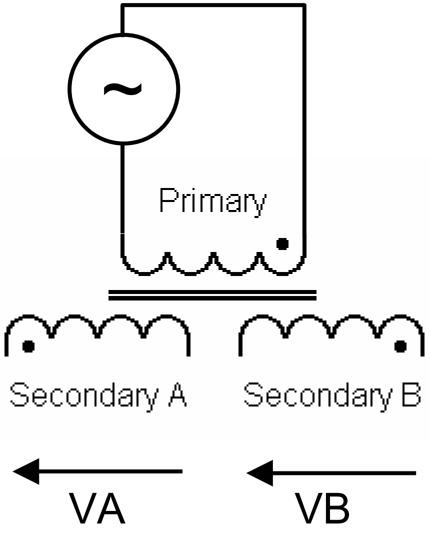




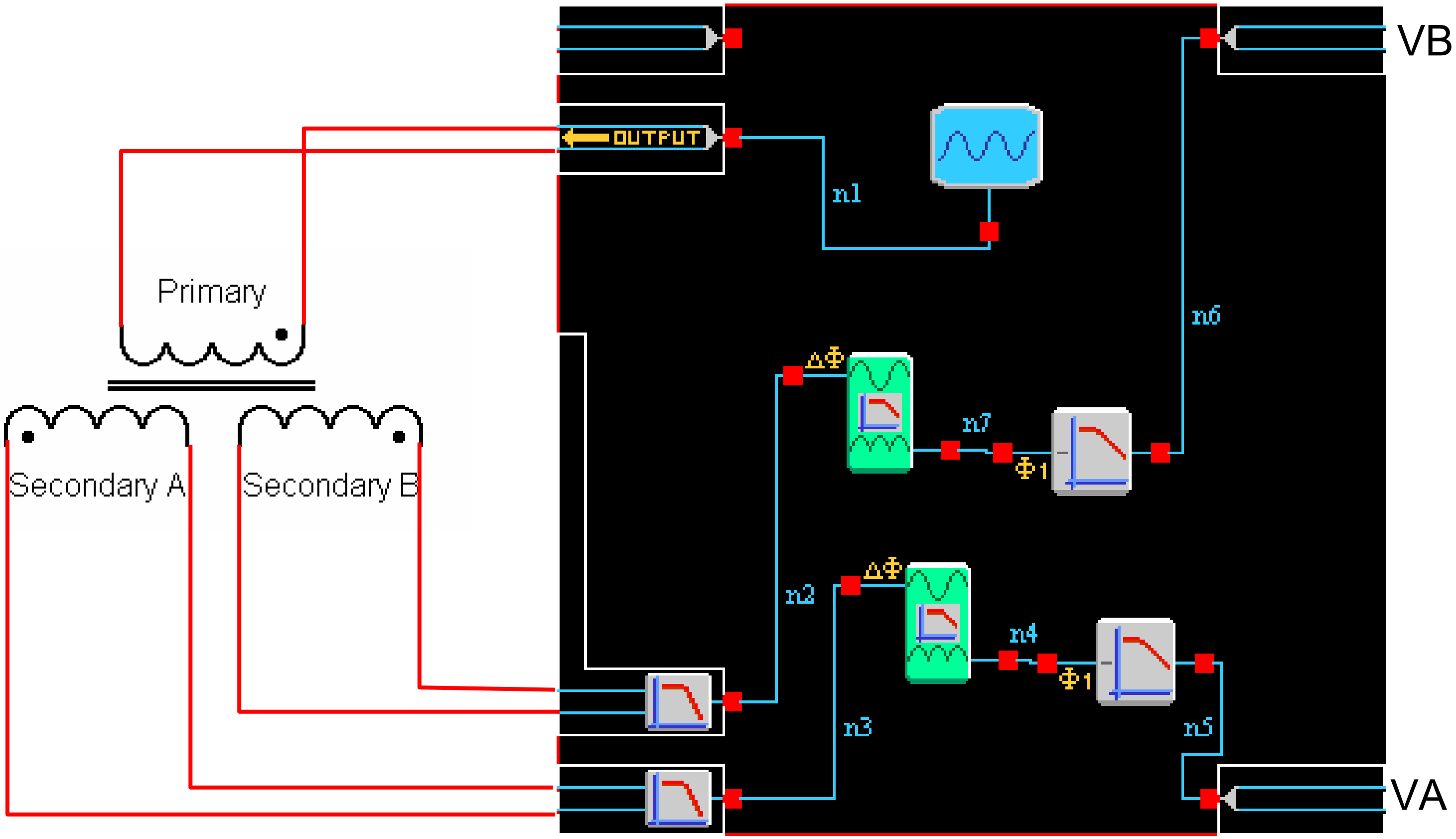
Position-sensing (2)

- **LVDT**
- **Sensitivity ranges:**
  - 0.05 mV/V/0.001” for long stroke LVDTs
  - 10 mV/V/0.001” for short stroke LVDTs
- **Various signal processing techniques (eg):**
  - Rectify and filter AC signals, then:
    - Calculate (VA + VB)/$V_{primary}$ (insensitive to the amplitude of the driving signal, and gives some noise rejection).
    - Calculate the **ratio** of the difference and the sum of the secondary voltages i.e. (VA – VB) / (VA+VB)
  - Measure the phase of the combined VA + VB secondary voltage i.e. -180 degrees at one extreme, zero degrees at the null position, and +180 degrees at the other extreme.





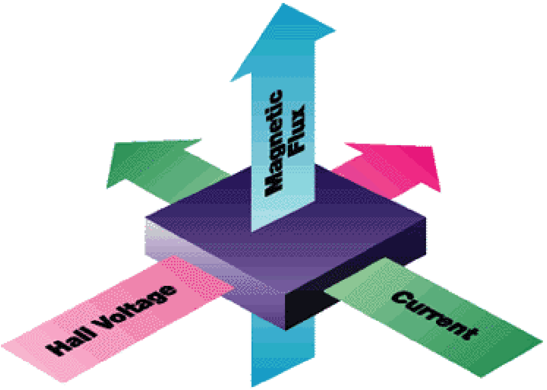
LVDT-sensing

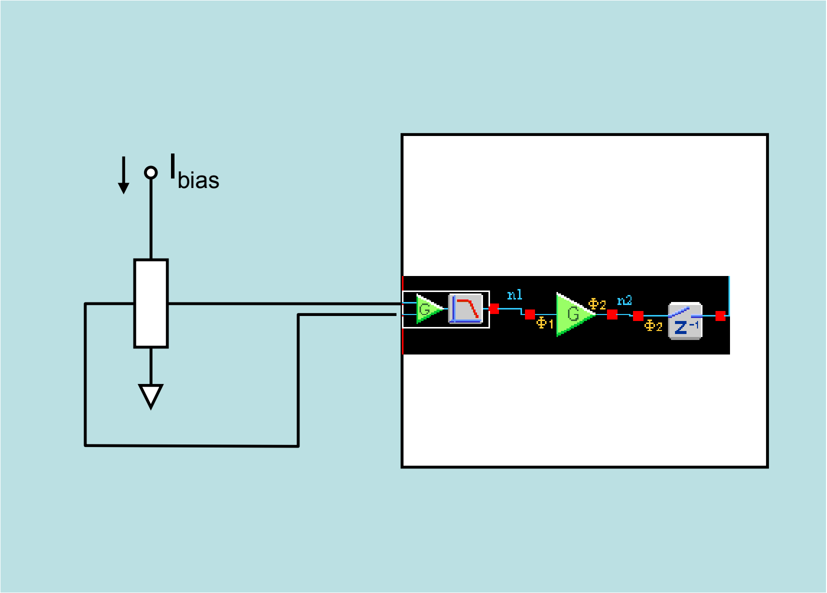


anadigm™

Magnetic-sensing

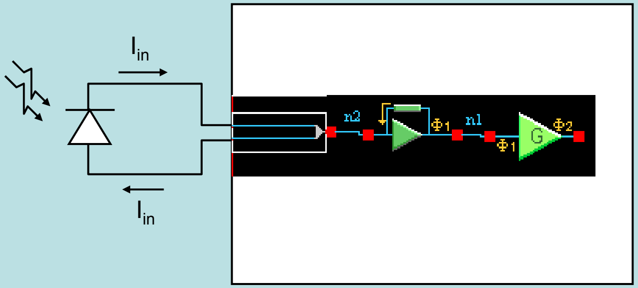
- **Hall effect devices**
  - Usually driven with a **constant** current
  - Differential output voltage, superimposed on a common-mode voltage approximately equal to half the excitation voltage.
  - Typical sensitivity:
    - 1-100 mV/kG
    - (Refrigerator magnet: 200 gauss)
  - Typical element resistance:
    - 1 to 10 ohms
  - Typical excitation current:
    - 20 to 200 mA.
  - Typical linearity:
    - 0.1% to 2%.

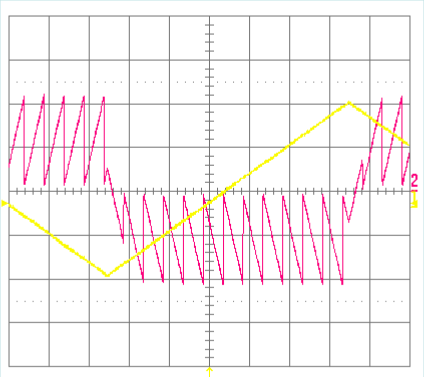




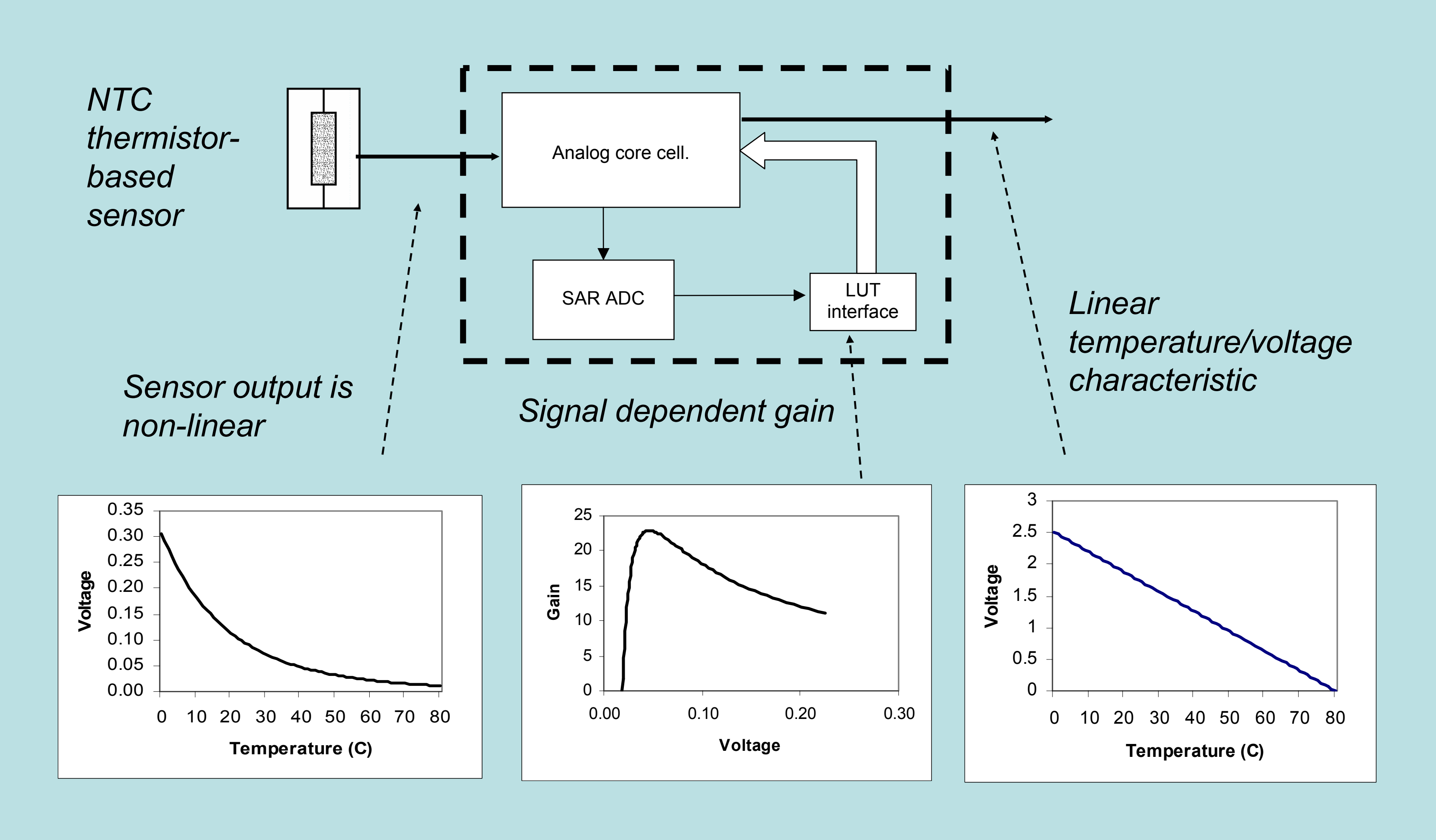
Optical-sensing

- **Photodiode/transistor**
  - Photo-generated current develops a voltage across a feedback “resistor”
  - Transimpedance is dependent on absolute capacitor values, ***not*** a capacitor ratio
  - Transimpedance CAM output is only valid at the end of each clock phase – follow by an appropriate CAM





Sensor Linearization



NTC thermistor-based sensor

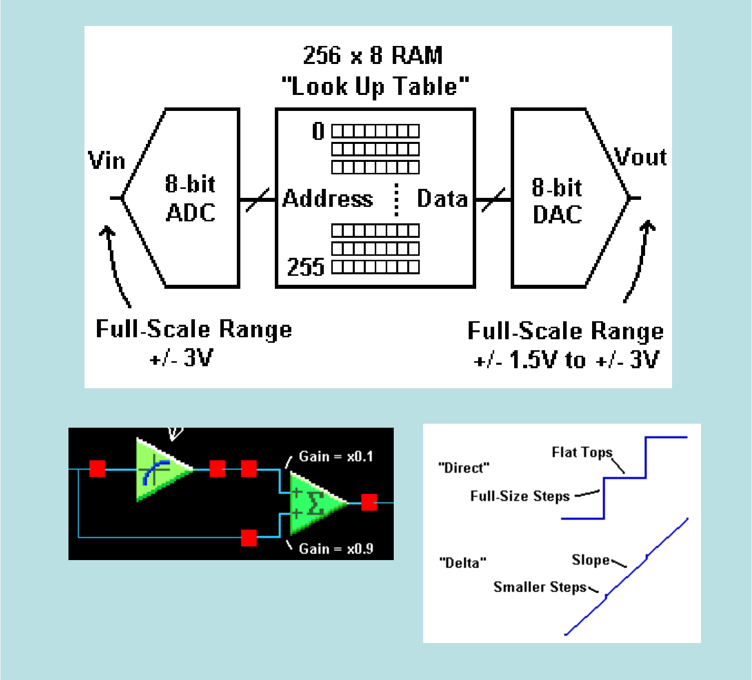
Sensor output is non-linear

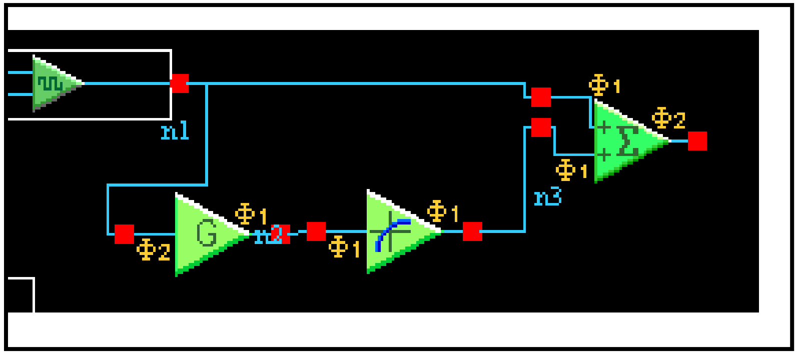
Signal dependent gain

Linear temperature/voltage characteristic

Sensor Linearization

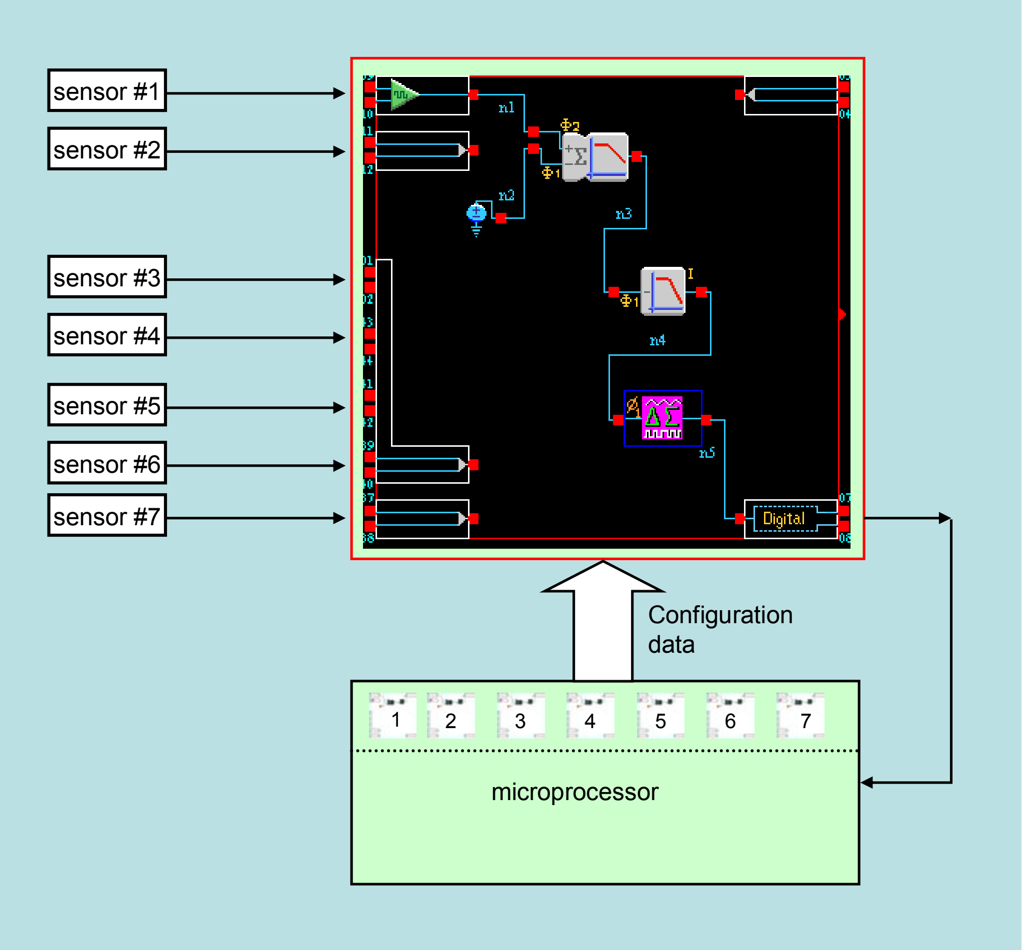
- **TransferFunction CAM**
  - Quantizes gain – use full-scale input and output for minimum non-linearity
  - When the input signal is nearly linear, use “delta” information rather than “direct”

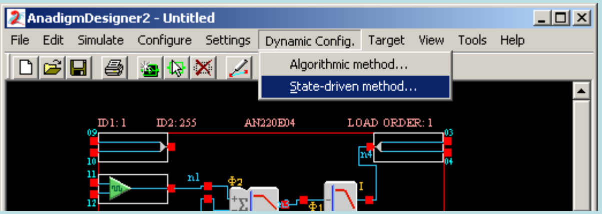




Hardware Multiplexing

- **State-driven dynamic reconfiguration:**





```
void main()
{
  // Get the Primary Configuration data
  int dataSize = 0;
  const an_Byte* pData = an_GetPrimaryConfigData(an_chip1_Primary,&dataSize);

  // Send it
  Download(pData, dataSize, 1);

  an_Circuit nCircuit = an_circuit1_chip1_Data;
  while (1)
  {

        // Work out which circuit is needed next
        switch(nCircuit)
        {
              case an_circuit1_chip1_Data:
                    nCircuit=an_circuit2__chip1_Data;
                    break;
              case an_circuit2_chip1_Data:
                    nCircuit=an_circuit3_chip1_Data;
                    break;

              // etc., for the other cases

              default:
                    nCircuit=an_circuit1_chip1_Data;
        }

        // Get the transition data
        dataSize = 0;
        pData = an_GetCircuitTransitionData(nCircuit, &dataSize);

        // Send it
        Download(pData, dataSize, 0);

        // Pause while the measurement is taken

  }

} //end of main()
```