EE247
Lecture 22

• Pipelined ADCs (continued)
  – Effect sub-ADC, gain stage, sub-DAC non-idealities on overall ADC
    performance (continued)
    • Correction for inter-stage gain nonlinearity
  – Implementation
    • Combining the bits
    • Practical circuits
    • Stage scaling
    • Stage implementation
      – Circuits
      – Noise budgeting
    • How many bits per stage?
  – Algorithmic ADCs utilizing pipeline structure
  – Advanced background calibration techniques

• Time Interleaved Converters
• ADC figures of merit

Pipeline ADC
Block Diagram

• Idea: Cascade several low resolution stages to obtain high overall resolution
  (e.g. 10bit ADC can be built with series of 10 ADCs each 1-bit only!)
• Each stage performs coarse A/D conversion and computes its quantization
  error, or "residue"
Summary So Far
Pipelined A/D Converters

- Cascade of low resolution stages
  - By adding inter-stage gain = 2^{B_{eff}}
  - No need to scale down Vref for stages down the pipe
  - Reduced accuracy requirement for stages coming after stage 1
- Addition of Track & Hold function to interstage-gain
  - Stages can operate concurrently
  - Throughput increased to as high as one sample per clock cycle
  - Latency function of number of stages & conversion-per-stage
- Correction for circuit non-idealities
  - Built-in redundancy compensate for sub-ADC inaccuracies such as comparator offset (interstage gain: G=2^{B_{eff}}, B_{eff} < B_{n})
  - Error associated with gain stage and sub-DAC calibrated out

### Pipelined ADC Error Correction/Calibration Summary

<table>
<thead>
<tr>
<th>Error</th>
<th>Correction/Calibration</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e_{ADC}, V_{os})</td>
<td>Redundancy either same stage or next stage</td>
</tr>
<tr>
<td>(e_{gain})</td>
<td>Digital adjustment</td>
</tr>
<tr>
<td>(e_{DAC})</td>
<td>Either sufficient component matching or digital calibration</td>
</tr>
<tr>
<td>Inter-stage amplifier non-linearity</td>
<td>?</td>
</tr>
</tbody>
</table>
Inter-stage Gain Nonlinearity

- Invert gain stage non-linear polynomial
- Express error as function of $V_{RES1}$
- Push error compensator into digital domain through backend ADC


$V_{RES1}$

$\Delta$

$2^3$

$\epsilon(V_{X})$

$\epsilon(V_{RES1})$

$D_B$

9+2

$D_{B,corr}$

$\epsilon(D_B, p_2)$

$p_2 = \frac{a_3}{(2^3 + \epsilon_{gain})}$

$\epsilon(D_B, p_2) = p_2D_B^3 - 3p_2^2D_B^5 + 12p_2^3D_B^7 + \ldots$

- Pre-measured & stored in table look-up form
- $p_2$ continuously estimated & updated (account for temp. & other variations)

Inter-stage Gain Nonlinearity Compensation
Proof of Concept Evaluation Prototype

- Re-used 14-bit ADC in 0.35μm from Analog Devices [Kelly, ISSCC 2001]
- Modified only 1st stage with 3-b gain to open-loop amplifier built with simple diff-pair + resistive load instead of the conventional feedback around high-gain amp
- Conventional 9-b backend, 2-bit redundancy in 1st stage
- Real-time post-processor off-chip (FPGA)


Measurement Results
12-bit ADC w Extra 2-bits for Calibration

(a) without calibration
(b) with calibration
Combining the Bits

- Example: Three 2-bit stages, no redundancy

\[ D_{\text{out}} = D_1 + \frac{1}{2^{B_1\text{eff}}} D_2 + \frac{1}{2^{B_2\text{eff}}} \cdot \frac{1}{2^{B_3\text{eff}}} D_3 \]

\[ D_{\text{out}} = D_1 + \frac{1}{4} D_2 + \frac{1}{16} D_3 \]

Combining the Bits

- Only bit shifts
- No arithmetic circuits needed
Combining the Bits
Including Redundancy

- Example: Three 2-bit stages, incorporating 1-bit redundancy in stages 1 and 2

\[ V_{in} \]

\[ B_1 = 3 \quad B_{1_{eff}} = 2 \]

\[ B_2 = 3 \quad B_{2_{eff}} = 2 \]

\[ B_3 = 2 \]

\[ \text{Stage 1} \]

\[ \text{Stage 2} \]

\[ \text{Stage 3} \]

“6 Wires”

???

\[ D_{out[5:0]} \]

Combining the Bits

- Bits overlap
- Need adders

\[ D_{out} = D_1 + \frac{1}{2^{B_{1_{eff}}}}D_2 + \frac{1}{2^{B_{2_{eff}}} \cdot 2^{B_{2_{eff}}}}D_3 \]

\[ D_{out} = D_1 + \frac{1}{4}D_2 + \frac{1}{16}D_3 \]

\[ D_1 \quad XXX \]

\[ D_2 \quad XXX \]

\[ D_3 \quad XX \]

\[ D_{out} \quad DDDDDD \]

\[ D_{out[5:0]} \]
Combining the Bits Example

Example

\[
\begin{align*}
D_1 &= 001 \\
D_2 &= 111 \\
D_3 &= 10 \\
D_{\text{out}} &= 011000
\end{align*}
\]

Pipeled ADC Stage Implementation

- Each stage needs T/H hold function
- Track phase: Acquire input/residue from previous stage
- Hold phase: sub-ADC decision, compute residue
Stage Implementation

- Usually no dedicated T/H amplifier in each stage (Except first stage in some cases – why?)
- T/H implicitly contained in stage building blocks

Stage Implementation

- DAC-subtract-gain function can be lumped into a single switched capacitor circuit
- "MDAC"
1.5-Bit Stage Implementation Example


1.5-Bit Stage Implementation Acquisition Cycle

1.5-Bit Stage Implementation

Conversion Cycle


\[ V_{DAC} \times C_s = -V_o \times C_f \]
\[ Q_{CF} = -V_{DAC} \times C_s \]
\[ Q_{Total} = Q_{CF} \times \Phi_1 \times \Phi_1 - V_{DAC} \times C_s \]
\[ V_o \times C_f = V_o \times C_f + V_o \times C_s - V_{DAC} \times C_s \]
\[ V_o = V_i \left( 1 + \frac{C_s}{C_f} \right) - V_{DAC} \times \frac{C_s}{C_f} \]
\[ C_s = C_f \]
\[ V_o = 2V_i - V_{DAC} \]

\[
\begin{array}{c|cc}
D1, D0 & V_{DAC} \\
\hline
1 & 1 & -V_R \\
0 & 1 & 0 \\
0 & 0 & +V_R \\
\end{array}
\]
1.5 Bit Stage Implementation Example

\[ V_o = \begin{cases} 
1 + \frac{C_i}{C_f} V_i - \frac{C_i}{C_f} V_{ref} & \text{if } V_i > V_{ref}/4 \\
1 + \frac{C_i}{C_f} V_i & \text{if } -V_{ref}/4 \leq V_i \leq +V_{ref}/4 \\
1 + \frac{C_i}{C_f} V_i + \frac{C_i}{C_f} V_{ref} & \text{if } V_i < -V_{ref}/4 
\end{cases} \]

Note:
- Interstage gain set by C ratios
- Accuracy better than 0.1%
- Up to 10bit level no need for gain calibration


1.5-Bit Stage Implementation
Timing of Stages

Conversion

Acquisition
Pipelined ADC Stage Power Dissipation & Noise

- Typically pipeline ADC noise dominated by inter-stage gain blocks
- Sub-ADC comparator noise translates into comparator threshold uncertainty and is compensated for by redundancy

\[ V_{\text{in}} \rightarrow \text{Stage 1} \rightarrow \text{Stage 2} \rightarrow \text{Stage 3} \rightarrow \]

\[ V_{\text{in}} \rightarrow G_1 \rightarrow G_2 \rightarrow G_3 \]

\[ V^{\text{in}}_{\text{noise}} = \sqrt{V_{n1}^2 + \frac{V_{n2}^2}{G_1^2} + \frac{V_{n3}^2}{G_1^2 G_2^2} + \cdots} \]

Pipelined ADC Stage Scaling

- Example: Pipeline using 1-bit\text{\_eff} stages

\[ V_{\text{in}} \rightarrow G_1=2 \rightarrow G_2=2 \rightarrow G_3=2 \]

\[ V_{\text{in}} \rightarrow \text{Gm} \rightarrow \text{Gm} \rightarrow \text{Gm} \rightarrow \]

\[ V_{\text{in}} \rightarrow C_1/2 \rightarrow C_2/2 \rightarrow C_3/2 \rightarrow \]

- Total input referred noise power:

\[ N_{\text{tot}} \propto kT \left[ \frac{1}{C_1} + \frac{1}{G_1^2 C_2} + \frac{1}{G_1^2 G_2^2 C_3} + \cdots \right] \]

\[ N_{\text{tot}} \propto kT \left[ \frac{1}{C_1} + \frac{1}{4C_2} + \frac{1}{16C_3} + \cdots \right] \]
Pipelined ADC Stage Scaling

- If all caps made the same size, backend stages contribute very little noise
- Wasteful power-wise, because:
  - Power $\sim Gm$
  - Speed $\sim Gm/C$
  - Fixed speed $\Rightarrow Gm/C$ fixed $\Rightarrow$ Power $\sim C$

\[ N_{\text{tot}} \propto kT \left[ \frac{1}{C_1} + \frac{1}{4C_2} + \frac{1}{16C_3} + \ldots \right] \]

- How about scaling caps down by $G^2=2^2=4x$ per stage?
  - Same amount of noise from every stage
  - All stages contribute significant noise
  - To keep overall noise the same $\Rightarrow$ noise/stage must be reduced
  - Power $\sim Gm \sim C$ goes up!
Stage Scaling
Example: 2-bit_{eff}/stage

Extreme 1: All Stages the Same Size

Noise Per Stage

Power Per Stage

Extreme 2: All Stages Contribute the Same Noise

Noise Per Stage

Power Per Stage

- Optimum capacitior scaling lies approximately midway between these two extremes

Ref: D. W. Cline, P.R. Gray "A power optimized 13-b 5MSamples/s pipelined analog-to-digital converter in 1.2um CMOS," JSSC 3/1996

Pipeline ADC
Stage Scaling

- Power minimum is "shallow"

- Near optimum solution in practice: Scale capacitors by stage gain

- E.g. for effective stage resolution of 1bit (Gain=2):

\[ N_{tot} \propto kT \left[ \frac{1}{C} + \frac{1}{2C} + \frac{1}{4C} + ... \right] \]
Stage Scaling Example


How Many Bits Per Stage?

• Many possible architectures
  – E.g. $B_{\text{eff}}=3$, $B_{\text{eff}}=1$, ...
  – vs. $B_{\text{eff}}=1$, $B_{\text{eff}}=1$, $B_{\text{eff}}=1$, ...

• Complex optimization problem, fortunately optimum tends to be shallow...

• Qualitative answer:
  – Maximum speed for given technology
    • Use small resolution-per-stage (large feedback factor)
  – Maximum power efficiency for fixed, "low" speed
    • Try higher resolution stages
    • Can help alleviate matching & noise requirements in stages following the 1st stage

Ref: Singer VLSI 96, Yang, JSSC 12/01 (14bit ADC w/o calibration)
### 14 & 12-Bit State-of-the-Art Implementations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>Architecture</td>
<td>3-1-1-1-1-1-1-1-1-1-1-1-1-3</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-2</td>
</tr>
<tr>
<td>SNR/SFDR</td>
<td>~73dB/88dB</td>
<td>~66dB/75dB</td>
</tr>
<tr>
<td>Speed</td>
<td>75MS/s</td>
<td>80MS/s</td>
</tr>
<tr>
<td>Power</td>
<td>340mW</td>
<td>260mW</td>
</tr>
</tbody>
</table>

### 10 & 8-Bit State-of-the-Art Implementations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Architecture</td>
<td>1.5bit/stage</td>
<td>2.8 - 2.8 - 4</td>
</tr>
<tr>
<td>SNR/SFDR</td>
<td>~55dB/66dB</td>
<td>~48dB/56dB</td>
</tr>
<tr>
<td>Speed</td>
<td>125MS/s</td>
<td>200MS/s</td>
</tr>
<tr>
<td>Power</td>
<td>40mW</td>
<td>30mW</td>
</tr>
</tbody>
</table>
Algorithmic ADC

- Essentially same as pipeline, but a single stage is reused for all partial conversions
- For overall \( B_{\text{overall}} \) bits \( \rightarrow \) need \( B_{\text{overall}} / B_{\text{stage}} \) clock cycles per conversion
  - Small area, slow
  - Trades conversion time for area

Least Mean Square Adaptive Digital Background Calibration of Pipelined Analog-to-Digital Converters

- Slow, but accurate ADC operates in parallel with pipelined (main) ADC
- Slow ADC samples input signal at a lower sampling rate \( (f/n) \)
- Difference between corresponding samples for two ADCs \( (e) \) used to correct fast ADC digital output via an adaptive digital filter (ADF) based on minimizing the Least-Mean-Squared error

Example: "A 12-bit 20-MS/s pipelined analog-to-digital converter with nested digital background calibration"

- Pipelined ADC operates at 20Ms/s @ has 1.5bit/stage
- Slow ADC → Algorithmic type operating at 20Ms/32=625ks/s
- Digital correction accounts for bit redundancy
- Digital error estimator → minimizes the mean-squared-error


Algorithmic ADC Used for Calibration of Pipelined ADC (continued from previous page)

- Uses replica of pipelined ADC stage
- Requires extra SHA in front to hold residue
- Undergoes a calibration cycle periodically prior to being used to calibrate pipelined ADC

12-bit 20-MS/s Pipelined ADC with Digital Background Calibration

Sampling capacitors scaled (1B_{eff}/stage):
• Input SHA: 6pF
• Pipelined ADC: 2pF, 0.9, 0.4, 0.2, 0.1, 0.1, ... 0.1
• Algorithmic ADC: 0.2pF

Chip area: 13.2mm²
• Does not include digital calibration circuitry estimated ~1.7mm²
• Area of Algorithmic ADC <20%


Measurement Results
12-bit 20-MS/s Pipelined ADC with Digital Background Calibration

Without Calibration
|INL|<4.2LSB

With Calibration
|INL|<0.5LSB

### Measurement Results

12-bit 20-MS/s Pipelined ADC with Digital Background Calibration

#### Performance Summary (3.3 V, 25 °C)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Without Cal.</th>
<th>With Cal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>0.35µm 2P4M CMOS</td>
<td></td>
</tr>
<tr>
<td>Sampling rate</td>
<td>20 Msamples/s</td>
<td></td>
</tr>
<tr>
<td>Active area</td>
<td>7.5 mm²</td>
<td></td>
</tr>
<tr>
<td>Full-Scale Input</td>
<td>1.6 Vp-p</td>
<td></td>
</tr>
<tr>
<td>Analog Power Diss.</td>
<td>190 mW</td>
<td></td>
</tr>
<tr>
<td>Total Power Diss.</td>
<td>217 mW</td>
<td></td>
</tr>
<tr>
<td>Max. [NL] (Pip. ADC)*</td>
<td>4.21 LSB</td>
<td>0.47 LSB</td>
</tr>
<tr>
<td>Max. [DNL] (Pip. ADC)*</td>
<td>0.66 LSB</td>
<td>0.41 LSB</td>
</tr>
<tr>
<td>SNDR (Alg. ADC)*</td>
<td>48.6 dB</td>
<td></td>
</tr>
<tr>
<td>SNDR (Pip. ADC)*</td>
<td>58.2 dB</td>
<td></td>
</tr>
<tr>
<td>SFDR (Pip. ADC)*</td>
<td>59.4 dB</td>
<td></td>
</tr>
<tr>
<td>THD (Pip. ADC)*</td>
<td>59.4 dB</td>
<td></td>
</tr>
<tr>
<td>CMRR*</td>
<td>73.6 dB</td>
<td></td>
</tr>
<tr>
<td>$f_{fs} = 58$ kHz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADC Architectures

- Slope type converters
- Successive approximation
- Flash
- Interpolating & Folding
- Residue type ADCs
  - Two-step Flash
  - Pipelined ADCs
- Time-interleaved / parallel converter
- Oversampled ADCs

Time Interleaved Converters

- Example:
  - 4 ADCs operating in parallel at sampling frequency $f_s$
  - Each ADC converts on one of the 4 possible clock phases
  - Overall sampling frequency = $4f_s$
  - Note T/H has to operate at $4f_s$!

- Extremely fast: Typically, limited by speed of T/H

- Accuracy limited by mismatch among individual ADCs (timing, offset, gain, …)
Time Interleaved Converters Timing

- Note: Effective sampling rate $\rightarrow 4f_s$

ADC Figures of Merit

- Objective: Establish measure/s to compare performance of various ADCs

- Can use FOM to combine several performance metrics to get one single number

- What are reasonable FOM for ADCs?
ADC Figures of Merit

\[ FOM_1 = f_s \cdot 2^{ENOB} \]

- This FOM suggests that adding an extra bit to an ADC is just as hard as doubling its bandwidth

- Is this a good assumption?

Ref: R. H. Walden, "Analog-to-digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, April 1999

Survey Data

Ref: R. H. Walden, "Analog-to-digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, April 1999
ADC Figures of Merit

\[ FOM_2 = \frac{\text{Power}}{f_s \cdot 2^{\text{ENOB}}} \quad [J / \text{conv}] \]

- Sometimes inverse of this metric is used
- In typical circuits power ~ speed, FOM$_2$ captures this tradeoff correctly
- How about power vs. ENOB?
  - One more bit 2x in power?


ADC Figures of Merit

- One more bit means...
  - 6dB SNR, 4x less noise power, 4x larger C
  - Power ~ Gm ~ C increases 4x

- Even worse: Flash ADC
  - Extra bit means 2x number of comparators
  - Each of them needs double precision
  - Transistor area 4x, Current 4x to keep same current density
  - Net result: Power increases 8x
ADC Figures of Merit

- FOM\(_2\) seems not entirely appropriate, but somehow still standard in literature, papers

- "Tends to work" because:
  - Not all power in an ADC is "noise limited"
  - E.g. Digital power, biasing circuits, etc.

- Better use FOM\(_2\) to compare ADCs with same resolution!

ADC Figures of Merit

\[ FOM_3 = \frac{\text{Power}}{\text{Speed}} \]

- Compare only power of ADCs with approximately same ENOB
- Useful numbers:
  - 10b (~9 ENOB) ADCs: 1 mW/MSample/sec
    Note the ISSCC 05 example: 0.33mW/MS/sec!
  - 12b (~11 ENOB) ADCs: 4 mW/MSample/sec
10-Bit ADC Power/Speed

12-Bit ADC Power/Speed
EECS 247 - Lecture 22: Data Converters - Nyquist Rate ADCs

Performance Trend

Bandwidth x Resolution [Hz-LSB]

$\approx 2x/5$ years


ADC Architectures

- Slope type converters
- Successive approximation
- Flash
- Interpolating & Folding
- Residue type ADCs
  - Two-step Flash
  - Pipelined ADCs
- Time-interleaved / parallel converter
- Oversampled ADCs
Analog-to-Digital Converters

• Two categories:
  – Nyquist rate ADCs \( f_{\text{sig}}^{\text{max}} \sim 0.5f_{\text{sampling}} \)
    • Maximum achievable signal bandwidth higher compared to oversampled type
    • Resolution limited to max. ~14 to 16 bits
  – Oversampled ADCs \( f_{\text{sig}}^{\text{max}} << 0.5f_{\text{sampling}} \)
    • Maximum possible signal bandwidth lower compared to nyquist rate ADCs
    • Maximum achievable resolution high (18 to 20bits!)

The Case for Oversampling

**Nyquist sampling:**

- Signal
- \( f_s > 2B + \delta \)
- Sampler
- "Nyquist" ADC
- DSP

**Oversampling:**

- Signal
- "wide" transition
- Sampler
- Oversampled ADC
- DSP

- Nyquist rate \( f_N \sim 2B \)
- Oversampling rate \( M = f_s/f_N >> 1 \)
Nyquist v.s. Oversampled Converters
Antialiasing Requirements

Nyquist Sampling
- $f_s \sim 2f_B$
- Anti-aliasing Filter
- $f_s \gg 2f_B$

Oversampling
- $f_s \gg 2f_B$

Input Signal
- $|X(f)|$
- $f_B$
- $f_s$
- $2f_s$

Oversampling Benefits
- Almost no stringent requirements imposed on analog building blocks
- Takes advantage of the availability of low cost, low power digital filtering
- Relaxed transition band requirements for analog anti-aliasing filters
- Reduced baseband quantization noise power
- Allows trading speed for resolution
ADC Converters
Baseband Noise

- For a quantizer with quantization step size $\Delta$ and sampling rate $f_s$:
  - Quantization noise power distributed uniformly across Nyquist bandwidth ($f_s/2$)
  - Power spectral density:
    $$N_e(f) = \frac{\Delta^2}{f_s} \left( \frac{1}{12} \right)$$
  - Noise is distributed over the Nyquist band $-f_s/2$ to $f_s/2$

Oversampled Converters
Baseband Noise

$$S_B = \int_{-f_s/2}^{f_s/2} N_e(f) df = \int_{-f_s/2}^{f_s/2} \left( \frac{\Delta^2}{f_s} \right) \left( \frac{1}{12} \right) df$$

where for $f_B = f_s/2$

$$S_{B0} = \frac{\Delta^2}{12}$$

$$S_B = S_{B0} \left( \frac{2f_B}{f_s} \right) = \frac{S_{B0}}{M}$$

where $M = \frac{f_s}{2f_B}$ = oversampling ratio
Oversampled Converters

Baseband Noise

\[ S_B = S_{B0} \left( \frac{2f_B}{f_s} \right) = S_{B0} \frac{f_s}{2f_B} \]

where \( M = \frac{f_s}{2f_B} \) = oversampling ratio

- 2X increase in \( M \)
  - \( \rightarrow \) 3dB reduction in \( S_B \)
  - \( \rightarrow \frac{1}{2} \) bit increase in resolution/octave oversampling

To further increase the improvement in resolution:
- Embed quantizer in a feedback loop (patented by Cutler in 1960s!)
  - Noise shaping (sigma delta modulation)

---

Pulse-Count Modulation

\( V_{in} = 2/8 \)

Nyquist ADC

\( 010 \)

Oversampled ADC, \( M = 8 \)

Mean of pulse-count signal approximates analog input!
**Pulse-Count Output Spectrum**

- Signal band of interest: low frequencies, $f < B << f_s$
- Quantization error: high frequency, $B \cdots f_s / 2$
- Separate with digital low-pass filter!

**Oversampled ADC Predictive Coding**

- Quantize the difference signal rather than the signal itself
- Smaller input to ADC $\rightarrow$ Buy dynamic range
- Only works if combined with oversampling
- 1-Bit digital output
- Digital filter computes “average” $\rightarrow$ N-bit output
Oversampled ADC

Decimator:
- Digital (low-pass) filter
- Removes quantization noise for \( f > B \)
- Provides anti-alias filtering for DSP
- Narrow transition band, high-order (digital filters with high order consume significantly smaller power & area compared to analog filters)
- 1-Bit input, N-Bit output (essentially computes “average”)

Modulator or Analog Front End (AFE)

- Objectives:
  - Convert analog input to 1-Bit pulse density stream
  - Move quantization error to high frequencies \( f >> B \)
  - Operates at high frequency \( f_s >> f_N \)
    - \( M = 8 \ldots 256 \) (typical)\ldots1024
    - Since modulator operated at high frequencies → need to keep analog circuitry “simple”

→ \( \Sigma A = \Delta \Sigma \) Modulator
Sigma-Delta Modulators

Analog 1-Bit $\Sigma\Delta$ modulators convert a continuous time analog input $v_{IN}$ into a 1-Bit sequence $D_{OUT}$

The loop filter $H$ can be either switched-capacitor or continuous time.
- Switched-capacitor filters are "easier" to implement + frequency characteristics scale with clock rate.
- Continuous time filters provide anti-aliasing protection.
Oversampling A/D Conversion

- Analog front-end $\rightarrow$ oversampled noise-shaping modulator
  - Converts original signal to a 1-bit digital output at the high rate of $(2BM)$
- Digital back-end $\rightarrow$ digital filter (decimator)
  - Removes out-of-band quantization noise
  - Provides anti-aliasing to allow re-sampling @ lower sampling rate

$B = \frac{f_s}{2M}$

1st Order $\Sigma\Delta$ Modulator

1st order modulator, simplest loop filter $\rightarrow$ an integrator

$H(z) = \frac{z^{-1}}{1 - z^{-1}}$

Note: Non-linear system with memory $\rightarrow$ difficult to analyze