Some applications of multirate techniques

Signal Processing Systems Fall 2025

Lecture 9 (Monday 24.11.)

Outline

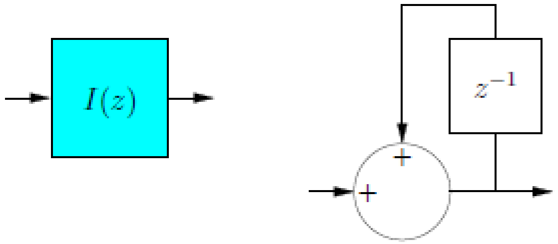
- Decimation & interpolation using CIC filters
- Multirate narrowband FIR filtering
- Oversampling A/D conversion

1. Cascaded integrator-comb (CIC) filters

- Alternative way of representing antialiasing/-imaging filters for sample rate conversions
- Well suited for HW implementations: no multipliers, efficient recursive computations
- CIC filters are constructed by cascading two components, integrators and combs
- First-order CIC filter = one integrator + one comb
    - Moving average filter, when scaled with $1/D$
    - Low-pass filter: sinc-shaped frequency response
- Frequency response can be controlled by cascading several integrator-comb pairs

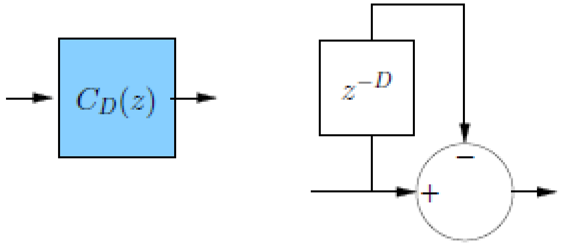
INTEGRATOR

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



COMB

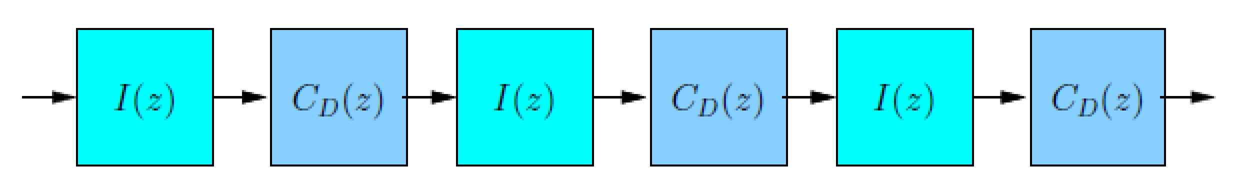
$$C_D(z) = 1 - z^{-D}$$



FIRST-ORDER CIC FILTER

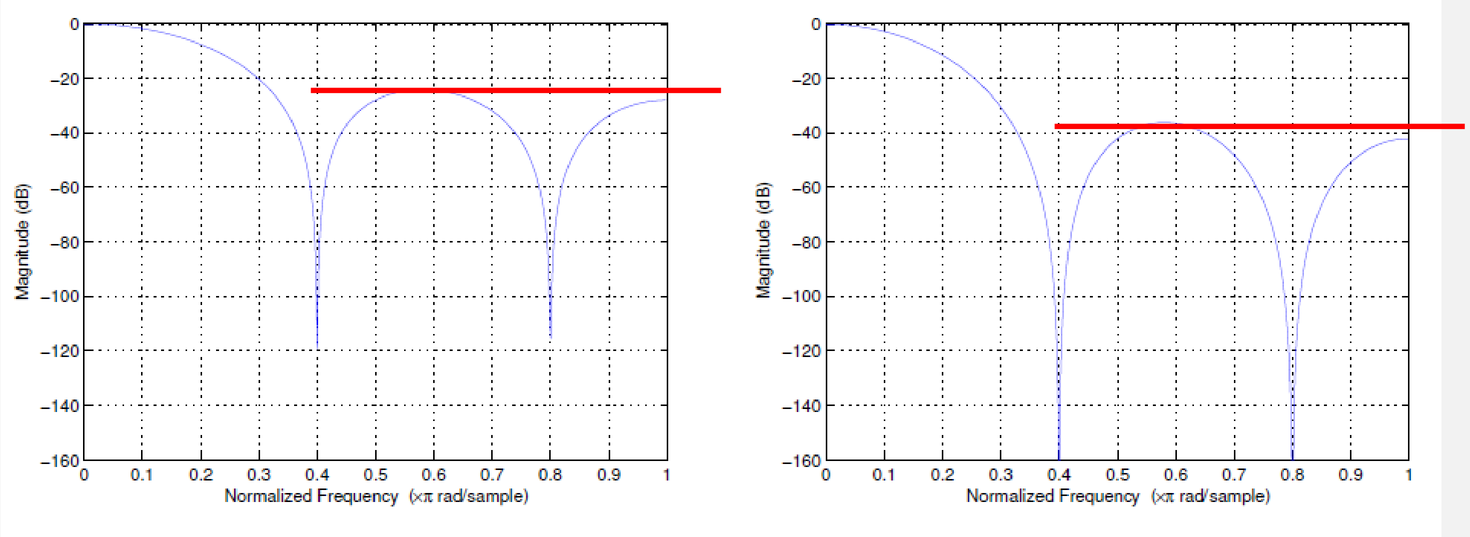
$$\frac{1}{D}C_D(z)I(z) = \frac{1}{D}\underbrace{\frac{1 - z^{-D}}{1 - z^{-1}}}_{\text{running sum filter}} = \underbrace{\frac{1}{D}\sum_{d=0}^{D-1} z^{-d}}_{\text{moving average filter}}$$

CIC filter of order N = 3



Note: The integrator and comb are linear systems and we can put the blocks in any order here.

- Comparison of frequency responses: 2 versus 3 stages (D = 5)
- In the latter case, more attenuation in the stop-band
- Note that the filters are far from ideal for low-pass filtering
  - For example, the passband is not flat



(a) Frequency response ($N = 2$). (b) Frequency response ($N = 3$).

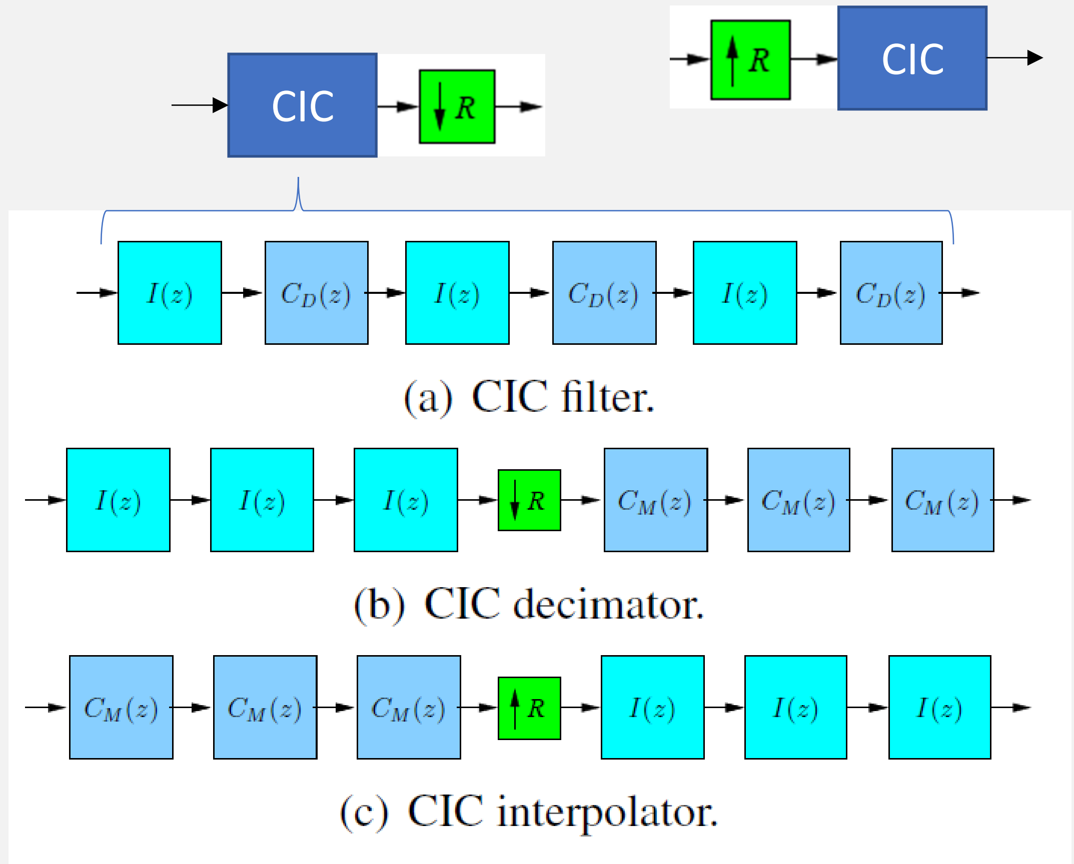
CIC decimators and interpolators

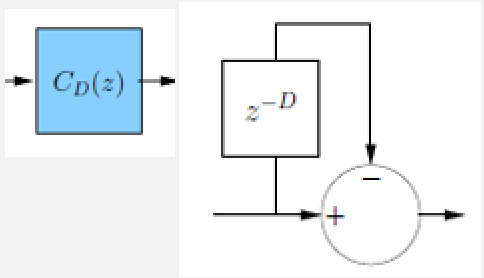
Decimation: we organize the components of the filter so that a cascade of N integrators is followed by a N-stage cascade of combs. The downsampler, which follows the filter, can then be moved between these cascades using the noble identities.

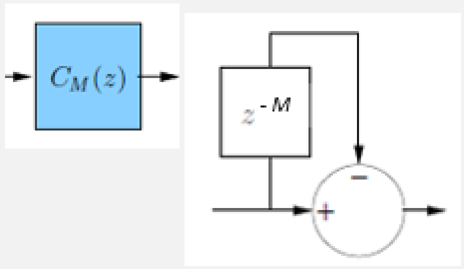
Interpolation: the combs are followed by the integrators. The upsampler, which preceeds the filter, can then be moved between the cascades.

In the CIC filter used, delay factor ***D = RM***, where *R* is the rate change factor and *M* is a design factor (typ. 1 or 2).
When the noble identities are applied, the comb stages get the delay factor equal to *M*. Thus, $C_D(z) = C_M(z^R)$.

As shown, CIC filters are far from ideal. To tolerate this, the input signal should be <u>narrowband</u>. In addition, a CIC rate changer is typically accompanied by a FIR filter, which make the <u>passband response flat</u>.







CIC filtering is a fixed-point technique

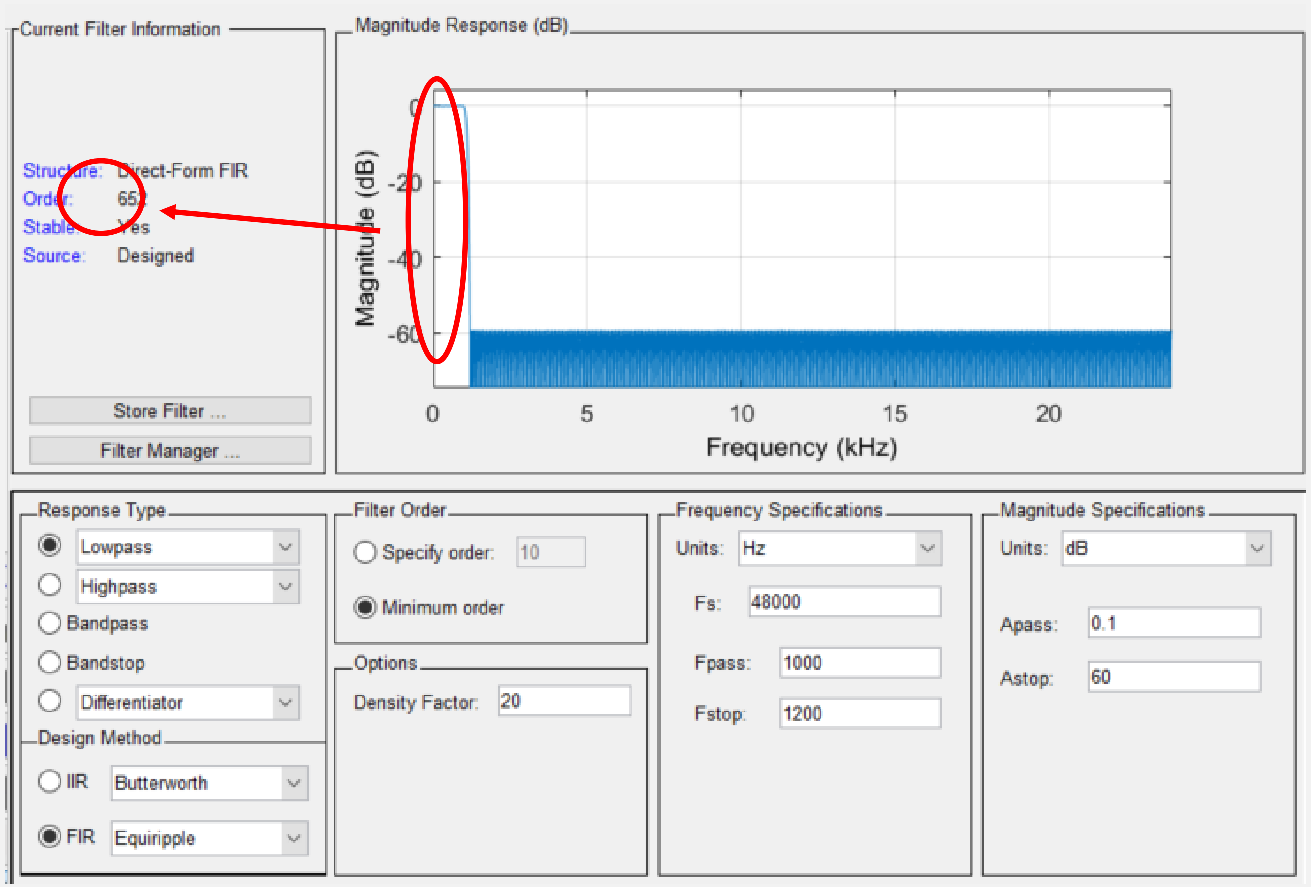
- Design Task 4 Problem 3
- It is essential that CIC filter is implemented using fixed-point two’s complement arithmetic

2. Multirate implementation of filters

- **Problem**: narrow passband FIR filter with short transition band. A large number of coefficients in direct implementation.
- **Example**.

Sampling frequency F_0	48 kHz
Passband cutoff frequency f_p	1000 Hz
Stopband cutoff frequency f_s	1200 Hz
Maximum passband ripple A_p	0.1 dB
Minimum stopband attenuation A_s	60 dB

653 coefficients needed !!!
Computation rate:
48,000 x 653 MAC/s (31.344 x 10^6)

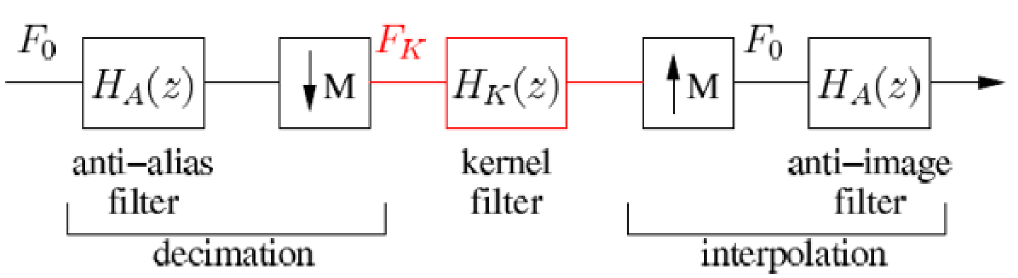


- **How to reduce the rate of MAC operations?**

Answer: multirate design

Approach:

1. **Decimate** to lower sample rate
    - Using one or more decimation filter stages
2. **Filter**: Perform filtering at lower sample rate F_K
    - Cut-off frequencies from original specification
    - Called kernel filter
3. **Interpolate** back to the original sample rate
    - Using one or more interpolation stages

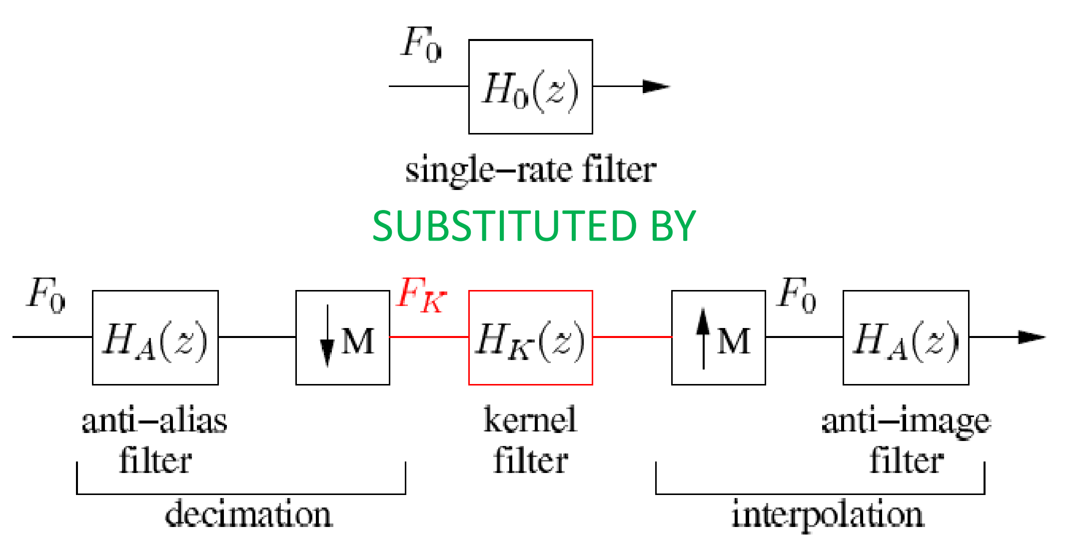


Why it can be efficient?

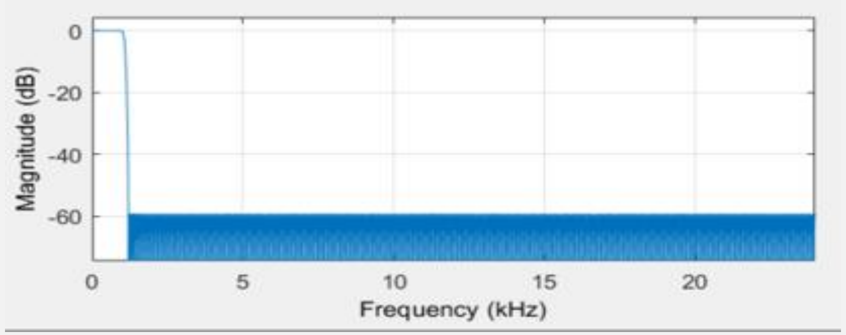
- Decimation and interpolation implemented using noble identity / polyphase decomposition based techniques
    - See previous lecture
- Kernel filtering versus original ”single-rate” filtering
    - Kernel filter has lower number of coefficients
    - It has also lower sample rate (lower MAC operation rate)

1.1. Single-stage multirate design

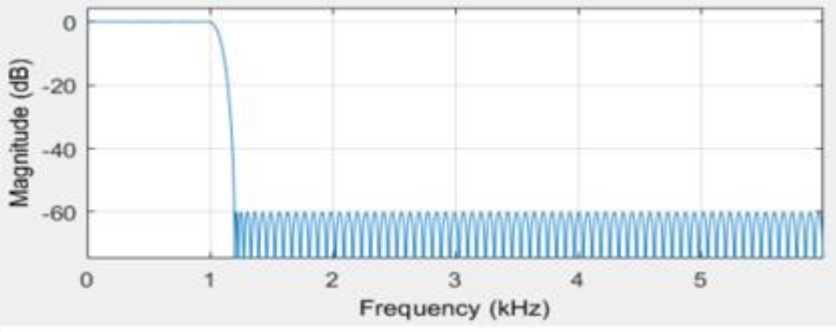
- Reduction of the sample rate can be done using **one or more** decimation stages. Considering single-stage design first
- Passband and stopband cutoff frequencies of the **kernel filter** $H_K(z)$ correspond to the cutoffs of the original filter, f_p and f_s
- The same filter $H_A(z)$ is used for both antialiasing and anti-imaging
- Antialias low-pass filtering must be implemented so that no signal is aliased from the higher frequencies to the passband range $[0, f_p]$



F_0, F_K denote sample rates



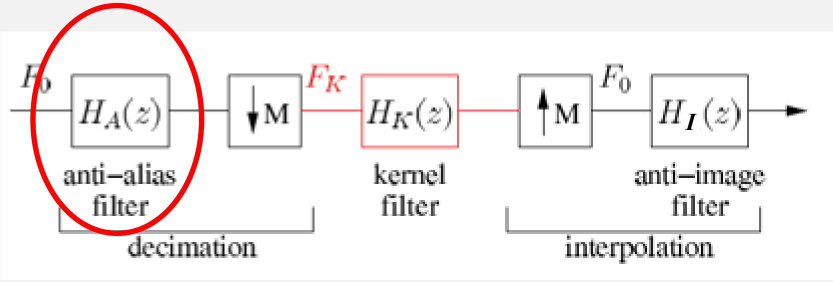
Single-rate filter

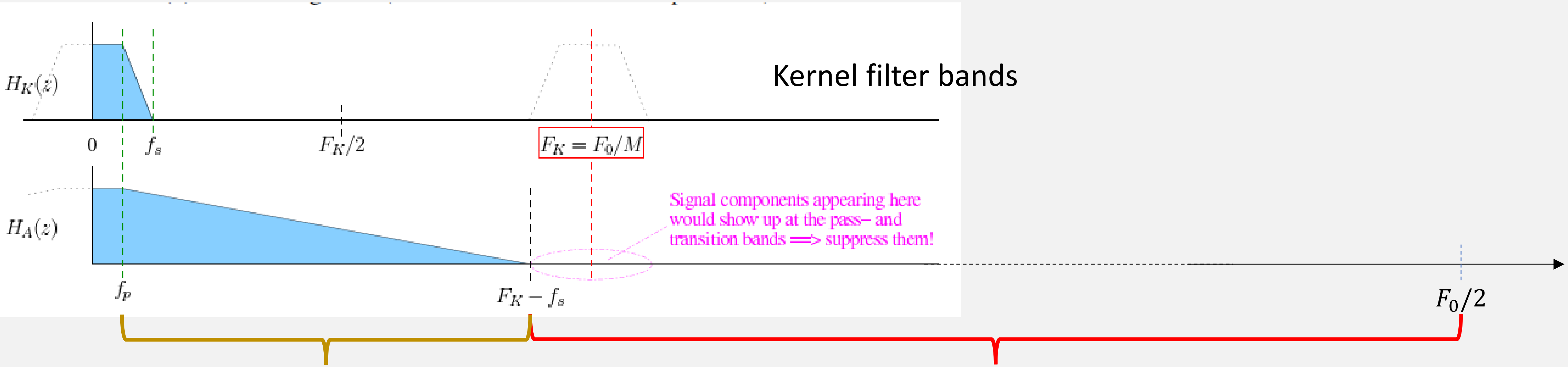


Kernel filter (M=4)

Specifying anti-alias filter

- Antialias low-pass filtering must be implemented so that no signal is aliased from the higher frequencies to the kernel’s pass- and transition band
- Sketch of the situation:



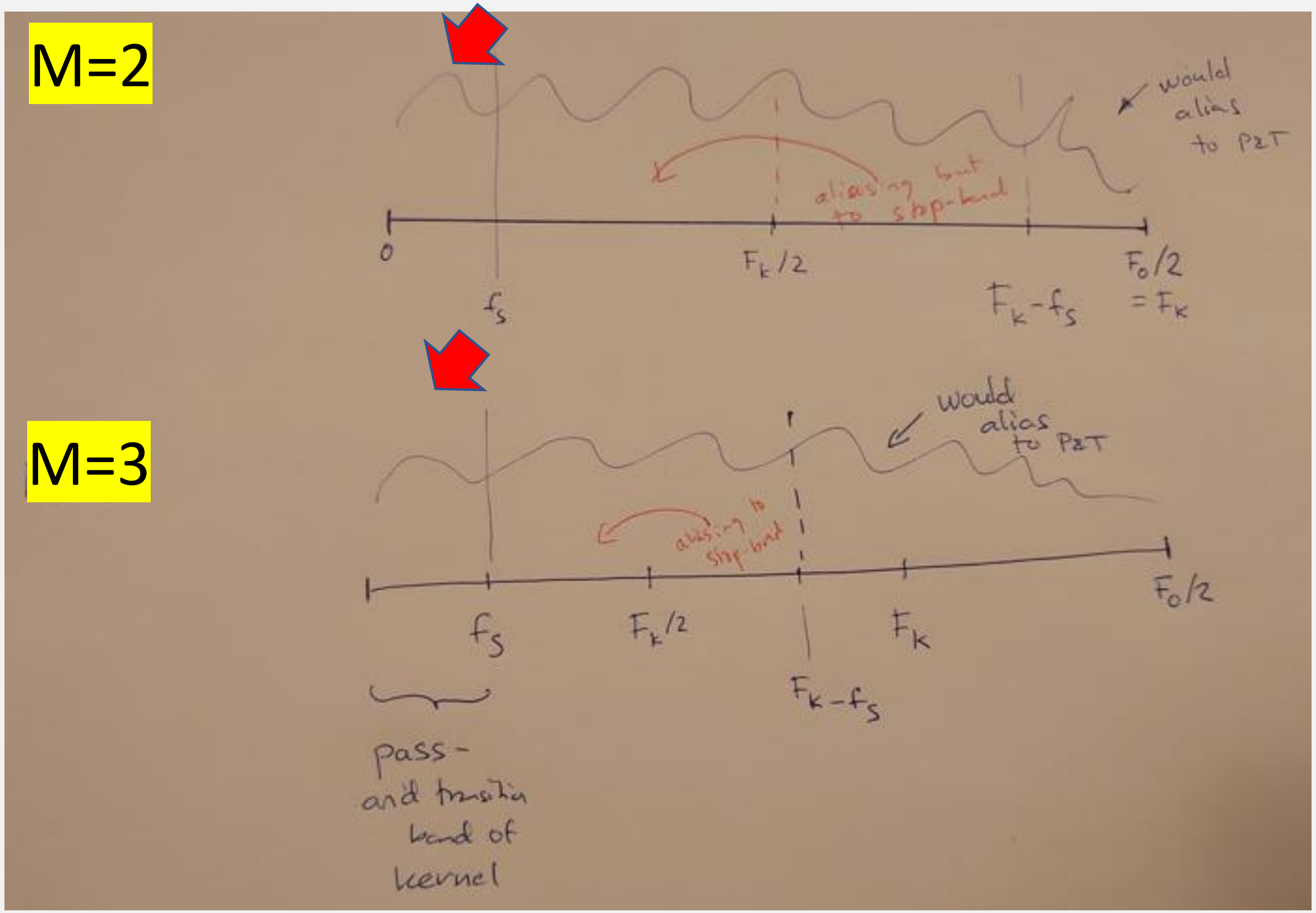


long transition band
=> **not many coefficients**

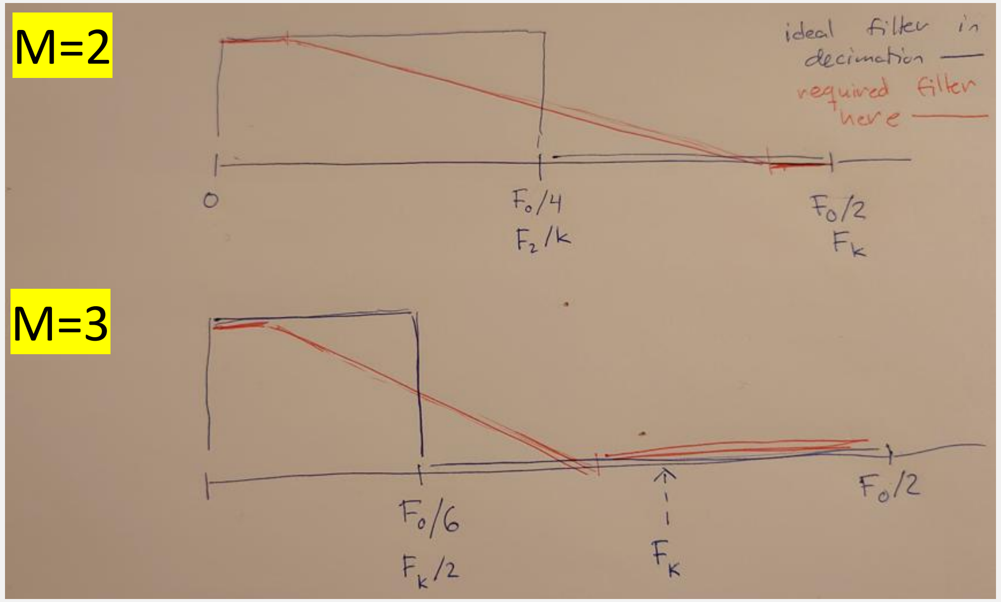
Suppressed by the anti-alias filter

Examples. M = 2, 3

AVOID ALIASING TO THIS REGION



IDEAL DECIMATION FILTERS

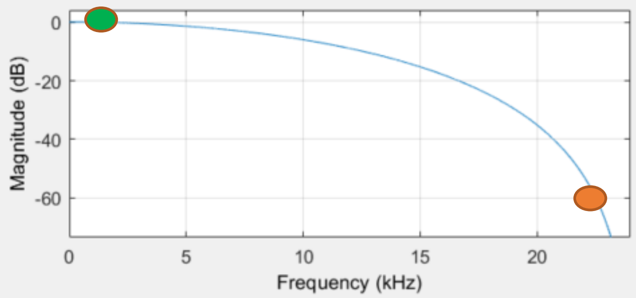


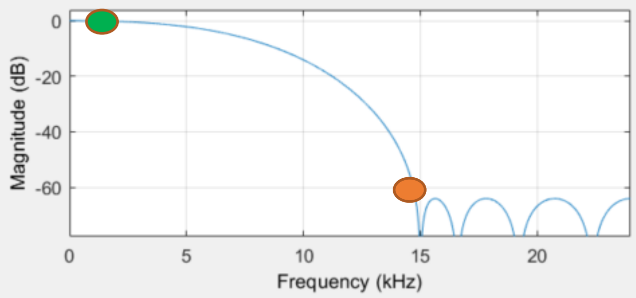
Ideal antialias filter not needed! Just filter what is necessary: do not care about stop-band aliasing when choosing cutoffs as kernel filter takes care of it.
Result: Small number of anti-alias filter coefficients

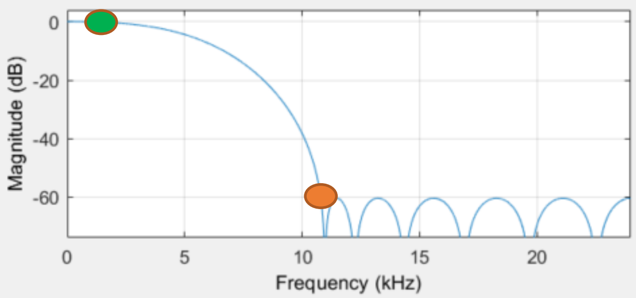
Example. Anti-alias filter designs

Designs for downsampling factors $M = 2, 3, 4$
Kernel filter sample rate $F_K = F_0 / M$
Antialias stopband cutoff $F_K - f_s$ $f_s = 1200$
Same passband ripple and stopband attenuation as in

Downsample factor M	Antialias filter sample rate	Antialias filter passband cutoff	Kernel filter sample rate (F_K)	Antialias filter stopband cutoff	Number of filter coefficients
2	48000	1000	24000	**22800**	**4**
3	48000	1000	16000	**14800**	**11**
4	48000	1000	12000	**10800**	**15**







MAC operations per second in **polyphase decomposition based** implementation:

Downsample factor M	MAC operations per second
2	192000/2 = 96000
3	528000/3 = 176000
4	720000/4 = 180000

When M is increased, there is increase in downsampling computations.

But, the kernel filter can be operated at lower and lower rate and its length is reduced.

Example. Kernel filter designs

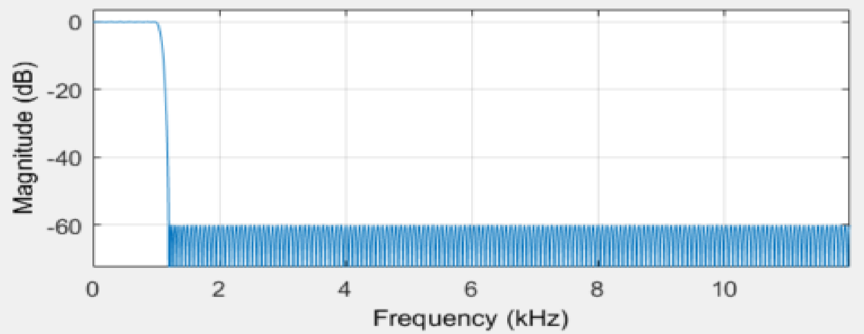
Designs for downsampling factors M = 2, 3, 4
Kernel filter sample rate $F_K = F_0 / M$
Passband and stopband cutoffs from narrownband filter spec
Same passband ripple and stopband attenuation

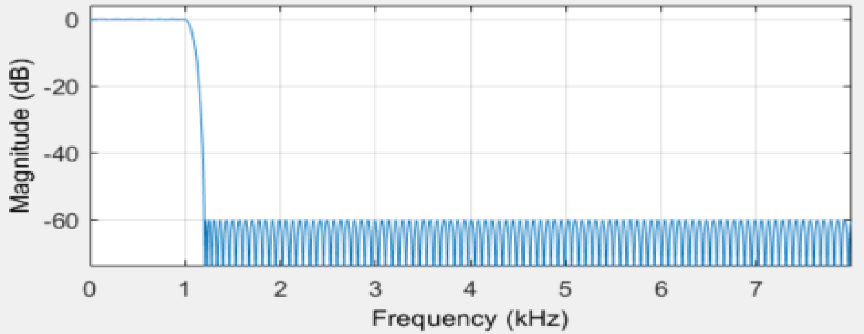
Downsample factor M	Kernel filter sample rate	Kernel filter passband cutoff	Kernel filter stopband cutoff	Number of filter coefficients
2	**24000**	1000	1200	**330**
3	**16000**	1000	1200	**221**
4	**12000**	1000	1200	**166**
Original	48000	1000	1200	653

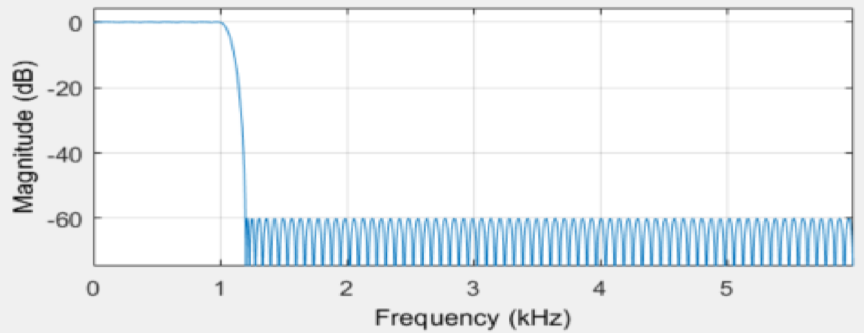
MAC operations per second in direct implementation:

Downsample factor M	MAC operations per second
2	7,920,000
3	3,536,000
4	1,992,000
Original	31,344,000

Note: there is clear advantage in increasing the downsample factor M, roughly proportional to M^2
Optimum: balance between decimation / interpolation and kernel computational complexities

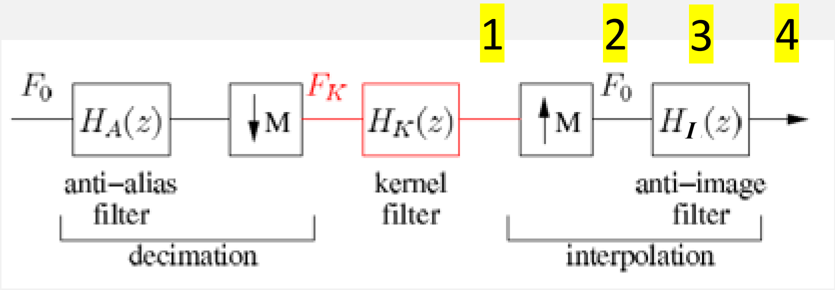






Effect of upsampling

- When we upsample the signal, imaging components are introduced to the signal



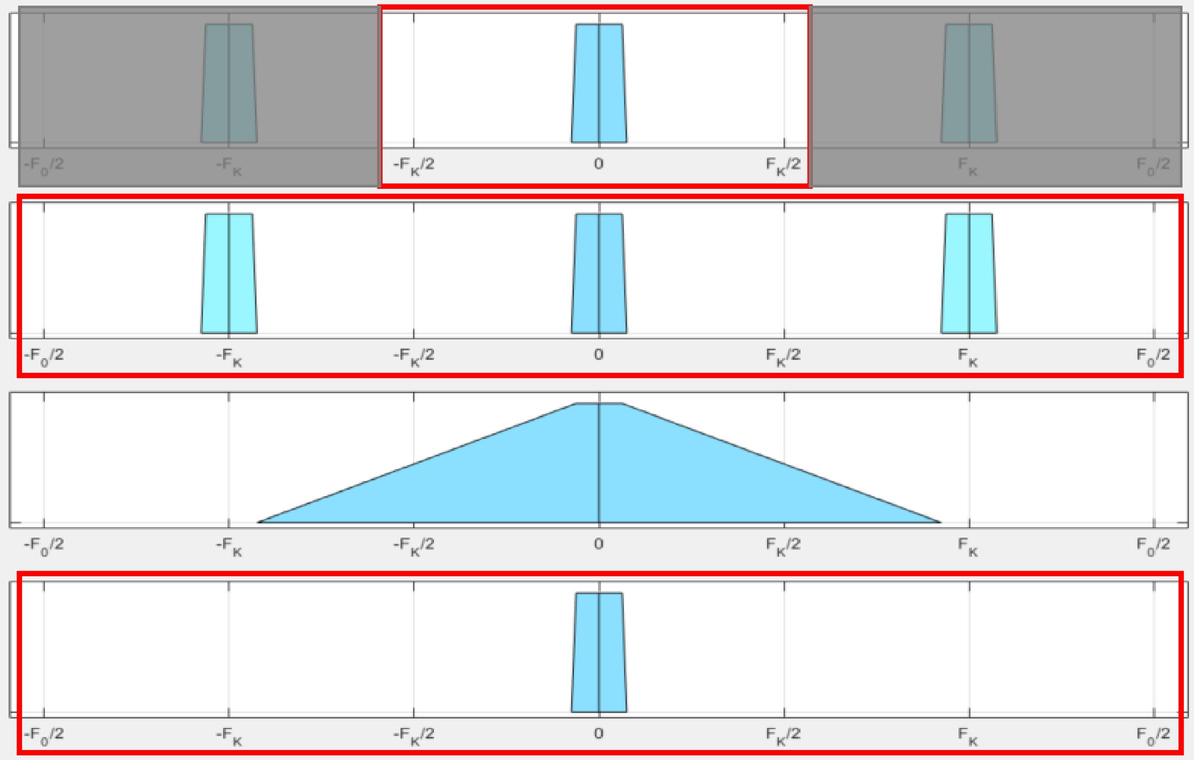
Example. M = 3

1 Output of kernel filter, sample rate F_K

2 Output of upsampling filter, sample rate $F_0 = 3F_K$

3 Filter for suppressing image components

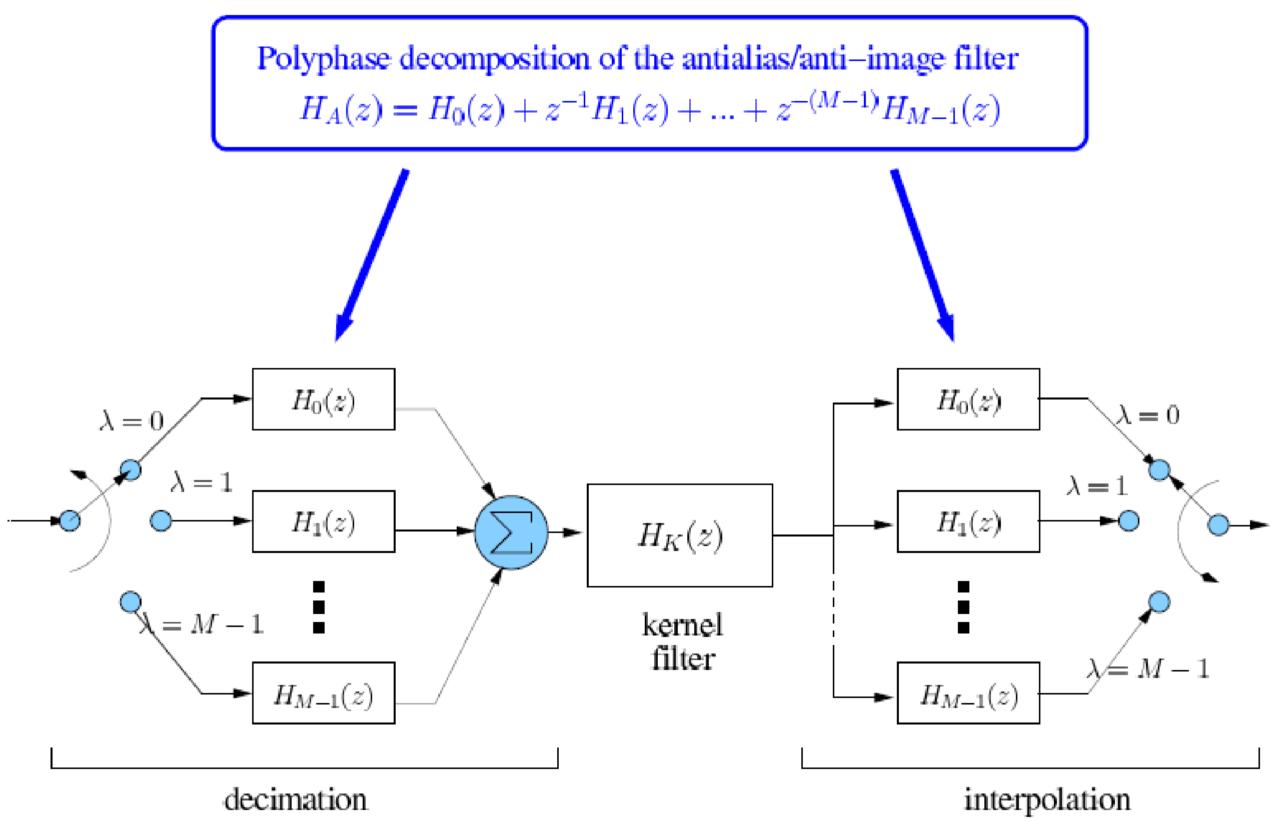
4 Interpolation output, sample rate F_0



The anti-image filter has same passband and stopband cutoff frequency specifications as the anti-alias filter. So, we use

$$H_I(z) = H_A(z)$$

Resulting multirate filter design



Computational complexity (number of MAC operations per second):

$$
\begin{aligned}
R_{\text{mac}} &= F_0 \times N_A \times (1/M) && \text{decimation} \\
&\quad + (F_0/M) \times N_K && \text{kernel} \\
&\quad + F_0 \times N_A \times (1/M) && \text{interpolation} \\
&= F_0 \times (N_K/M + 2N_A/M)
\end{aligned}
$$

Example. MAC rates for different values of M

M	Decimation	Kernel	Interpolation	Total
2	96,000	7,920,000	96,000	**8,112,000**
3	176,000	3,536,000	176,000	**3,888,000**
4	180,000	1,992,000	180,000	**2,352,000**
5	192,000	1,276,800	192,000	**1,660,800**
6	200,000	888,000	200,000	**1,288,000**
8	210,000	504,000	210,000	**924,000**
10	288,000	316,800	288,000	**892,800**
12	304,000	224,000	304,000	**832,000**
14	394,000	164,600	394,000	**952,600**
18	789,300	104,000	789,300	**1,682,600**

Note: As M increases, the kernel filter length reduces and the antialias/image filter length increases. Due to polyphase decompositions, decimation and interpolation MAC rates **do not increase drastically** when filter length increases.

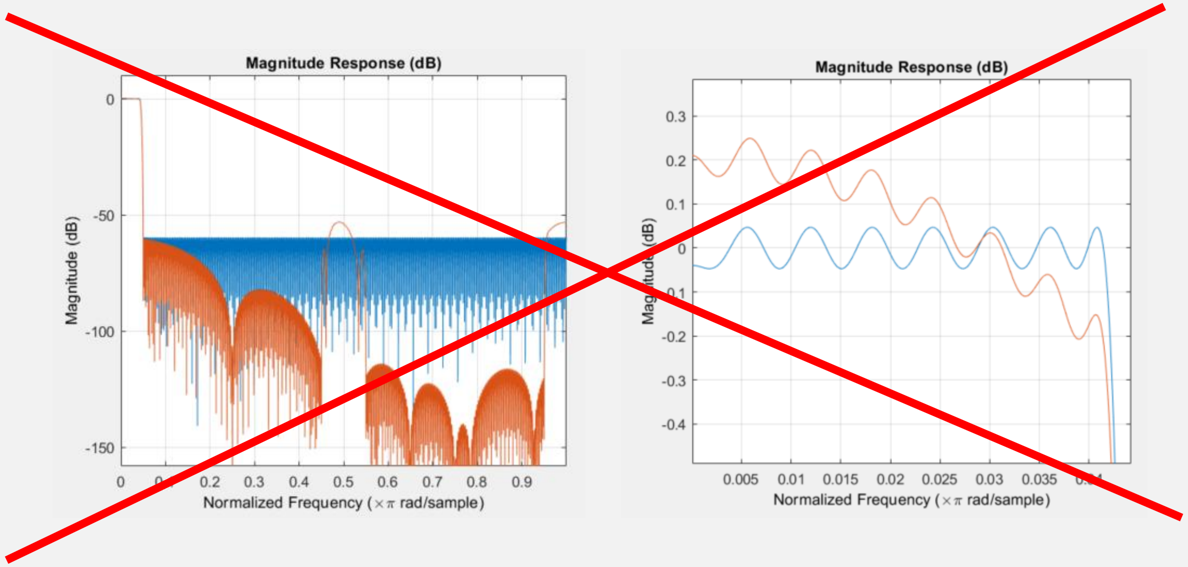
Note. Direct implementation : 31,344,000 MAC/s (48000 x 653)
Best in table (M=12) : 832,000 MAC/s. **Only 2,65%.**

Response of the filter

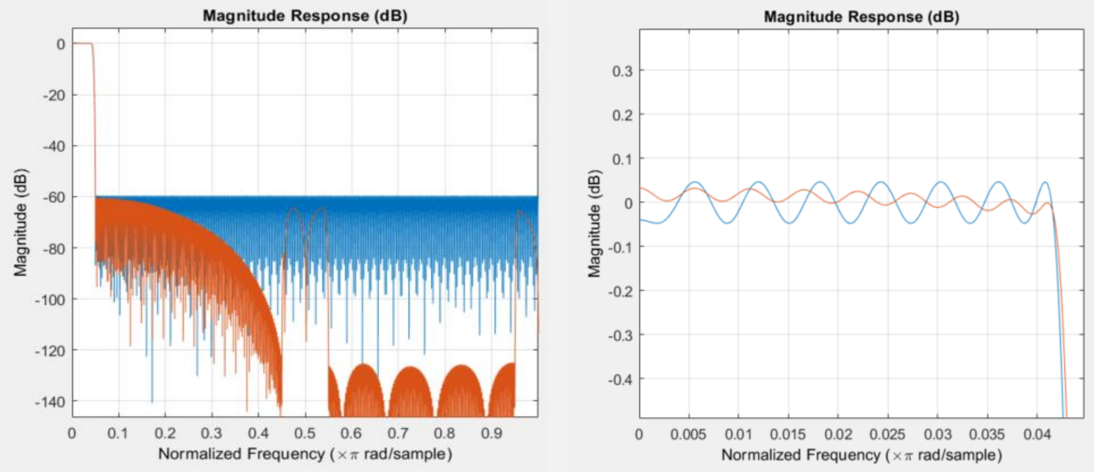
Requirement was:

Maximum passband ripple A_p	0.1 dB
Minimum stopband attenuation A_s	60 dB

For all: Passband ripple 0.1 dB, stopband attenuation 60 dB



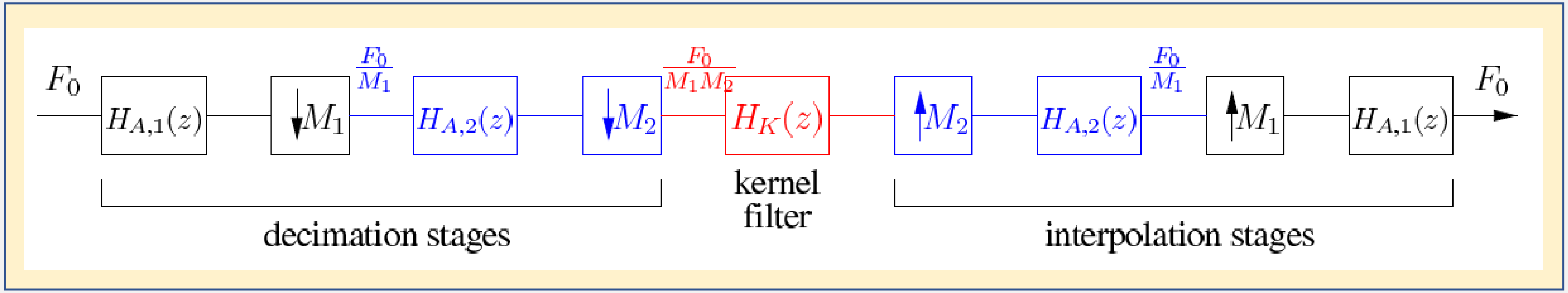
Kernel: 0.03 dB, 61 dB, antialias/image: 0.02 dB, 65 dB



- Simulating the resulting multirate filter to get its impulse response
- Mapping it to the frequency response for inspection
  - How well does it match the given specification?
  - What passband ripples and stopband attenuations should be used for the antialias/image filter and kernel filter?
- **Example**. Simulation for the example case, M = 4
  - Tool: multirateN.m in Design Task 4

1.2. Multistage design

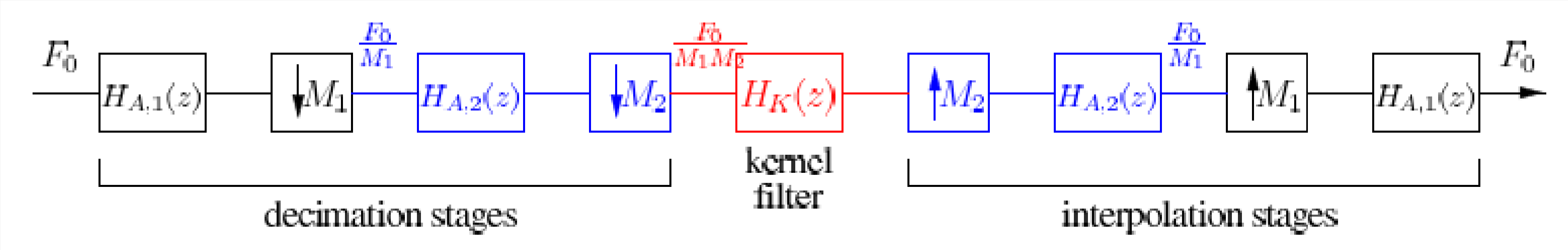
- Decimation to the kernel filter sample rate can be done gradually using multiple decimation stages.
- Symmetrically, the sample rate is increased to the original rate by multiple interpolation stages
- May provide some advantage in implementation compared to the single-stage design
  - In practice, experimental comparisons needed



- See **intro4b.pdf** (Sec. 3) for guidelines on choosing the design parameters and formulae for computational complexity analysis (**needed in DT4 Problem 2**)

Intro4b.pdf (Section 3)

- Choice of parameters
  - Choice of down/upsampling factors
  - Cutoff frequencies of antialias/anti-image filters
  - Setting passband ripples and stopband attenuations
- Analysis of computational complexity
  - How to compute the rate of MAC operations



Summary: multirate implementation of filters

- FIR filtering can have huge computational requirement in the case of narrow passbands and short transition bands
- Multirate approach can be used to reduce computational requirements of filtering. Many reasons for that:
  - Kernel filter has wider passband & transition band (w.r.t. operating sample rate) => less coefficients
  - Kernel filter is operating at lower rate
  - Antialias/image filters do not have to be ideal for associated decimations/interpolations => less coefficients
  - Antialias/image filters can be implemented using polyphase decomposition and commutation

3. Oversampling A/D conversion

- For certain ADC word length, we have certain quantization error power

$$N_{quant} = \frac{1}{\Delta}\int_{-\Delta/2}^{+\Delta/2} x^2 dx = \frac{\Delta^2}{12}$$

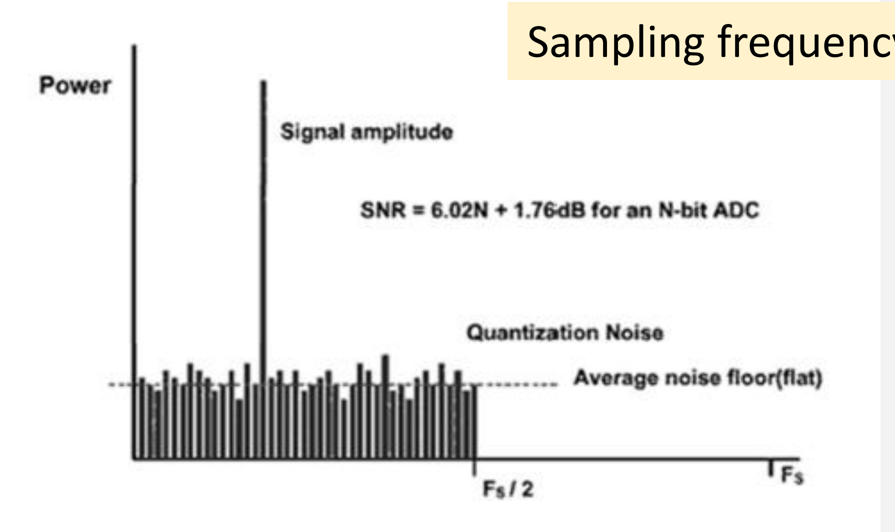
- Spectrum of the quantization noise is approximately flat, that is, the power spectral density (PSD) for sample rate F_s is

$$S(f) = \frac{N_{quant}}{F_s} \qquad \int_{-F_s/2}^{+F_s/2} S(f)\mathrm{d}f = N_{quant}$$

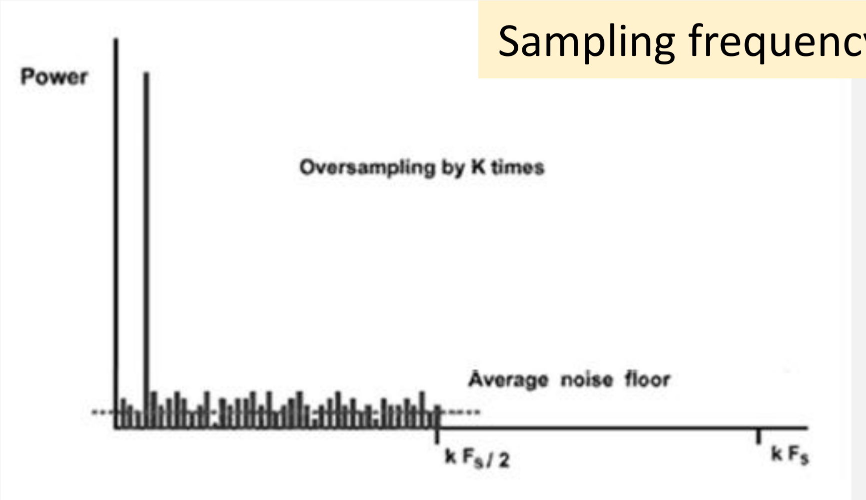
- The higher the sample rate is the more distributed the quantization noise is

$\Rightarrow$ The ADC noise floor is lower for higher sampling frequency!

Sampling frequency F_s

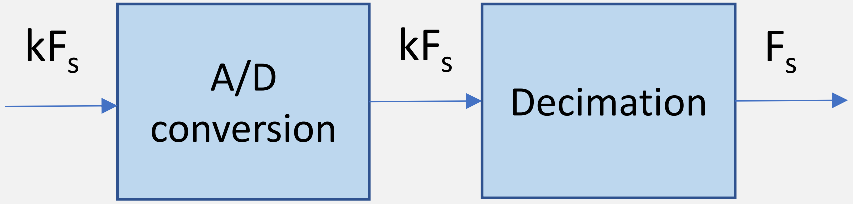


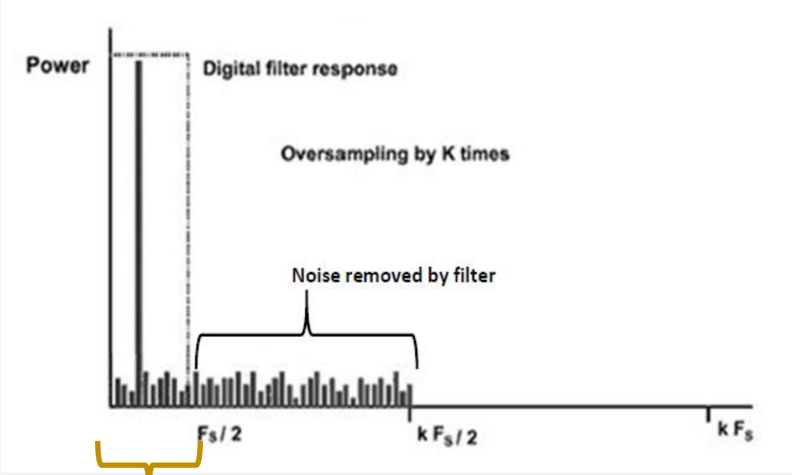
Sampling frequency $k \times F_s$



Suppressing quantization noise

- This analysis suggests that we can filter out portions of quantization noise by
  - Using high sample rate for ADC
  - Decimating the ADC output signal **to the target sample rate** Fs





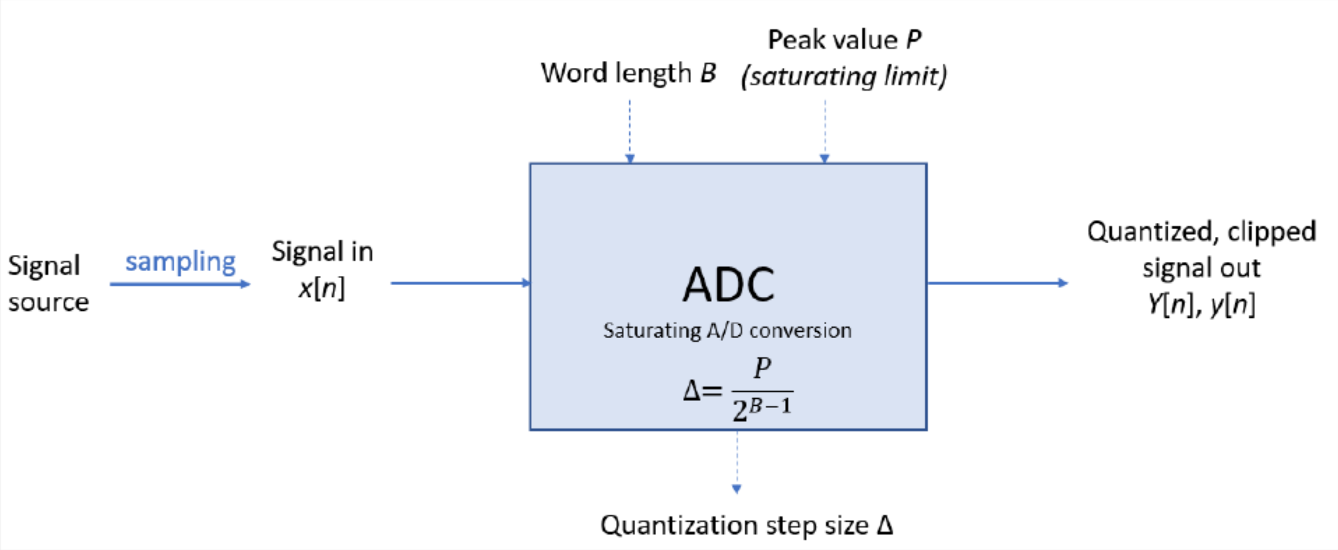
- In decimation, we use a lowpass filter, whose passband has width $F_s/2$
  - The quantization noise in the range $[\,F_s/2, kF_s/2]$ is suppressed
  - We get

$$N' = \int_{-F_s/2}^{+F_s/2} \frac{N_{quant}}{kF_s}\, df = \frac{\Delta^2}{12k} = \frac{(\Delta/\sqrt{k})^2}{12}$$

  - **So, effective quantization level difference is now $\Delta/\sqrt{k}$**

Relationship between word length & quantization step size

Saturating ADC:



$$\Delta = \frac{P}{2^{B-1}} \Rightarrow B = 1 + \log_2 \frac{P}{\Delta}$$

Tradeoff : oversampling / ADC word length

- For the k times oversampled A/D conversion, the effective quantization level difference is $\Delta/\sqrt{k}$

- Quantization level difference Δ' corresponds to the word length $B' = 1 + \log_2 \frac{P}{\Delta'}$

- For $\Delta' = \Delta/\sqrt{k}$, we get

$$B' = 1 + \log_2 \frac{P}{\Delta/\sqrt{k}} = 1 + \log_2 \frac{P}{\Delta} + \log_2 \sqrt{k} = B + \frac{1}{2} \log_2 k$$

Conclusion: If we use 4^n $\times$ oversampling, we can reduce the A/D word length by n bits and the quantization noise is still at the same level

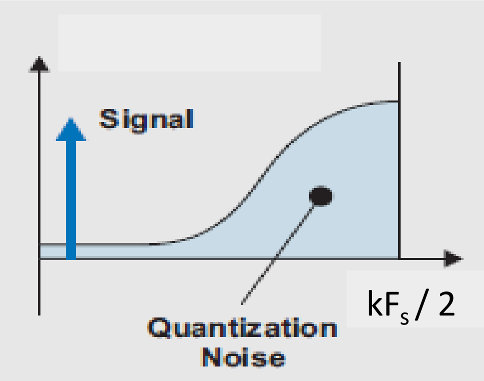
Oversampling increases effective word length

$k = 4 : B' = B + 1$
$k = 16 : B' = B + 2$
$k = 64 : B' = B + 3$

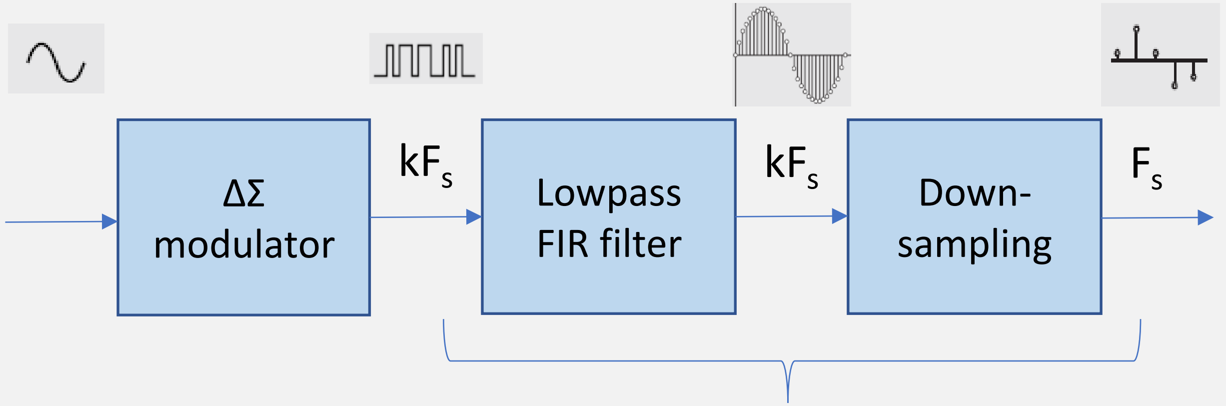
ΔΣ modulator based ADC

It is even possible to use a one-bit ADC with very high sampling frequency (**ΔΣ modulator**) and then use decimation with narrowband filtering to obtain signal samples with certain word length.

ΔΣ modulator does **shaping for the quantization noise**. It is not flat as in the previous images: quantization noise is pushed to higher frequencies.



So, oversampling implementation provides even more benefit as this part is suppressed in the following decimation stage.



Decimation phase, can be implemented using polyphase decomposition, noble identities & multirate filtering techniques

Summary of oversampling ADC

- High sampling rate & narrowband FIR filtering may be used to increase the effective number of bits of A/D conversion
- Extreme case is to use 1-bit A/D
- In this context, narrowband filtering may be implemented using multirate filtering techniques

i.e., something like this could follow ΔΣ modulator