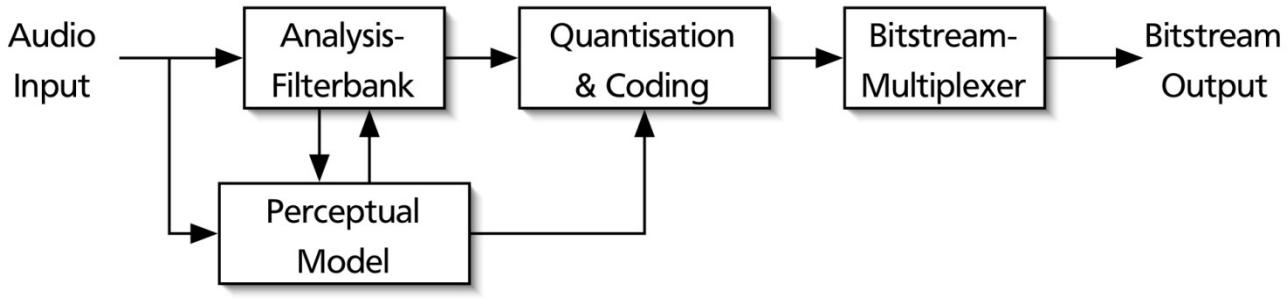
# Filter Banks I

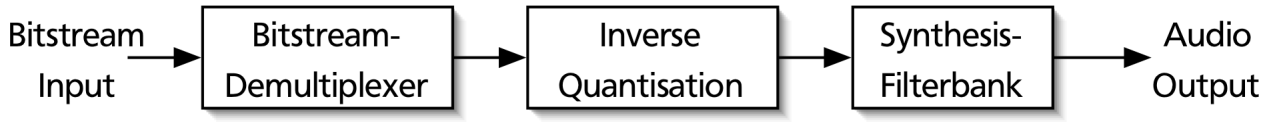
Prof. Dr. Gerald Schuller

Fraunhofer IDMT & Ilmenau University of Technology
Ilmenau, Germany

# Structure of perceptual Audio Coders




Encoder




Decoder

# Filter Banks

- essential element of most audio coders
- transform from time to frequency domain and vice versa

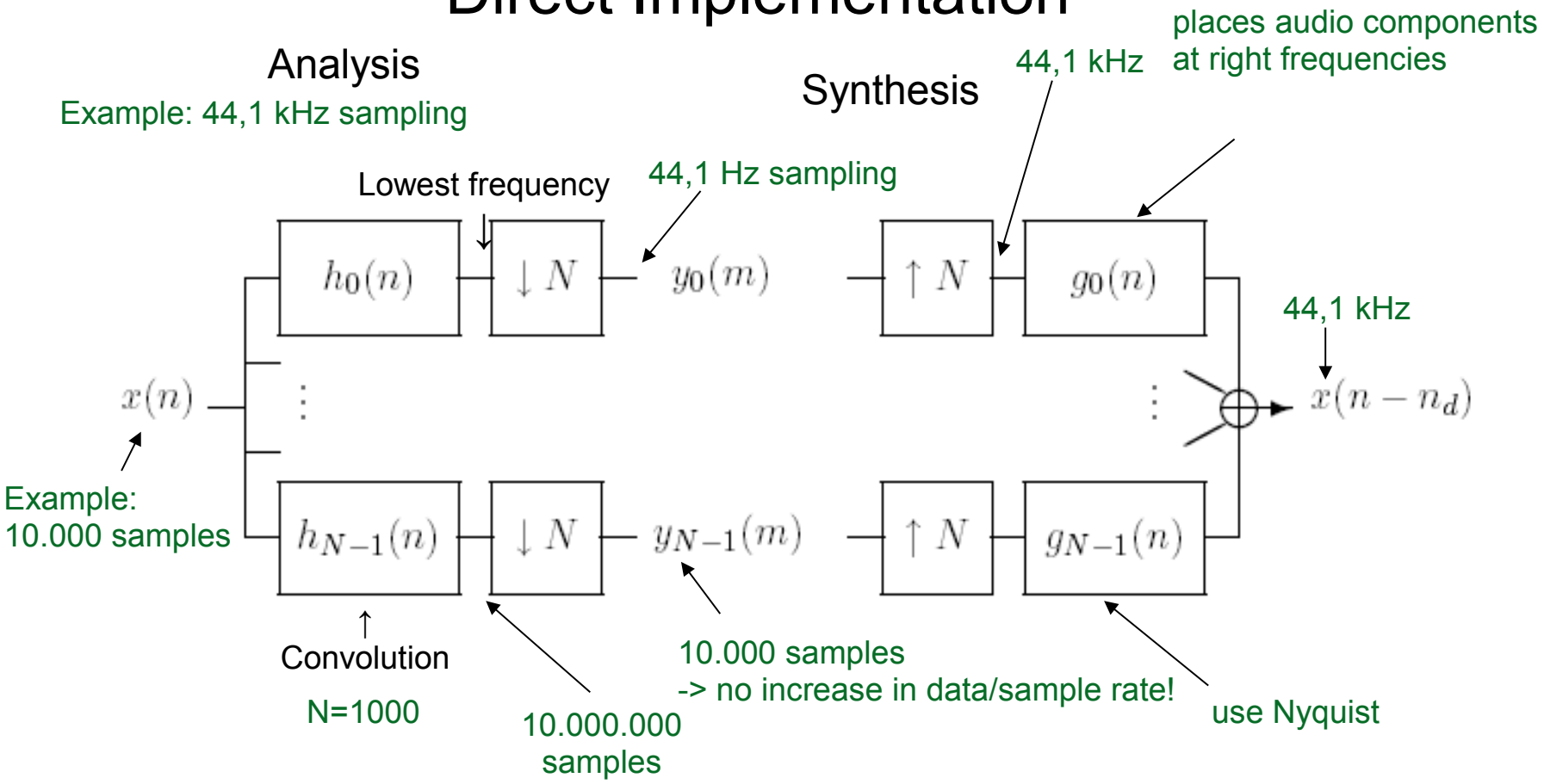
  - Goal:
    - Good filter bank
    - Compress audio signals
  - Approach:
    - Redundancy Reduction
    - Irrelevance Reduction

# Filtering

- Remember, a digital (bandpass) filter can be represented by the **convolution** of the audio signal x(n) with the impulse response of the filter h(n). The output y(n) is then obtained by

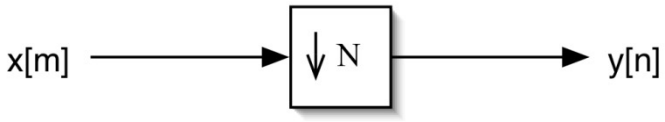
$$y(n) = x(n) * h(n) = \sum_{n'} x(n-n')\,h(n')$$

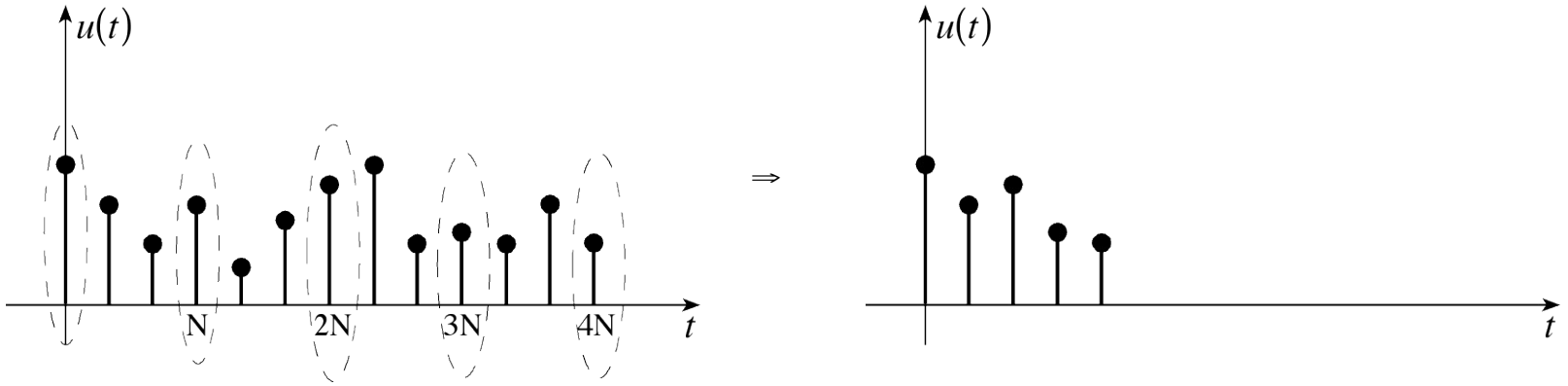
# Critically sampled Analysis and Synthesis Filter Bank, Direct Implementation



# Down-Sampling

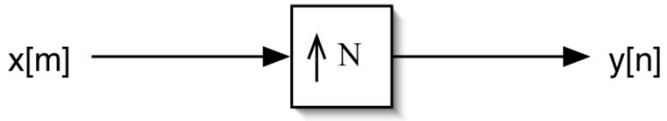
- The operation of “down-sampling“ by factor N describes the process of keeping every Nth sample discarding the rest

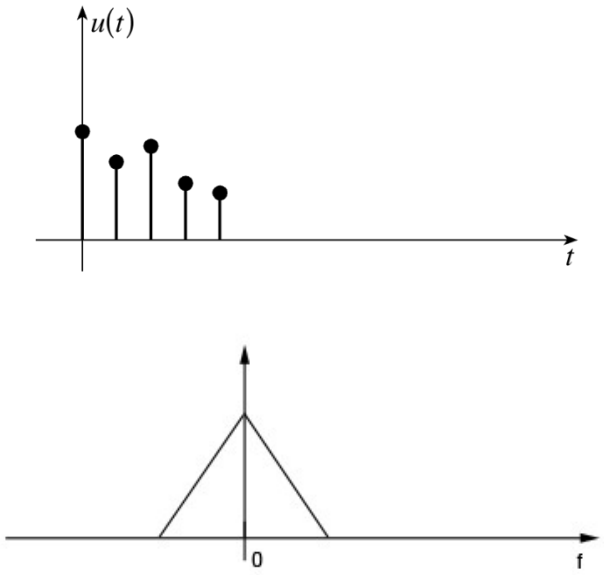


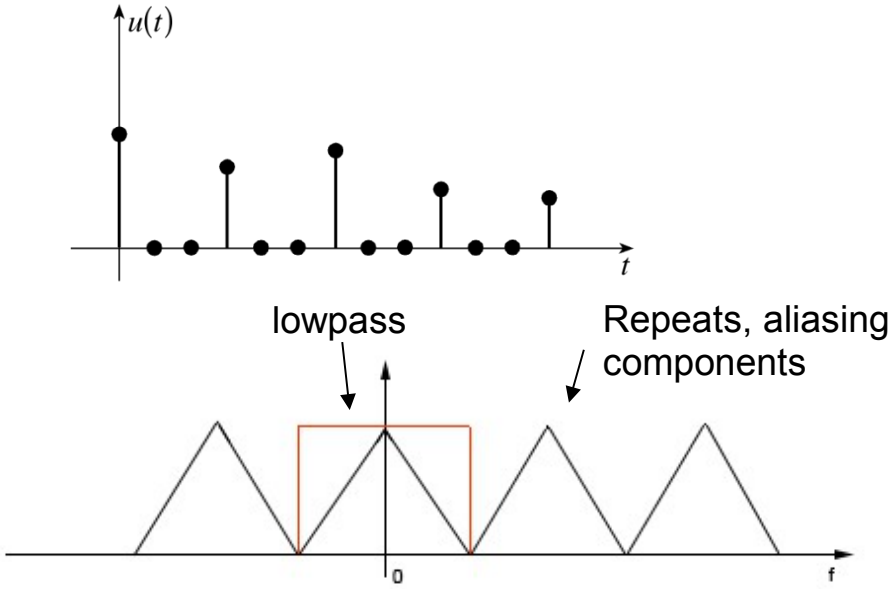
# Up-Sampling

- The operation of “up-sampling“ by factor N
  describes the insertion of N-1 zeros between every sample of the input




⇒

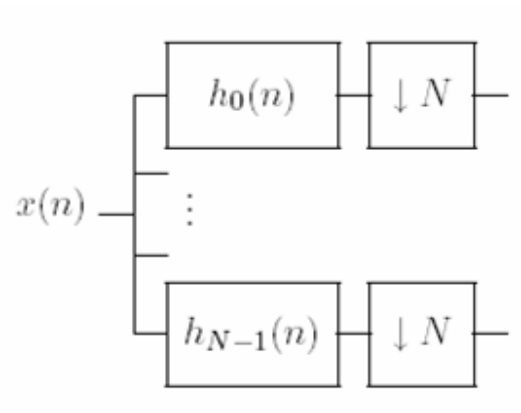


# Filter Bank Structure - The Analysis Filter Bank Direct Implementation

Example:

- N=1024 filters (power of 2 for efficient FFT impl.)
- $f_s$=44100Hz sampling frequency
- $f_g$=22050Hz Nyquist frequency

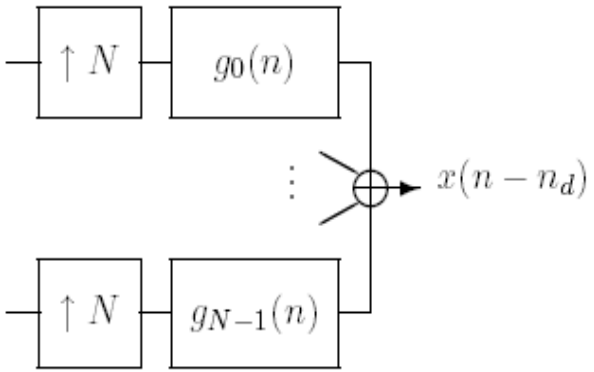
$$\frac{f_g}{N} = \frac{f_s / 2}{N} = 21.5Hz$$




If we have N filters, and no down-samplers, then we would have N*$f_s$ samples per second after filtering – more than input!

- hence down-samplers.
- with down-samplers: number of samples stays constant, **downsampling factor = number of subbands**
- This means "**critical sampling**"

# Filter Bank Structure - The Synthesis Filter Bank Direct Implementation




- Up-sample each subband by N to restore original sampling rate
- Apply passband filter to each subband signal
- Add each subband signal to generate output signal

# Filter Bank Structure - **Direct Implementation** Python Example

Implement 1 branch, subband k=1, of the analysis and synthesis filter bank with N=16 subbands with 32kHz sampling rate (hence the passband is between 1 kHz and 2 kHz), in **direct implementation**.
Start with designing a bandpass filter using the scipy.signal.remez function, which is an “equi-ripple” FIR filter design function:

```
ipython –pylab
import scipy.signal as signal
N=16
b=signal.remez(8*N,[0,500,1000,2000,2500,16000],[0,1,0],[100,1,100],Hz=32000,
type='bandpass')
#Check the design:
plot(b)
title('Filter Impulse Response')
xlabel('Time in Samples')
w,H=signal.freqz(b)
plot(w,20*log10(abs(H)+1e-6))
title('Filter Magnitude Frequency Response')
xlabel('Normalized Frequency')
ylabel('dB')
```

# Analysis Filter Bank Structure - **Direct Implementation,** Python Example

Now the **analysis filtering and down sampling**:

```
import sound as snd
[s,rate]=snd.wavread('sndfile.wav')
print("length of sound in samples: ", len(s))
plot(s)
title('Original Signal')
#Filter implementation:
filtered=signal.lfilter(b,1,s)
print("length of filtered sound in samples: ", len(filtered))
plot(filtered)
#play filtered sound:
snd.sound(filtered, 32000)
#Now Down-sampling with factor N:
N=16
filteredds=filtered[::N]
plot(filteredds)
#Listen to it at 1/N'th sampling rate:
snd.sound(filteredds, 2000)
```

# Synthesis Filter Bank Structure - **Direct Implementation,** Python Example

Now the **up-sampling and synthesis filtering**:

```
#Up-sampling:
filteredus=np.zeros(len(filteredds)*N)
filteredus[::N]=filteredds
#Listen to the up-sampled sound:
snd.sound(filteredus, 32000)
#Synthesis Filtering:
#Bandpass Synthesis Filter implementation to attenuate the spectral copies:
filteredsyn=signal.lfilter(b,1,filteredus)
plt.plot(filteredsyn)
plt.title('Up-sampled and Filtered Signal')
plt.xlabel('Time in Samples')
plt.ylabel('Sample Values')
plt.show()
snd.sound(filteredsyn, 32000)
```

# Filter Bank Structure - **Direct Implementation** Python Example

This can be executed from our python file as:

```
Python 1branchFBdirectImpl.py
```

**Observe**: After the synthesis filtering the signal again sounds like after the analysis filtering, even though we had down-sampling and up-sampling in between. This means we did not loose much information after down-sampling!

# Definition: Perfect Reconstruction

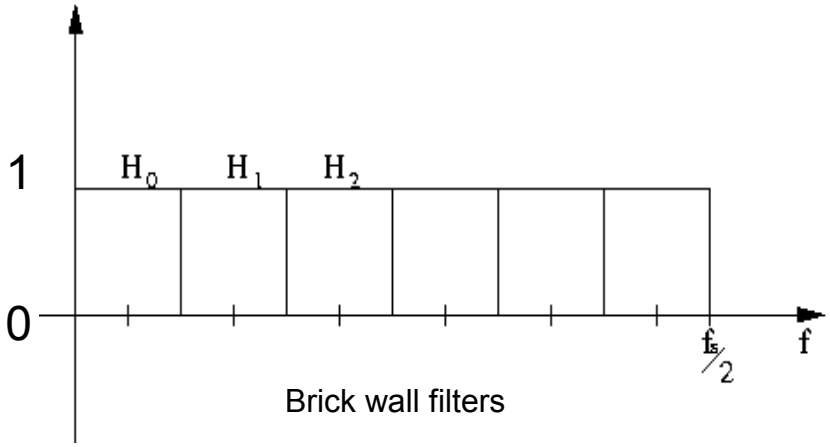
- The property of the output signal out of cascaded analysis and synthesis filter bank being identical to the input signal (except for a time shift $n_d$) is called "**Perfect Reconstruction**" (PR):

$$\text{output} = x\left(n - n_d\right)$$

- A filter bank having this property is called a "Perfect Reconstruction Filter Bank".

# Filter Bank Structure – Perfect Reconstruction

Example PR filter bank: DFT




Brick wall filters

Thought experiment: ideal `brick wall filters`

Brick wall: magnitude in passband is one, otherwise zero

Nyquist Theorem: we can down-sample the subband signals by factor N without loss of information

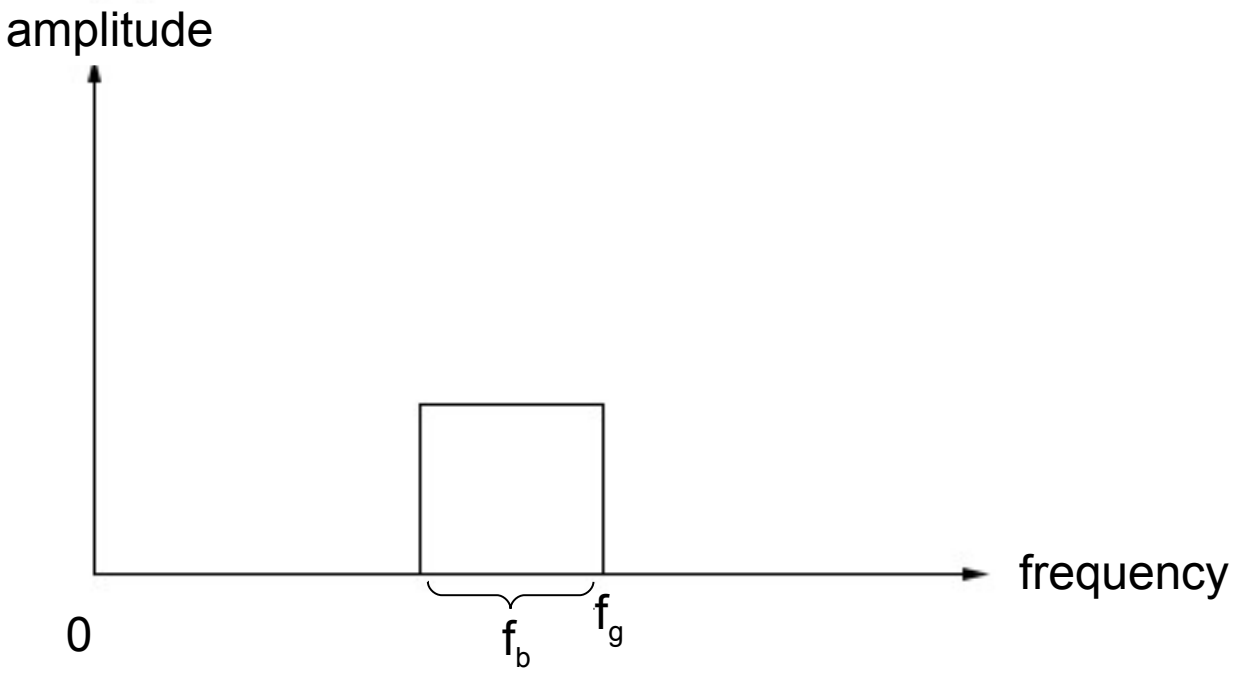
With suitable brick wall synthesis filters, perfect reconstruction (input = output) could be achieved

# Bandpass Nyquist

Goal:

- Keep critical downsampling (downsampling rate N is equal to number of subbands).
- -> No increase in number of samples.
- Still want to obtain perfect reconstruction!
- -> Ideally aliasing cancels!

# Example Bandpass Signal



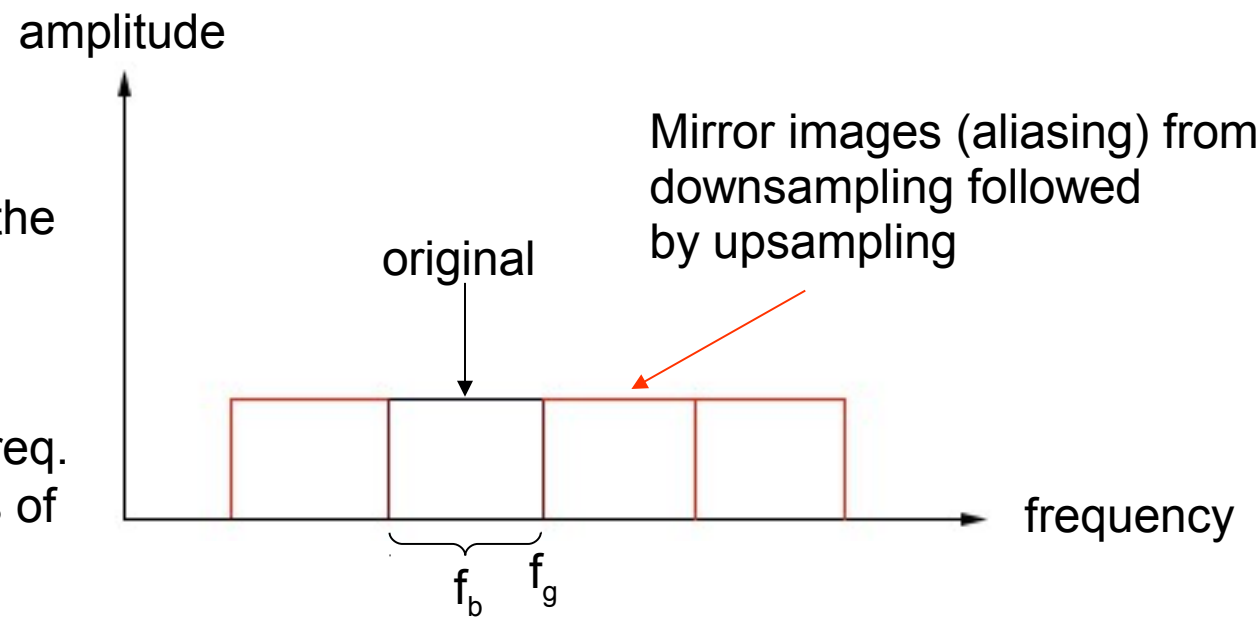

Bandpass Nyquist: sampling with twice the Bandwidth ($f_b$)!



Prof. Dr.-Ing. K. Brandenburg, bdg@idmt.fraunhofer.de Prof. Dr.-Ing. G. Schuller, shl@idmt.fraunhofer.de

# After Downsampling and Upsampling:

Bandpass Nyquist:
Sampling at least twice the bandwidth f_b enables the reconstruction of the bandwidth limited signal (if the lower end of the freq. band is multiple integers of the bandwidth).




Reconstruction: apply ideal bandpass filter for original frequency range ("fish out" original), no overlap with aliasing.

Problem: ideal bandpass filters are not realizable!

# Ideal Filters

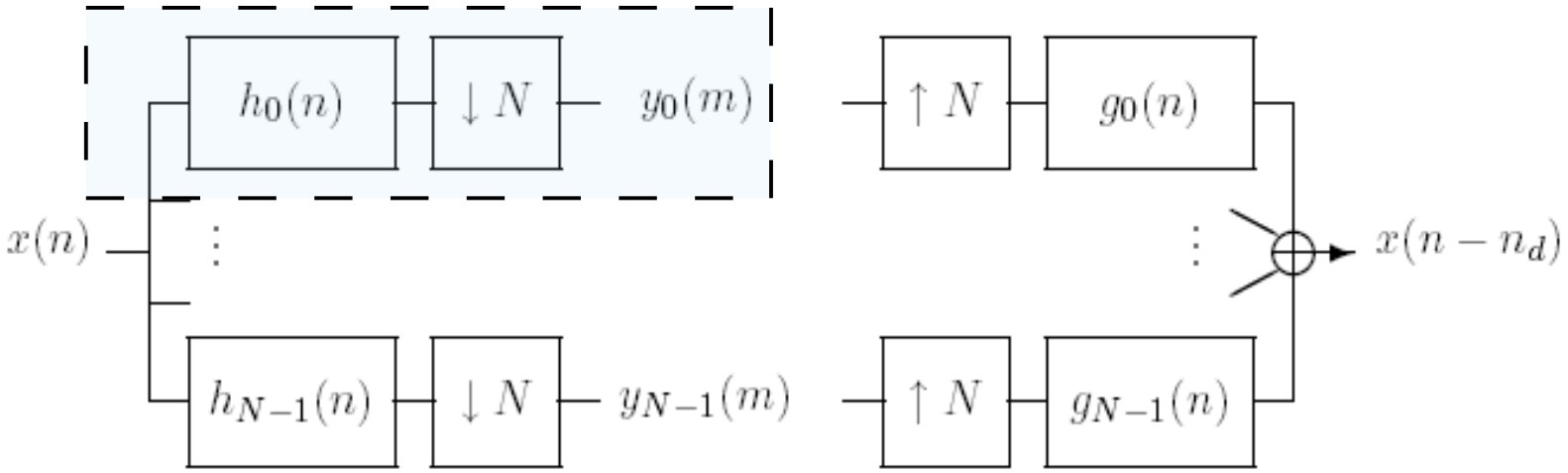
- Ideal filters are not realizable
- In the time domain they would mean a convolution of our signal with a Sinc function
- Sinc function is infinitely long and not causal, meaning it causes infinite delay
- We can not simply use a DFT or FFT to obtain an ideal filter in the frequency domain either
- Because the DFT also represents a filter bank, but a special type
- Its equivalent filters are far from perfect filters (hence we cannot make ideal filters with it), not good enough for our purposes (audio coding and the ear), as we will see
- Don‘t use your eye (looking at waveforms) to guess what the ear might be hearing (quite different processing)

# Basic Principle: z-Transformation (1)

- Goal: Realizable FB with critical sampling and perfect reconstruction (PR)
- Problems with ideal filter banks:
  - Brick wall filters not realizable (infinite delay!)
- Approach:
  - Find a suitable mathematical description for realizable Perfect Reconstruction Filter Banks

# Analysis Side: Use "**Noble Identities**" to Exchange Filtering and Downsampling of Each Subband

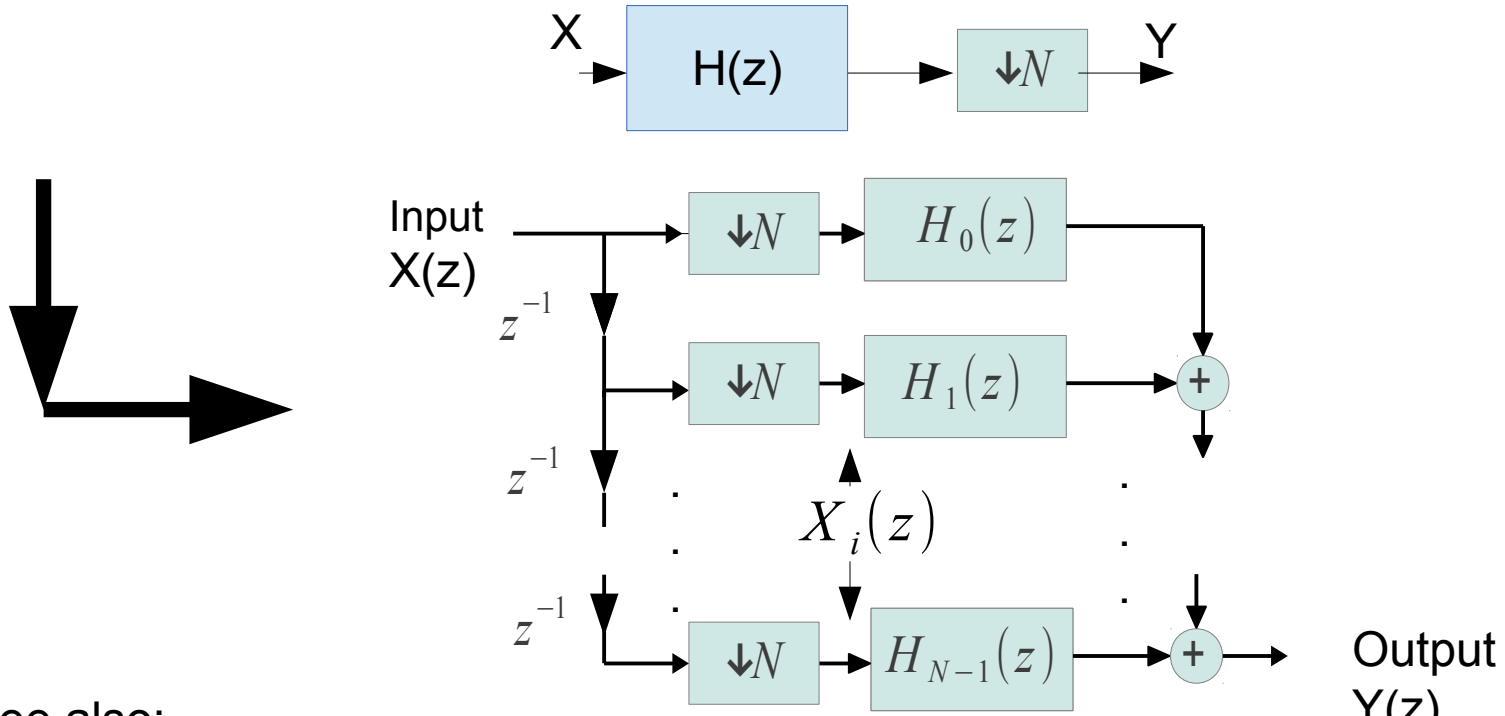
Take one subband:




Analysis Filter Bank

Synthessis Filter Bank

# Use “**Noble Identities**” to Exchange Filtering and Downsampling of Each Subband




See also:
http://www.tu-ilmenau.de/fileadmin/public/it_dsv/ADSP/Lecture_WS12_13/08_12-12-03DSP2_NobleIdentitiesFilters.pdf



Prof. Dr.-Ing. K. Brandenburg, bdg@idmt.fraunhofer.de Prof. Dr.-Ing. G. Schuller, shr@idmt.fraunhofer.de

# Noble Identities, Polyphase Vectors

The left hand side with the downsamplers can be seen as a serial to parallel converter into blocks of length N. We obtain blocks or vectors of length N for the signal x and the filter H, each containing the "polyphase" components.
The z-Transform vector of the polyphase components of input x:

$$\underline{X}(z)=[X_0(z),\ldots,X_{N-1}(z)]$$

The z-Transform vector of the polyphase components of the filter:

$$\underline{H_k}(z)=[H_{N-1,k}(z),\ldots,H_{0,k}(z)]$$

With

$$X_n(z)=\sum_{m=0}^{\infty} x(mN+n)\cdot z^{-m} \qquad H_{n,k}(z)=\sum_{m=0}^{\infty} h_k(mN+n)\cdot z^{-m}$$

$$n=0,\ldots,N-1$$

# Noble Identities, Polyphase Vectors

The last slide shows a representation as vector of polynomials. Observe that a polyphase vector like

$$\underline{X}(z) = [X_0(z), \ldots, X_{N-1}(z)]$$

can alternatively also be written as a polynomial of vectors,

$$\underline{X}(z) = \sum_{m=0}^{\infty} [x(mN), x(mN+1), \ldots, x(nM+N-1)] \cdot z^{-m}$$

We see that the vectors in the sum are the blocks of length N of our audio signal. The sum takes all blocks of length N of our audio signal and turns them into this polyphase polynomial.

# Noble Identities, Polyphase Vectors

The filtering and downsampling then becomes:

$$\underline{X}(z)\cdot\underline{H_k^T}(z)=Y_k(z)$$

Since we have not just 1 filter, but N filters, we can collect the N filter polyphase vectors of size N into a “polyphase matrix” of size NxN!. This then produces a polyphase vector of size N for the N resulting filter output or subbands:

$$\underline{Y}(z)=[Y_0(z),\ldots,Y_{N-1}(z)]$$

# Polyphase Description (5)

Arrange the N impulse response vectors $H_k(z)$ of length N into a *NxN* square matrix (can be invertible!):

$$\underline{\underline{H}}(z) = \left[\underline{H}_0(z), \underline{H}_1(z), \ldots, \underline{H}_{N-1}(z)\right]$$

$$\underline{Y}(z) = \left[Y_0(z), Y_1(z), \cdots, Y_{N-1}(z)\right]$$

subbands

# Polyphase Description, Analysis (1)

- Hence the form of the polyphase matrix for analysis is (Type 1 polyphase):

$$\underline{\underline{H}}(z) = \begin{bmatrix} H_{N-1,0}(z) & H_{N-1,1}(z) & \cdots \\ H_{N-2,0}(z) & \ddots & \\ \vdots & & \\ H_{0,0}(z) & & H_{0,N-1}(z) \end{bmatrix}$$

phase subband

- and each subband filter can hence be written as

$$H_k(z) = \sum_{n=0}^{N-1} z^{-n} H_{n,k}(z^N)$$

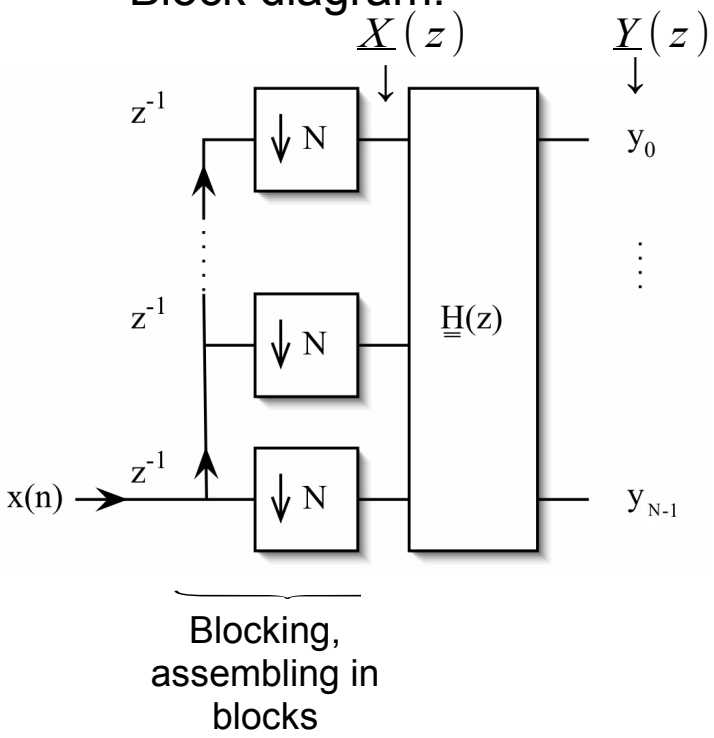
phase subband



Prof. Dr.-Ing. K. Brandenburg, bdg@idmt.fraunhofer.de Prof. Dr.-Ing. G. Schuller, shl@idmt.fraunhofer.de

# Polyphase Description, Analysis (2)

Block diagram:




$\underline{X}(z)$: vector

$\underline{\underline{H}}(z)$: matrix

Final equation of analysis filter bank:

$$\underline{Y}(z) = \underline{X}(z) \cdot \underline{\underline{H}}(z)$$

$\underline{\underline{H}}(z)$: Analysis Polyphase Matrix, *NxN*

$\underline{X}(z)$: Vector of polynomials, contains input samples

Mathematically very simple operation for entire filter bank including down sampling.

Observe that a multiplication with $z^{-1}$ can be interpreted as a **delay** of the signal by **1 sample**. It can be implemented as a **delay** or **memory** element.

Prof. Dr.-Ing. K. Brandenburg, bdg@idmt.fraunhofer.de Prof. Dr.-Ing. G. Schuller, shl@idmt.fraunhofer.de

# Polyphase Description,Example

Assume a signal x=[5,6,7,8,9,10] and N=3. Then we get the signal blocks **x**(m) with m in a range as we needed to fit the signal, as

**x**(0)=[5,6,7]

**x**(1)=[8,9,10]

The polyphase elements $X_n(z)$ with phase n=0...,N-1 are

$$X_0(z)=5+8\cdot z^{-1} \quad X_1(z)=6+9\cdot z^{-1} \quad X_2(z)=7+10\cdot z^{-1}$$

Or written as polynomial of blocks,

$$X(z)=\sum_{m=0}^{1}[5,6,7]\cdot z^{0}+[8,9,10]\cdot z^{-1}$$

The polyphase vector is $X(z)=[X_0(z), X_1(z), X_2(z)]=[5+8\cdot z^{-1}, 6+9\cdot z^{-1}, 7+10\cdot z^{-1}]$

Assume we have the first analysis impulse response of h0 =[3,4,5,6,7,8] for N=3. Then its polyphase vector is in general

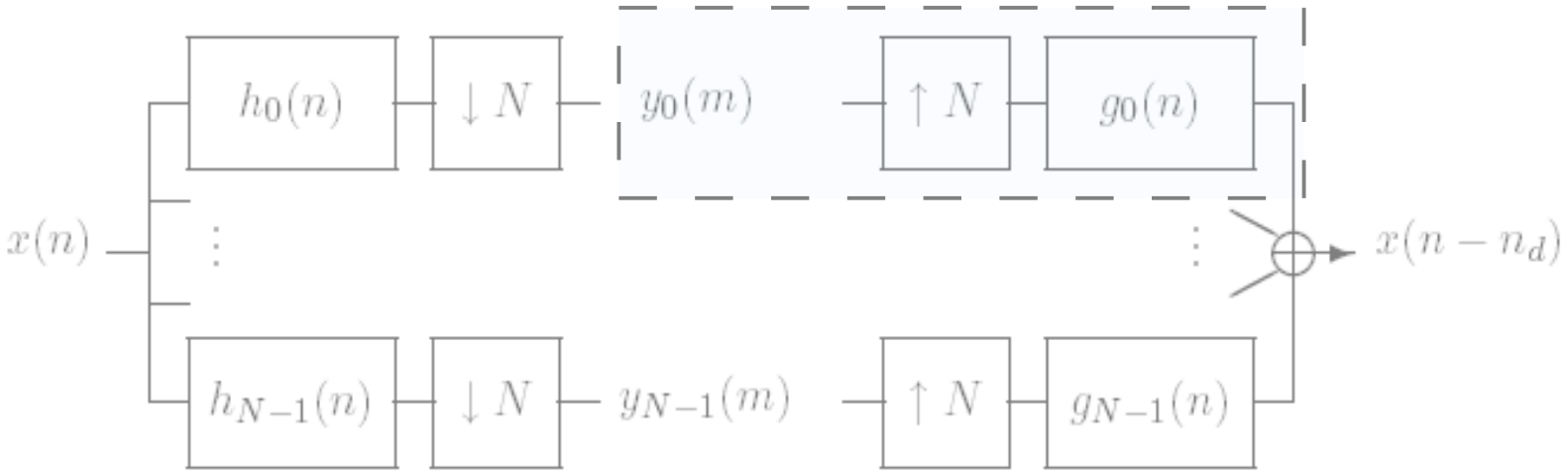
$$H_k(z):=[H_{N-1,k}(z), H_{N-2,k}(z), \ldots, H_{0,k}(z)]$$

(with our phases going down) and for this example,

$$H_0(z)=[5+8\cdot z^{-1}, 4+7\cdot z^{-1}, 3+6\cdot z^{-1}]$$

# Synthesis Side: Use "**Noble Identities**" to Exchange Filtering and Upsampling of Each Subband

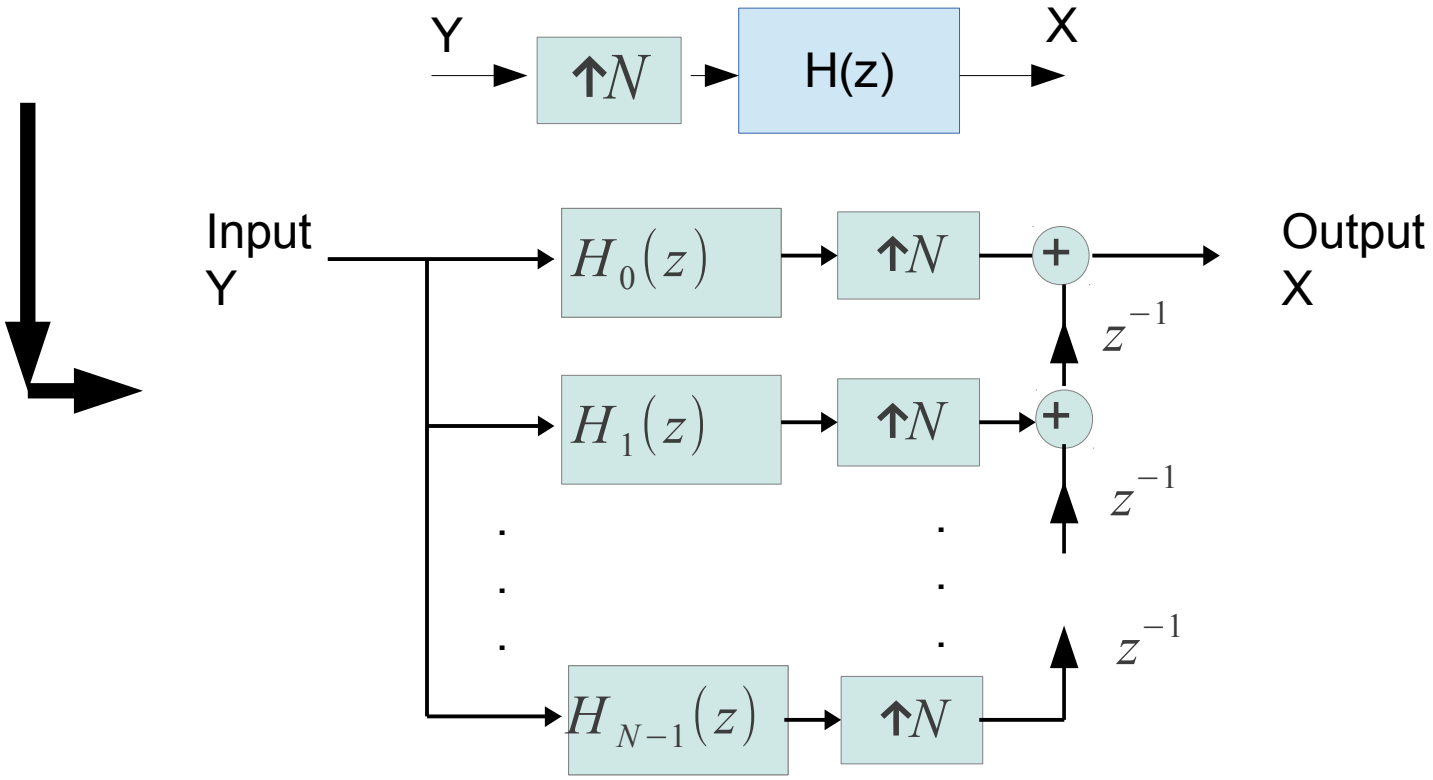
Take one subband:




Analysis Filter Bank

Synthesis Filter Bank

# Synthesis: Use "**Noble Identities**" to Exchange Filtering and Upsampling of Each Subband:



# Polyphase Description, Synthesis (1)

- The polyphase matrix for synthesis is (Type 2 polyphase):

$$\underline{\underline{G(z)}} = \begin{bmatrix} G_{0,0}(z) & G_{0,1}(z) & \cdots \\ G_{1,0}(z) & \ddots & \\ \vdots & & \\ G_{N-1,0}(z) & & G_{N-1,N-1}(z) \end{bmatrix}$$

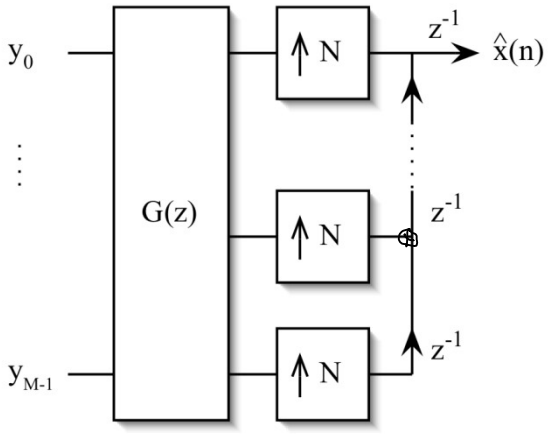
- Now each filter has its polyphase components along the rows, and each subband filter can be written as

$$G_k(z) = \sum_{n=0}^{N-1} z^{-n} G_{k,n}(z^N)$$

frequency phase

# Polyphase Description, Synthesis (2)

Block diagram:




filters of analysis

output of synth. FB

$$\hat{\underline{X}}(z) = \underline{Y}(z) \times \underline{\underline{G}}(z) = \underline{X}(z) \times \underline{\underline{H}}(z) \times \underline{\underline{G}}(z)$$

subbands  filters of synth.

perfect reconstruction (PR) results if

$$\underline{\underline{G}}(z) = z^{-d}\,\underline{\underline{H}}^{-1}(z) \qquad (z^{-d} = \text{Delay})$$

Since by substitution we get

$$\hat{\underline{X}}(z) = \underline{X}(z) \times \underline{\underline{H}}(z) \times z^{-d} \times \underline{\underline{H}}^{-1}(z)$$

$$\Rightarrow \hat{\underline{X}}(z) = z^{-d} \cdot \underline{X}(z)$$

Observe: PR requires ‘only‘ a matrix inversion

**Problem: How to invert a matrix of polynomials?**





TECHNISCHE UNIVERSITÄT ILMENAU

Fraunhofer IDMT

# Example: Construction of H(z)

- How to obtain a H(z), which only has the first coefficient of our polynomial unequal to zero

→ design $h_k(n)$ such that $\underline{H}_k(z)$ has no higher powers of z:

$h_k(mN+n)$ $\quad \underline{H}_k(z)$

$$1,2,x\cdots \rightarrow 1+2\cdot z^{-1}+x\cdot z^{-2}+\cdots$$

$$1,0,0,\cdots \rightarrow 1+0\cdot z^{-1}+0\cdots = 1$$

$$a,0,0,\cdots \rightarrow a$$

m=0,1,2,…

Hence only the first block of our impulse response can be unequal to 0, and it is limited to a length of N! (too short!)

# Examples: The DFT as a filter bank

**Convolution (blockwise)**

$$y_k(m)=\sum_{n=0}^{N-1} x(mN-n)\cdot h_k(n)$$

The Discrete Fourier Transform can be written as

$$Y_k(m)=\sum_{n=0}^{N-1} x(mN-N+1+n)\cdot e^{-j\frac{2\pi}{N}kn}$$

with the block index $m=1\ldots L$

The substitution $n'=N-1-n$ yields

$$Y_k(m)=\sum_{n=0}^{N-1} x(mN-n')\cdot e^{-j\frac{2\pi}{N}k\cdot(N-1-n')}$$

This is a critically sampled filter bank with the impulse response (design trick: $h_k(n)$ is only as long as one block)

$$h_k(n)=e^{\frac{-j\cdot 2\pi}{N}\cdot k\cdot(N-1-n)} \qquad n,k=0\ldots N-1$$

frequency time

The analysis polyphase matrix of the DFT is identical to the DFT transform matrix:

$$\underline{\underline{H}}(z)=\underline{\underline{F}}=DFT-Matrix$$

->Perfect reconstruction, but filters not good enough!



Prof. Dr.-Ing. K. Brandenburg, bdg@idmt.fraunhofer.de Prof. Dr.-Ing. G. Schuller, shl@idmt.fraunhofer.de

# Examples: DFT

The fourier matrix is defined as

$$\underline{\underline{F}}_{n,k} = W^{nk}$$

$$\underline{\underline{F}} = \begin{bmatrix} W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & \cdots & W^{(N-1)} \\ \vdots & \vdots & & \vdots \\ W^0 & W^{(N-1)} & \cdots & W^{(N-1)(N-1)} \end{bmatrix}$$

← is identical to the polyphase matrix of the DFT viewed as a filter bank

with $W = e^{-j2\frac{\pi}{N}}$

# Example: The $DCT_{IV}$

**Convolution (blockwise)**

$$y_k(m)=\sum_{n=0}^{N-1} x(mN-n)\cdot h_k(n)$$

Using the same substitution as with the DFT, the Discrete Cosine Transform type 4 can be written as:

$$Y_k(m)=\sum_{n=0}^{N-1} x(mN-n)\cdot\cos\left(\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left((N-1-n)+\frac{1}{2}\right)\right)$$

with impulse response (N: block length):

$$h_k(n)=\cos\left(\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left((N-1-n)+\frac{1}{2}\right)\right) \quad n,k=0\ldots N-1$$

Special property: filter bank is orthogonal

$$\frac{2}{N}\underline{\underline{H}}^{T}(z^{-1})=z^{-d}\cdot\underline{\underline{H}}^{-1}(z)$$

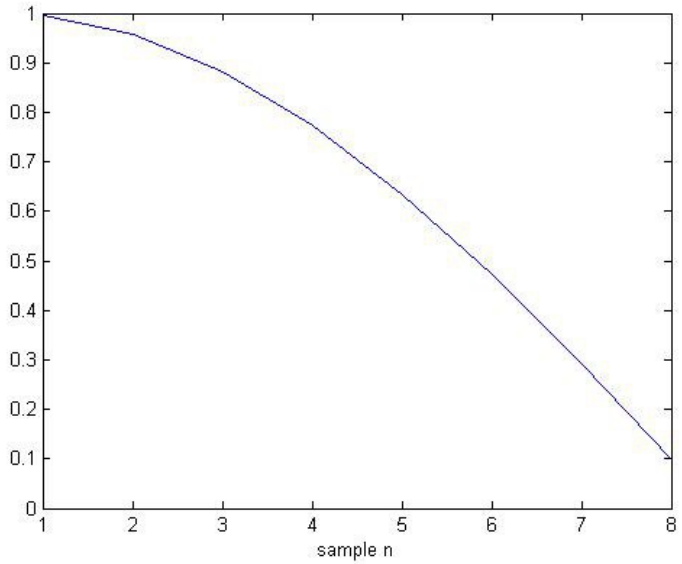
for Perfect Reconstruction, hence the synthesis is the transposed time reversed matrix

# DCT Type 4, with 8 Subbands (N=8)

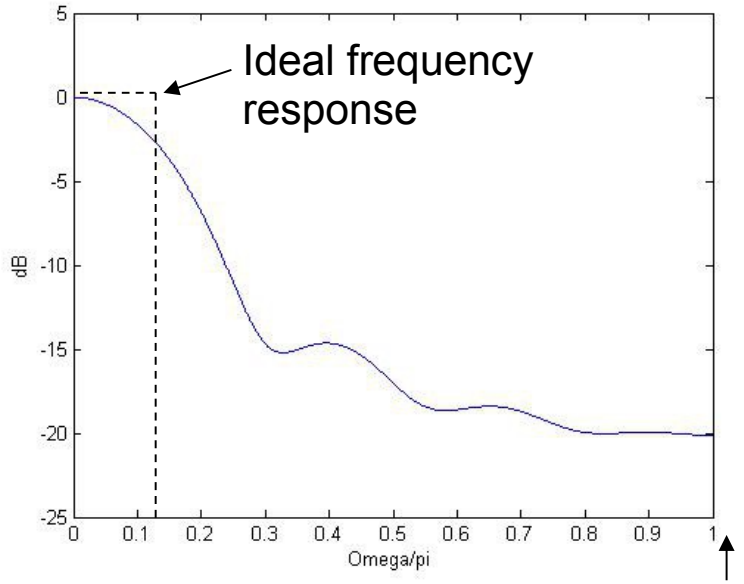
Filter for subband 0:

bad filter

Impulse response h_0(n)




Magnitude response

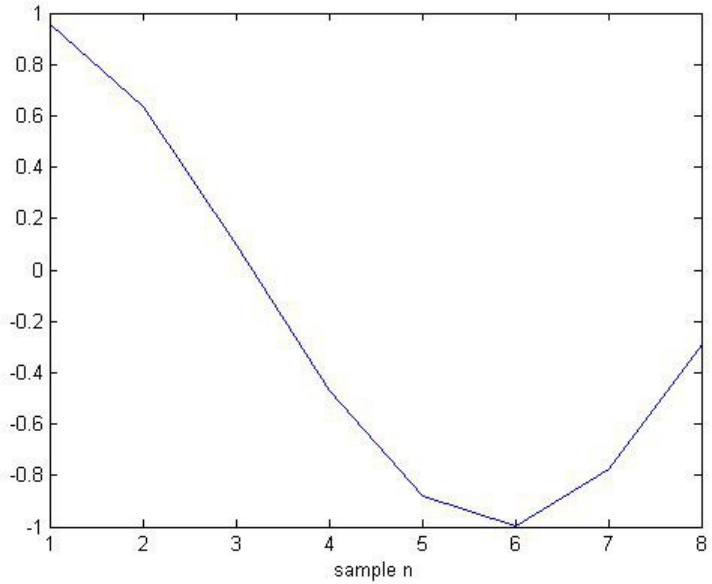



Nyquist frequency of input sampling rate

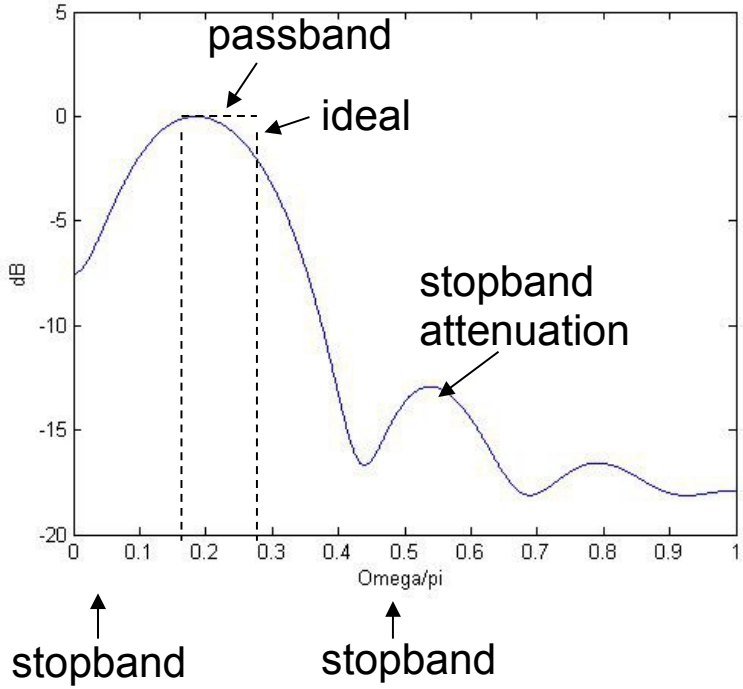
# DCT Type 4, with 8 Subbands (N=8)

Filter for subband 1:

Impulse response h_1(n)




Magnitude response



# Problem

- Except for the zeros, the stopband attenuation is not very high (still PR!)
- Problem especially for audio, since the zeros are not sufficient for good selectivity.
- Approach: design filter banks with longer filters, with better ability for higher stopband attenuation.
- → Really use z-domain for longer filters

# Python Examples

- Real-time python audio examples: you need a microphone and speakers connected.
- The first shows the real-time FFT of blocks of 1024 audio samples. Horizontally you seen the FFT bins or subbands, vertically the magnitude of the FFT coefficients/samples in dB:

  ```
  python  pyrecfftanimation.py
  ```

- **Observe:** The FFT subbands are symmetric around the center, the highest frequency (Nyquist frequency) is in the center. If you whistle, you see 2 peaks at the corresponding FFT subbands.

# Python Examples

- Next is a time-frequency representation, a spectrogram, which displays time on the vertical axis, and which shows the magnitude of the FFT coefficients as different colors:

  ```
  python pyrecspecwaterfall.py
  ```

- **Observe**: This shows the time-frequency nature of filter banks (of which the FFT is a special example). You have both, time and frequency dependencies.
- pyrecplayMDCT.py

# Python Examples

- Last is an example for the so-called MDCT filter bank. You see a decomposition of the audio signal into MDCT subbands. These subbands can then be processed, for instance we set every subband except for a few to zero. Then we display the result as a spectrogram waterfall diagramm, and use the inverse/synthesis MDCT for reconstrution and play the resulting sound back:

  ```
  python pyrecplayMDCT.py
  ```

- **Observe:** The MDCT does not have those symmetric 2 sides, it only has one side of the spectrum, with the lowest frequencies on the left side, and the hightest on the right.
- If we only keep a few subbands, it sounds muffled or „narrowband“.