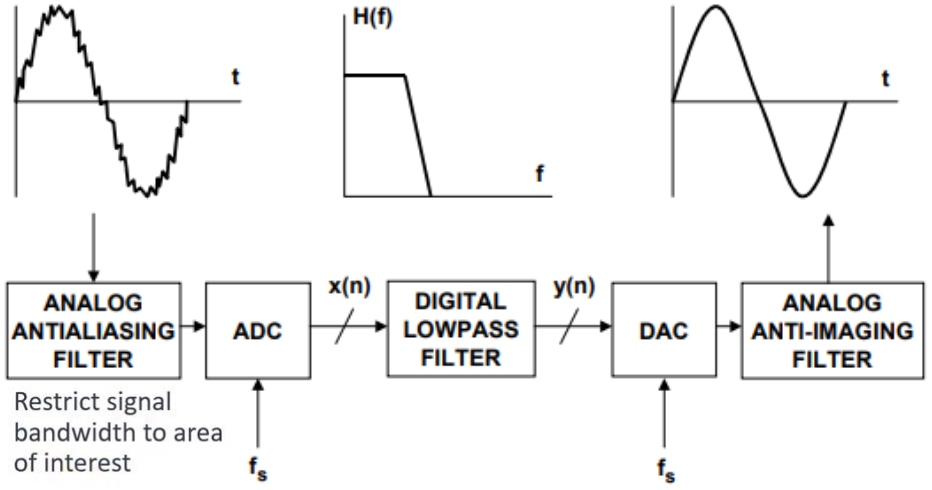
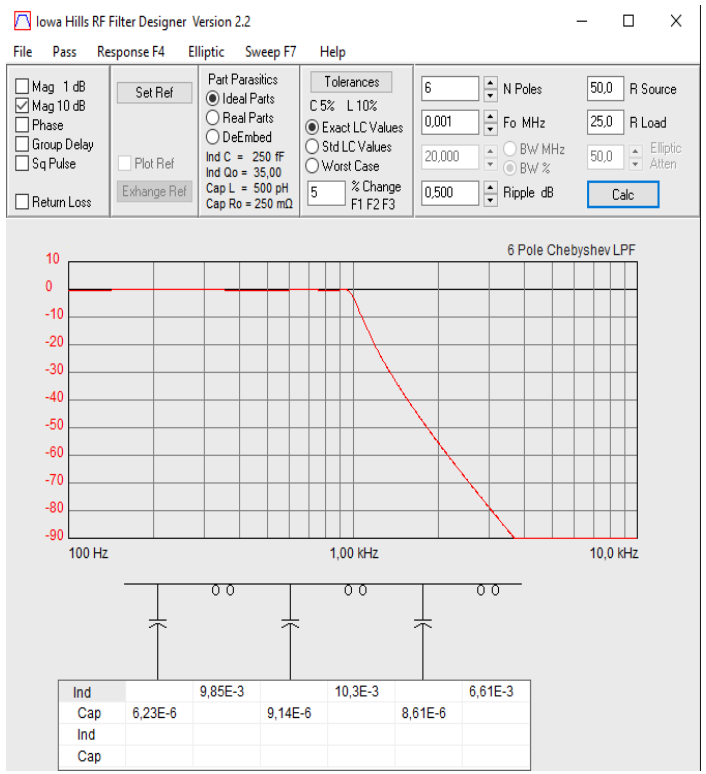
Digital Filters

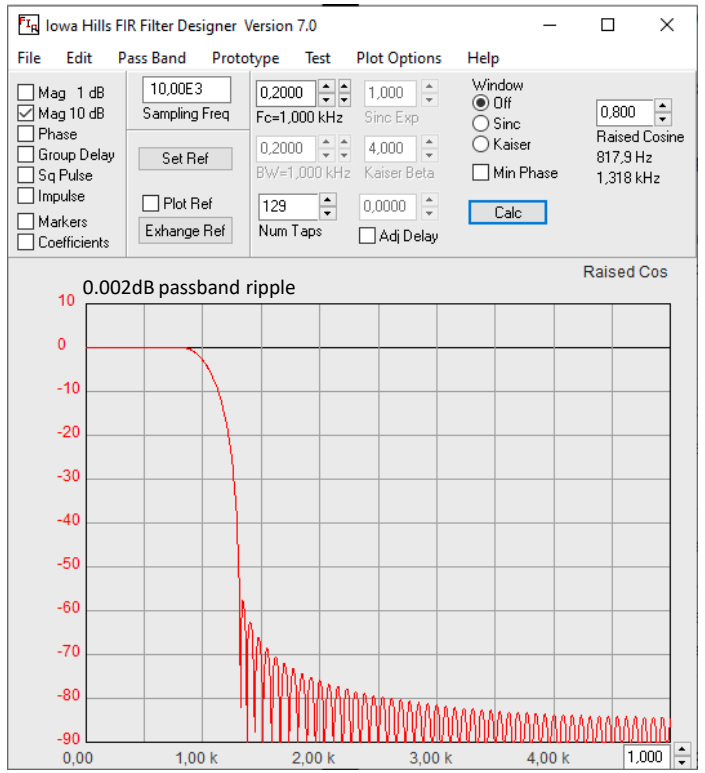


The sampling theorem essentially says that a signal has to be sampled at least with twice the frequency of the original signal. The sampling frequency required by the sampling theorem is called the *Nyquist frequency*.

An anti-aliasing filter (AAF) is a filter used before a signal sampler to restrict the bandwidth of a signal to satisfy the Nyquist–Shannon sampling theorem over the band of interest.



Need of 3 stages “sallen-key”

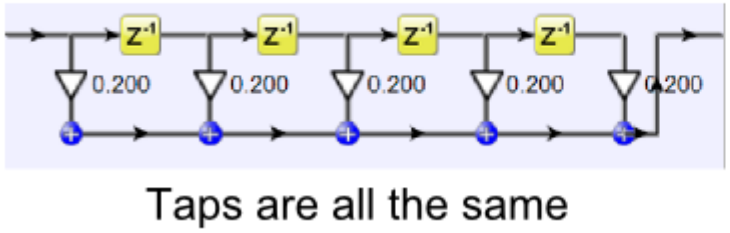


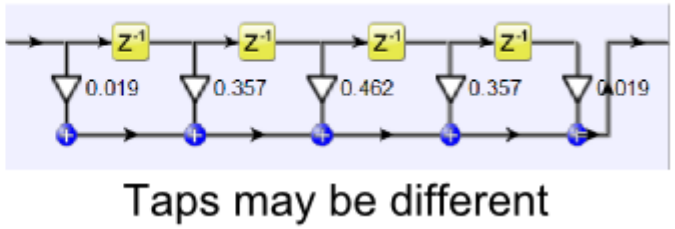
Probably not possible in analog design!
- low passband ripple
- sharp filter rolloff
- linear phase
- not dependent on component tolerances

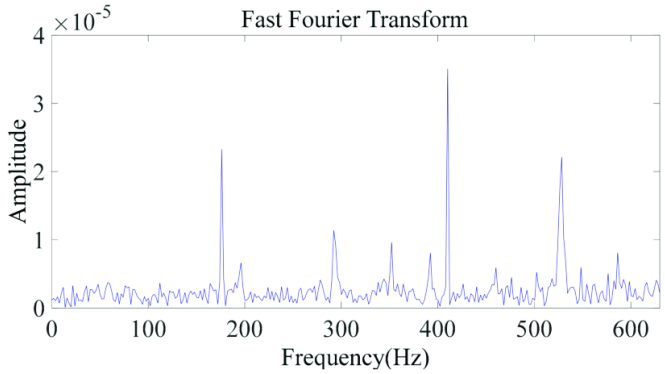
https://www.analog.com/media/en/training-seminars/design-handbooks/MixedSignal_Sect6.pdf

Scope of digital filters

Moving Average	FIR filter	IIR filter	FFT algorithm
Average of N input samples	Individualy weighted sum of N input samples	Individualy weighted sum of N input- and N output-samples	The discrete fourier transform (DFT) is obtained by decomposing a sequence of values into components of different frequencies

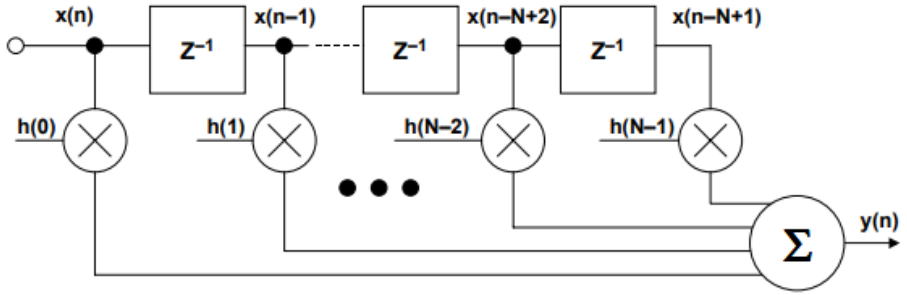






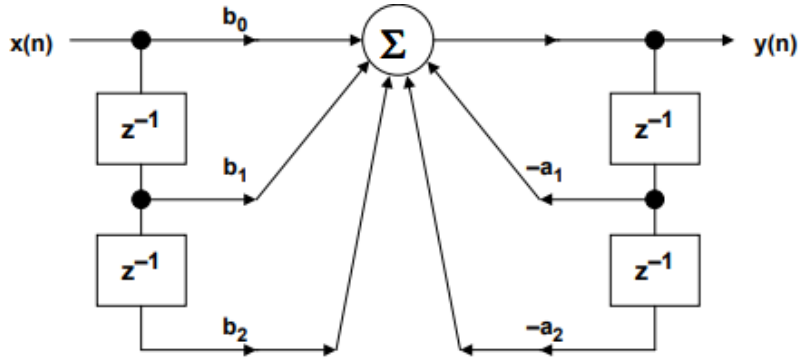
The two main types of digital filters

- Finite Impulse Response (FIR)
  - linear Phase
  - always stable
  - easy to Design
  - more computations



- $y(n) = h(n) * x(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k)$
- $*$ = Symbol for Convolution
- Requires N multiply-accumulates for each output

- Infinite Impulse Response (IIR)
  - nonlinear-phase response
  - Might have ripple in passband/stopband
  - potentially unstable
  - fewer computations



- $y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) - a_1y(n-1) - a_2y(n-2)$
- $y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k)$

FIR filter implementations typically require more multiplications and summations than IIR filters with similar filtering performance. However, because certain computer architectures are frequently better suited to performing FIR filtering, the computation speed of an IIR filter is not necessarily faster than an FIR filter.

https://www.physik.uzh.ch/local/teaching/SPI301/LV-2013-Help/lvanlsconcepts.chm/Comparing_FIR_andIIR_Filters.html

Cortex M4 with FPU

Cortex M4 core @120MHz IAR IDE compiler, no cache, no waitstate, >10’000 random arguments:

32-bit floating point **without** the FPU:	
op	exec. time (ns)
Addition	181 - 321
Substraction	181 - 321
Multiplication	259
Division	458 - 533

32-bit floating point **with** the FPU:	
op	exec. time (ns)
Addition	17
Substraction	17
Multiplication	17
Division	83

http://www.ganssle.com/tem/tem369.html

Computational effort needed

Signal Bandwidth = f_a
Sampling Frequency fs > $2f_a$
Sampling Period = 1 / f_s
Filter Computational Time + Overhead < Sampling Period

- Depends on Number of Taps
- Speed of DSP Multiplication-Accumulates (MACs)
- Efficiency of DSP

Example:
- Signal bandwidth f_a is 20kHz (audio)
- f_s needs to be > 2*20kHz; we choose 80kHz
- $1/f_s$ = 12.5µs
- FIR LPF with 129 taps; one multiplication takes CPU 4 cycles, one addition and on substraction: for each operation a budget of aprox. 0.02µs → f_{CPU} > 41MHz

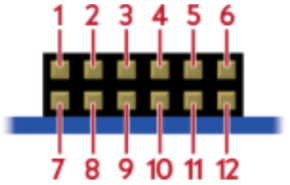
Implementing IIR 2nd order LP/HP filter, STM32F4 on Discoverykit

STM32F407VGT6 microcontroller featuring 32-bit Arm® Cortex®-M4 with FPU core
f_{CPU} max. = 168 MHz

- Cirrus CS5343 AD and CS4344 DA, stereo 24bit
- max 108kHz sampling rate
- I2S bus

Pmod I2S2 used with 24bit resolution @96kHz
I2S in full duplex mode

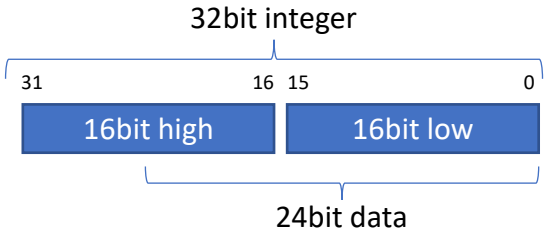
A/D SDOUT(PC2) data stream out A/D
MCLK(PC6) master clock for internal clocking ADC and DAC, 256*96kHz = 24.576MHz
SCLK (PB10) serial clock for clocking data bits, 2*32*96kHz = 6.144MHz
LRCK(PB12) left right channel select, 96kHz
D/A SDIN (PC3)



Pin 1	D/A MCLK
Pin 2	D/A LRCK
Pin 3	D/A SCLK
Pin 4	D/A SDIN
Pin 5	GND
Pin 6	VCC
Pin 7	A/D MCLK
Pin 8	A/D LRCK
Pin 9	A/D SCLK
Pin 10	A/D SDOUT
Pin 11	GND
Pin 12	VCC 3.3V

https://github.com/HB9GZE/Pub_STM32_Audio_DSP_5_3
credits to https://www.youtube.com/channel/UCwOkALY6oQbOL1zU7ApaPHg

Implementing IIR 2nd order LP/HP filter, STM32F4 on Discoverykit



Data representation:
The 24bit resolution gets represented on the I2S bus in a 32bit integer

```
int leftSample = (int) (receiveBuffer[0] << 16) | receiveBuffer[1];
```

Data transfer sequence:

left 16bit high
left 16bit low
right 16bit high
right 16bit low
left 16bit high
left 16bit low
right 16bit high
right 16bit low

```
uint16_t receiveBuffer[8];
uint16_t transmitBuffer[8];
```

DMA half complete int.

DMA complete int.

DMA Interrupt (non blocking):
Using DMA channel for data in/out transfer.

```
void HAL_I2SEx_TxRxHalfCpltCallback(I2S_HandleTypeDef *hi2s)
void HAL_I2SEx_TxRxCpltCallback(I2S_HandleTypeDef *hi2s)
```

IIR calculation / CMSIS library:

```
float outSampleF = l_b0 * inSampleF + l_b1 * l_in_z1 + l_b2 * l_in_z2 -
l_a1 * l_out_z1 - l_a2 * l_out_z2;
```

CMSIS: https://github.com/ARM-software/CMSIS/tree/master/CMSIS/DSP_Lib
arm_fir_f32.c or arm_iir_lattice_f32.c or fixedpoint arithmetik

Implementing IIR 2nd order LP/HP filter, STM32F4 on Discoverykit