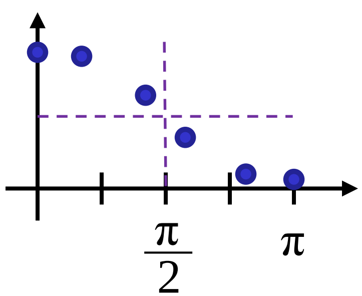
NYQUIST FILTERS

Generation of Nyquist Filters

- Use remez(•) in matlab but you must constrain the frequency points and amplitudes in certain ways
  - The **frequency vector** values must mirror each other in pairs around **π/2**
    - For example: `[0 0.2 0.4 0.6 0.8 1]`
      `[0 0.11 0.34 0.66 0.89 1]`
  - The **amplitude vector** values must mirror each other in pairs around a magnitude of **0.50**
    - For example:
```
[1 1 0 0]                      % low-pass
[0 0.05 0.10 0.90 0.95 1]      % high-pass
```
- Typically coefficients that should be zero will be close but not exactly zero when they are generated by remez(•)
  - Round these to make them exactly zero



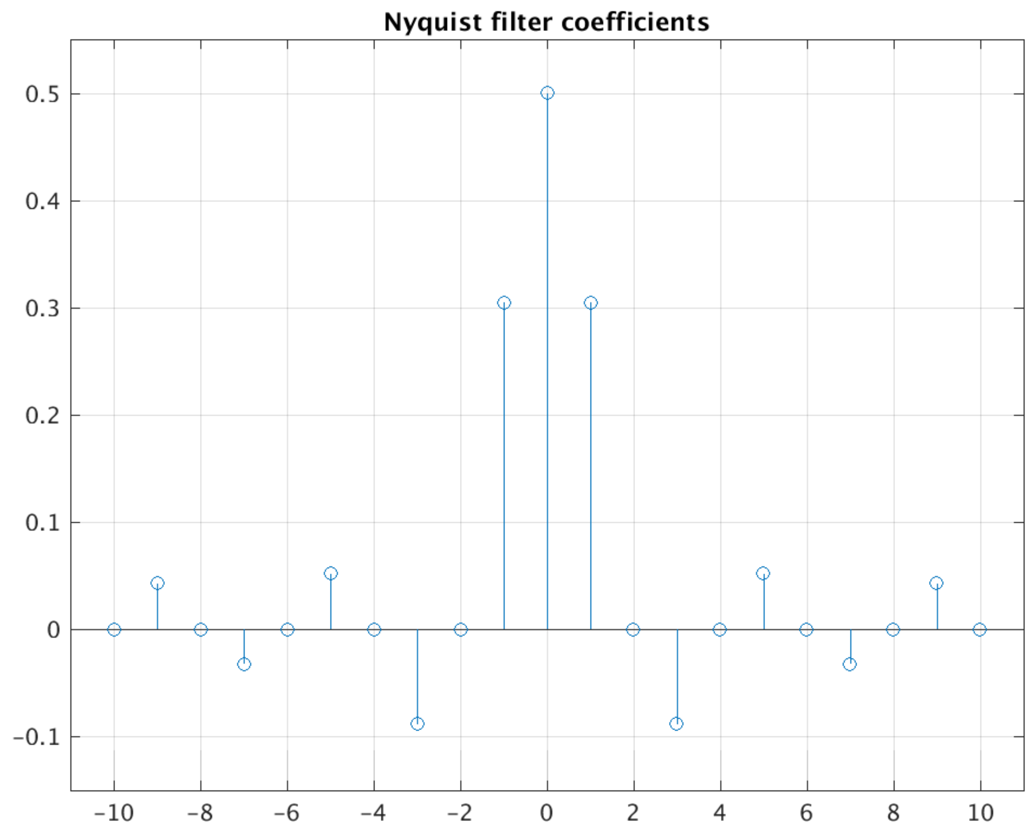
Nyquist Filter Example

- Copy and paste this program into a *.m file and experiment yourself in matlab!

```
% nyq.m
%
% 2015/03/04  Minor edits
% 2018/03/12  Added scaling of figure 2

% Set these
NumTaps = 21;
PrintOn = 1;

% Generate Nyquist filter coefficients
coeffs = remez(NumTaps-1, [0 0.45 0.55 1], [1 0.95 0.05 0]);

figure(1); clf;
stem(-10:10, coeffs);
axis ([-11 11 -0.15 0.55]);
title('Nyquist filter coefficients');
grid on;

if (PrintOn)  print -dtiff 1.tiff;  end

figure(2); clf;
freqz(coeffs);
title('Filter frequency response plotted by freqz(); Note -6dB at \pi/2');
subplot(2,1,1);      % select the top magnitude plot
axis([0 1 -45 5]);   % scale vertical axis more reasonably to see features
hold on;
plot(0.5, -6, 'ro');

if (PrintOn)  print -dtiff 2.tiff;  end

% Generate white-noise flat-spectrum signal
in = rand(1, 100000) - 0.5;

figure(3); clf;
psd(in);
axis([0 1 -18 -4]);
title('White-noise input signal to characterize filter; 100,000 samples');

if (PrintOn)  print -dtiff 3.tiff;  end

% Pass the white-noise signal through the filter
out = conv(coeffs, in);

figure(4); clf;
psd(out);
title('Filter frequency response plotted by psd(); 100,000 samples; note -6dB (-17dB) at \pi/2');
axis([0 1 -50 -5]);   % scale vertical axis more reasonably to see features
hold on;
plot(0.5, -17, 'ro');

if (PrintOn)  print -dtiff 4.tiff;  end
```

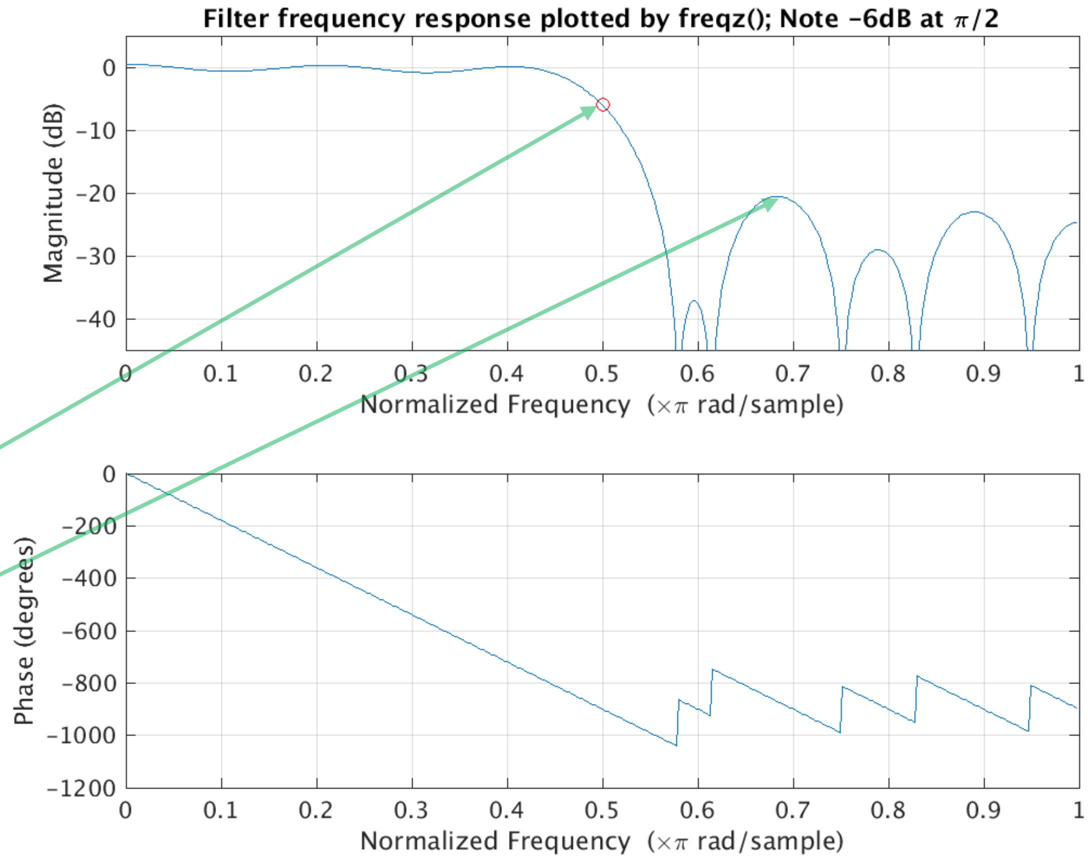
Nyquist Filter Coefficients Impulse Response

- 21-tap example
- It has significantly reduced hardware with almost half of its coeffs == zero
  - (N–1)/2 taps equal to zero for $N = 4k+1$
  - (N–3)/2 taps equal to zero for $N = 4k+3$



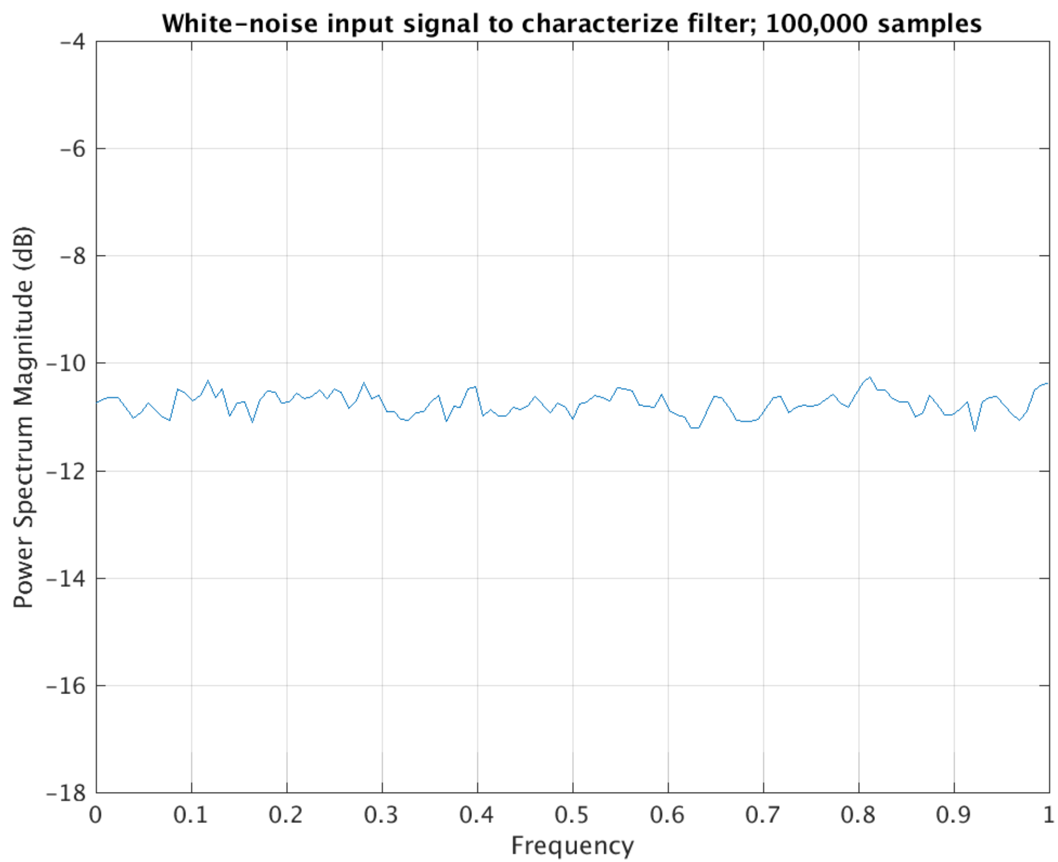
Filter Example

- The filter's frequency response plot made by `freqz(•)`
- Note these critical points to make a comparison later
  - –6 dB at π/2 (1/2 magnitude)
  - –20 dB at 0.68 π



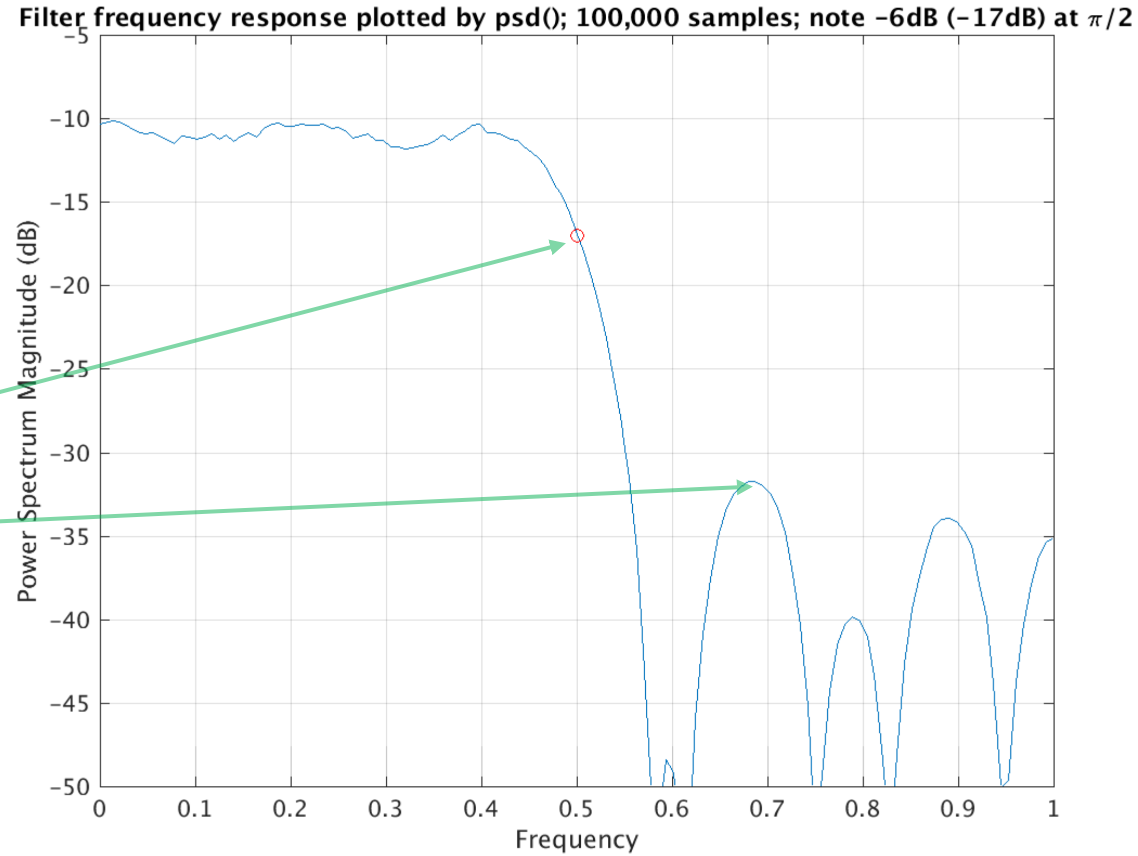
The Second Less-Accurate Method to Measure Filter Response

- "White noise" random signals have a (nearly) flat spectrum
- This example contains 100,000 samples
  - More samples will make the spectrum flatter
- We will pass this signal through our filter and view the output spectrum to gauge the filter's frequency response



The Second Less-Accurate Method to Measure Filter Response

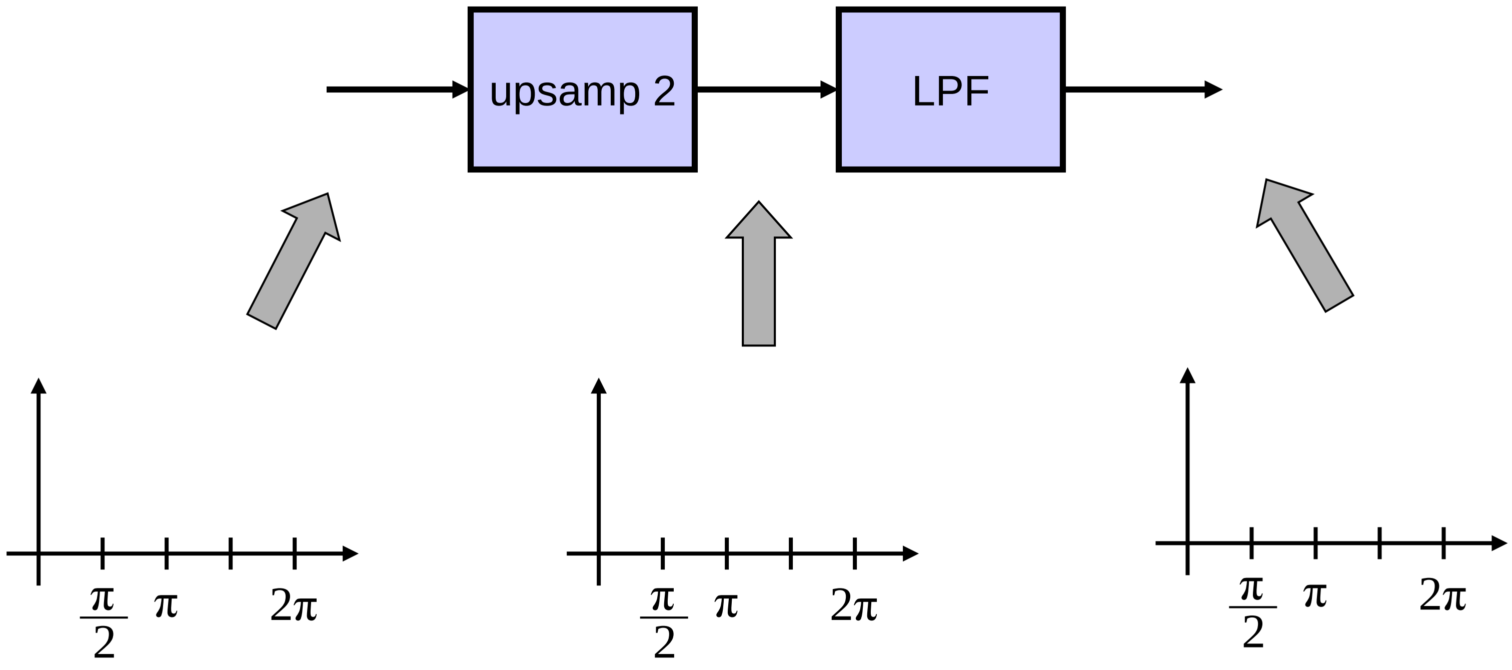
- This is the spectrum of the white-noise signal after being passed through the filter
- Note approximate values
  - –6 dB at π/2
  - –20 dB at 0.68 π
  - It matches freqz()!
- Recall that this method is best for actual bit-accurate HW designs



NYQUIST &
UPSAMPLING FILTERS

Nyquist Filters and Upsampled Signals

- First recall standard approach
  - Low-pass filter has cutoff frequency at π/2

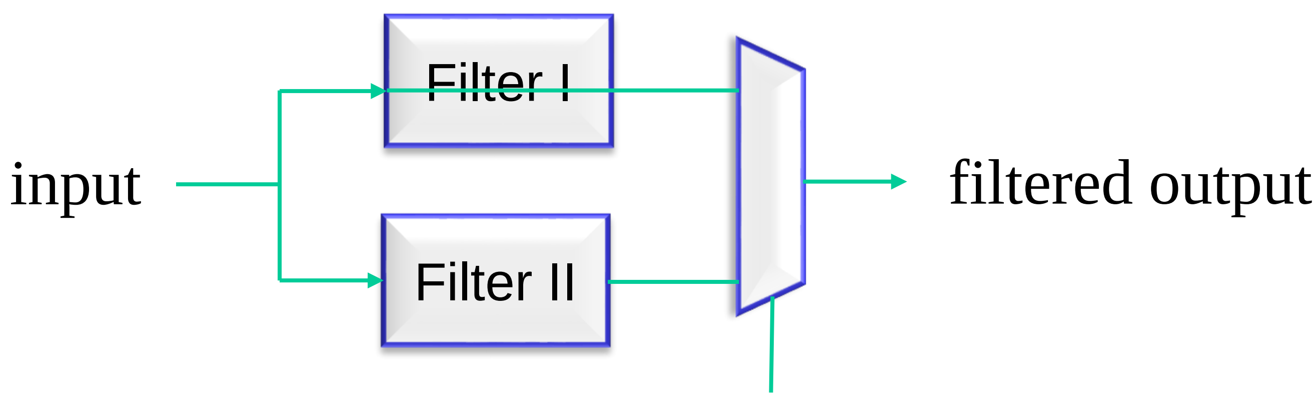


Nyquist Filters and Upsampled Signals

- Nyquist filters have almost half of their coefficients equal to zero
- Upsampled signals have every other sample equal to zero
- Lots of zeros ⇨ an opportunity!
- There are two alignments of data and filter coefficients
  1. The center tap of the filter aligns with a non-zero value in the upsampled data stream
     - The result is a trivial single multiply
     - With clever scaling, the multiplier can be reduced to a power-of-2 shift requiring no hardware whatsoever
  2. The other alignment
     - The result is a simplified filter with almost half the hardware because (N-1)/2 of the FIR multiplications are zero times zero
     - N delay registers are still needed however

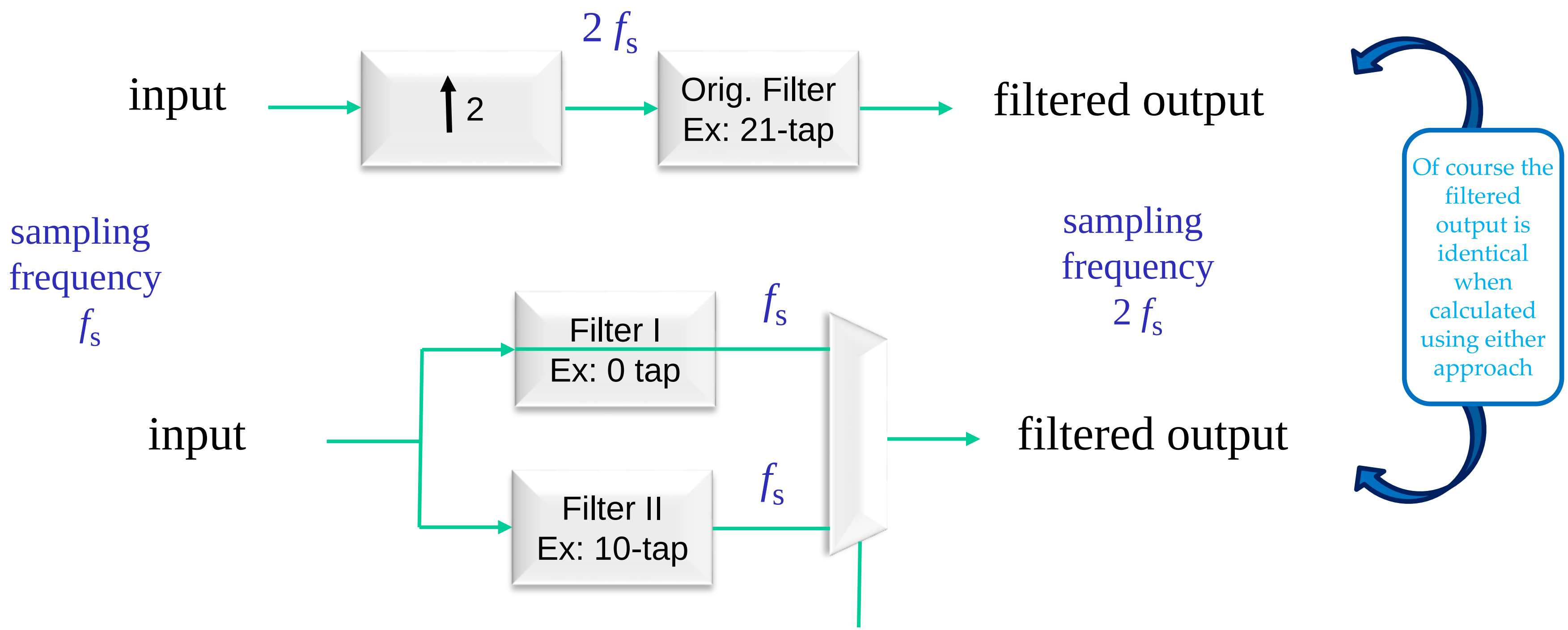
Nyquist Filters and Upsampled Signals

- The Nyquist filter can then be implemented very efficiently by dividing the filter into two components that compute the two alignments and reconstructing the output with a 2:1 mux which interleaves samples taken from each filter every other cycle



Nyquist Filters and Upsampled Signals

- Another great benefit: the two filters are operating in the slower pre-upsampled sampling frequency domain
- Upsampling is performed in the mux



Nyquist Filters and Upsampled Signals

- Recall that for common static CMOS circuits,
  $Power = C\,V^2 f$
- In summary, the optimized merged upsampler/filter using a Nyquist filter yields:
  - Approximately half the total hardware
    → $C' = C/2$
  - Filters operating at half the clock frequency
    → $f' = f/2$
  - Only a 2:1 mux operates at the faster upsampled clock frequency
  - *Power′* approximately 1/4 of the original power; probably a little less due to relaxed timing requirements of the simplified filter