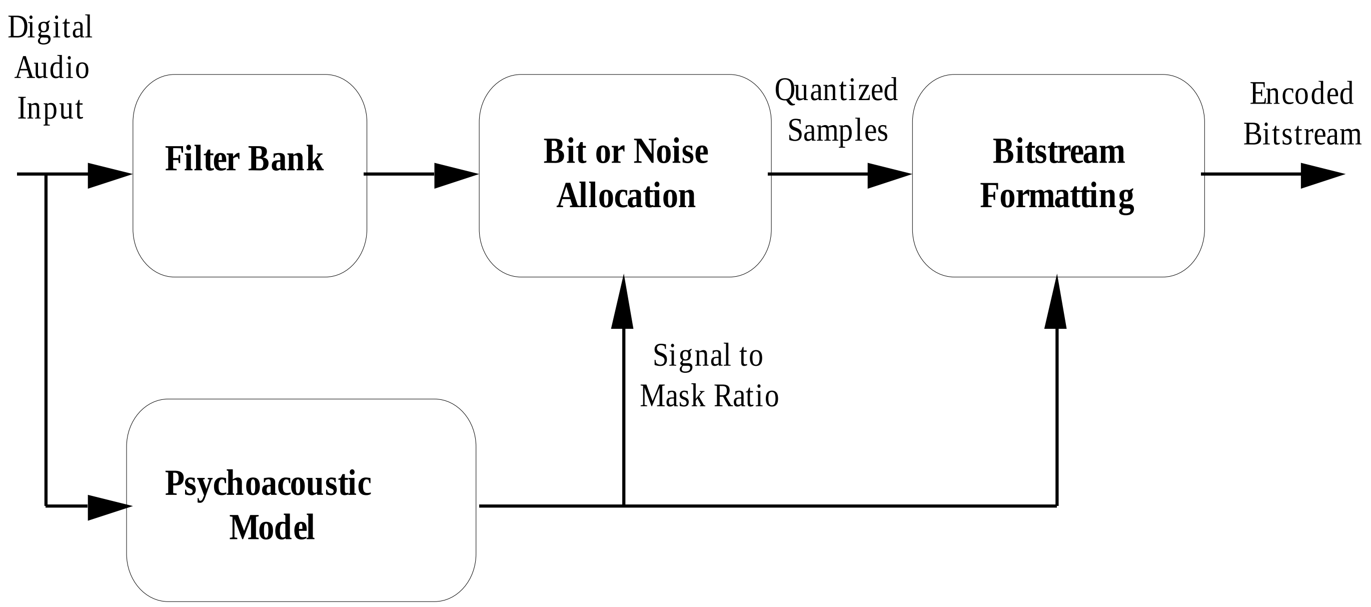
# PQMF Filter Bank, MPEG-1 / MPEG-2 BC Audio

TECHNISCHE UNIVERSITÄT ILMENAU

Fraunhofer IDMT

## The Basic Paradigm of T/F Domain Audio Coding



## MPEG Audio Standardization Philosophy (1)

Definition of a complete transmission chain consists of specification of

- Encoding algorithm
- Bitstream format
- Decoding algorithm

ITU-T Approach

- ITU-T standardizes all three parts
  ⇒ Encoder output predictable

MPEG Approach

- MPEG standardizes only bitstream format and decoder, not the encoder ("informative part")

## MPEG Audio Standardization Philosophy (2)

Motivation: open for further improvements, room for specific corporate know-how

But: No sound quality guaranteed !

## MPEG-1/2 Audio

MPEG-1 Audio

- Audio coding 32 - 48 kHz, mono/stereo
- Layer 1, 2, 3
- Layer-3 (aka .mp3) optimized for lower bit-rates
- Copy protection via SCMS included

MPEG-2 Audio

- Low sampling frequencies audio add 16 - 24 kHz to Layer 1, 2, 3
- Multichannel audio, BC (Backward Compatible)

## MPEG-1 Audio

Developed Dec. 88 to Nov. 92

Coding of mono and stereo signals

Bitrates from 32 kbit/s to 448 kbit/s

Three "Layers":

- Layer 1: lowest complexity
- Layer 2: increased complexity and quality
- Layer 3: highest complexity and quality at low bit-rates

Sampling frequencies supported:

- 48 kHz, 44.1 kHz, 32 kHz

## The main building blocks

Perceptual model

- using psychoacoustics, mostly proprietary

Filter bank

- subdividing the input signal into spectral components
- more lines → more coding gain
- longer impulse response → pre-echo artifacts

Quantization and coding

- this is the step introducing quantization noise
- spectral shape of quantization noise determines the audibility
- can be designed to leave encoding methods optional

## MPEG Audio - Short Description of the Layers (1)

Layer I

- Frame length: 384 samples (8 ms@ 48 kHz)
- Frequency resolution: **32 subbands from a PQMF filter bank**
- Quantization: Block-companding (12 samples), amplitude of subband samples indicated by “scalefactors” (SCF)

Layer II

- Frame length: 1152 samples (24 ms@ 48 kHz)

Frequency resolution: **32 subbands from a PQMF filter bank**

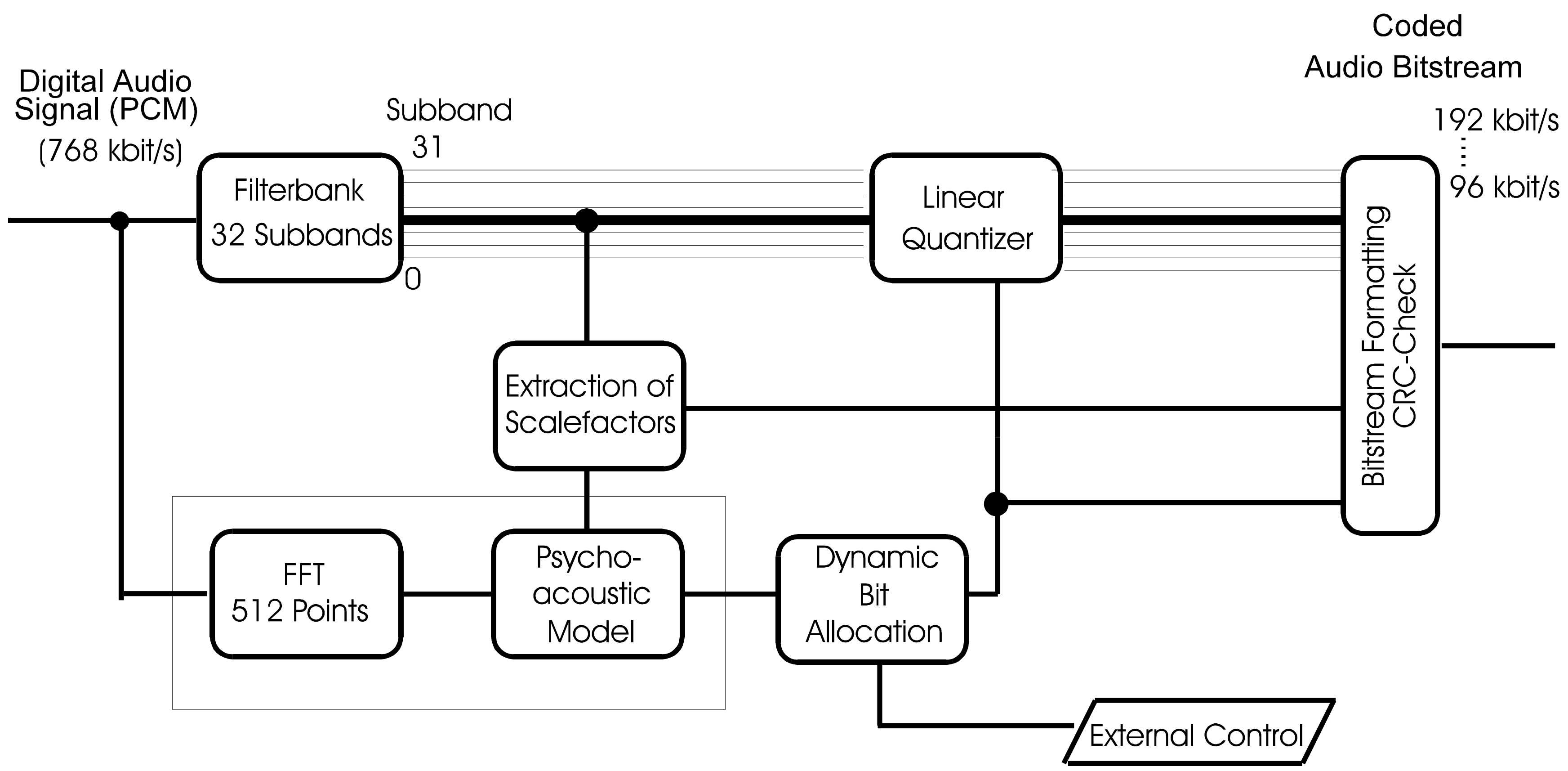
- Quantization: Block-companding (12 samples)
- Use of Scalefactor select information

## MPEG Audio - Short Description of the Layers (2)

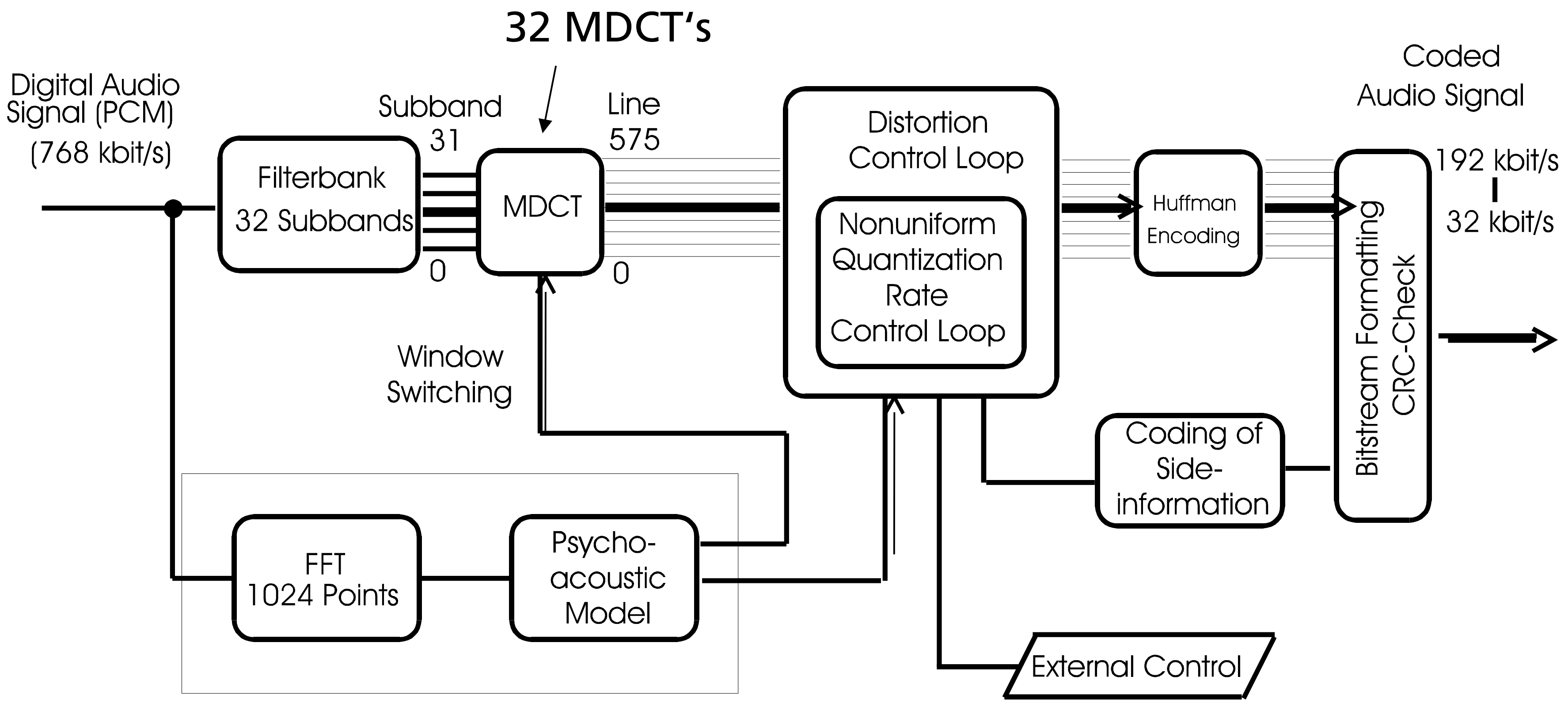
Layer III

- Standard frame length: 1152 samples (24 ms @ 48 kHz)
- Frequency resolution: **576 or 192 subbands**, from a **2-stage filter bank**:
  - a **32-band PQMF filter bank** in the first stage, followed by
  - a **6 or 18 band MDCT** in each of the 32 PQMF subbands (hence 32*6=192, or 32*18=576) in the second stage.
- Quantization: non-uniform with Huffman coding
- Use of Scalefactor Select Information

## Block Diagram MPEG-1 Layer 1



# Block diagram Layer-3



## Example for the Time/Frequency Resolution for the 2-Stage Layer III Coder

The first stage has a 32 subband QMF filter bank. For simplicity we only take an MDCT. We can visualize a 32 subband filter bank with our Python program “pyrecplayMDCT.py”, by editing the line for the number of subbands N to:

```
N=32;
```

And run it with

```
python pyrecplayMDCT.py
```

**Observe**: We get a very **narrow spectrogram**, because we only have 32 subbands, but it **runs very fast**, because we donwsample only by 32. We can also **listen to a subband** (by setting the other subbands to zero), and we will hear more than just a tone, because the subbands are wider than for AAC.

## Example for the Time/Frequency Resolution for the 2-Stage Layer III Coder

Now set it to the number of subbands after the second stage:

```
N=576
```

And run it with

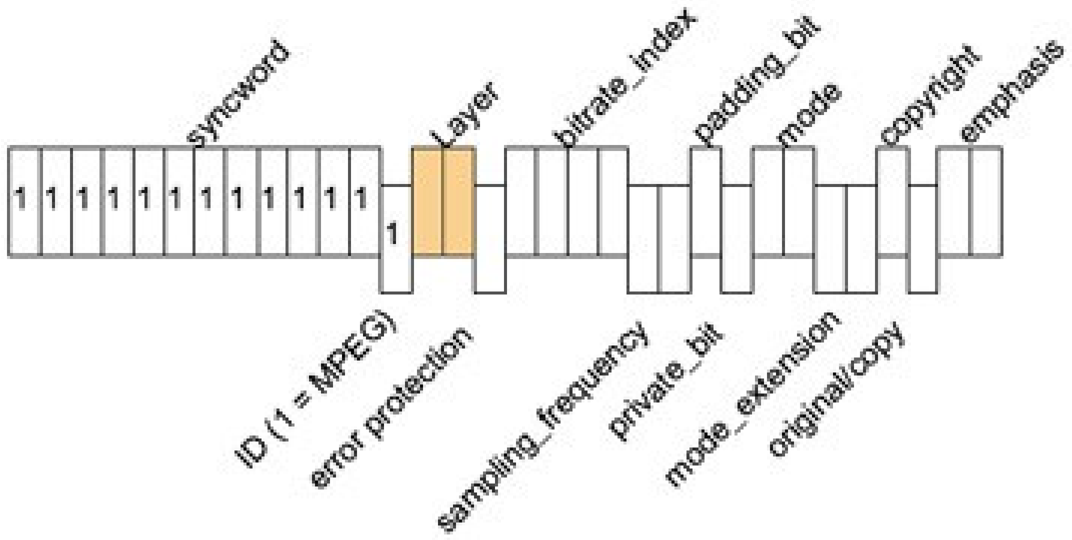
```
python pyrecplayMDCT.py
```

Here now it becomes more similar to the AAC filter bank. We see a **broader spectrogram**, but it **runs slower**, because we have an increased downsampling rate of 576.

The second layer does it by replacing every 12 or 18 time samples in each of the 32 subbands by 12 or 18 finer subbands.

When we now **listen** to a subband, it sound more like a **tone,** because the subbands are more narrow.

## MPEG - Layer-1, -2 and -3 Compression: Header



Observe: The syncword allows decoding starting in the middle of the file, which is important for broadcast applications.

## The Pseudo-Quadrature-Mirror Filter Bank (PQMF)

The PQMF is also used for

- parametric surround coding
- Parametric high frequency generation

For that reason we now take a closer look at the PQMF filter bank (See also lecture slides Filter Banks 2)

## The Pseudo-Quadrature-Mirror Filter Bank (PQMF)

- The PQMF is a filter bank which has only "**near**" **perfect reconstruction**, unlike the MDCT, which has really perfect reconstruction.
- In the same sense it is **near para-unitary or orthogonal,** which means the **synthesis filter are the time reversed analysis filters**.
- But it has the advantage that we have design methods to design filter banks with "good" longer filters, with more overlap in time than we use with the MDCT. This results in **improved nearby stopband attenuation**.
- An often used configuration is **32 subbands** with filters with length of **512 or 640 coefficients**, hence **16 or 20-times overlap** in time (compare that with 2-times overlap in the MDCT)

## PQMF Definition

The analysis filters of an N-band PQMF filter bank with filters of length L are defined as (see also Slides “Filter Banks 2”):

$$h_k(n)=h(n)\cdot\cos\left(\frac{\pi}{N}\cdot(k+0.5)\cdot(n-\frac{L}{2}+\frac{1}{2})+\Phi_k\right)$$

With the phase defined as:

$$\Phi_{k+1}-\Phi_k=\frac{(2r+1)\cdot\pi}{2}$$

With some integer r. For convenience we omit scaling factors.
The synthesis filters are the time reverse analysis filters:

$$g_k(n)=h_k(L-1-n)$$

(See also: https://ccrma.stanford.edu/~jos/sasp/Pseudo_QMF_Cosine_Modulation_Filter.html
Book: Spanias, Painter: “Audio Signal Processing and Coding”, Wiley Press)

## PQMF Reformulation

For the case of $\Phi_k = (-1)^{k+1} \cdot \frac{\pi}{4}$

And $L/N \, mod \, 4 = \pm 2$

We find that it is identical to our MDCT modulation function (with its time reversal now on the right side of the equation):

$$h_k(n) = h(n) \cdot \cos\left(\frac{\pi}{N} \cdot \left(k + \frac{1}{2}\right)\left(L - 1 - n + \frac{1}{2} - \frac{N}{2}\right)\right)$$

## PQMF Design

Take the frequency response of the baseband prototype or window function:

$$H(\omega) = DTFT(h(n))$$

Then we need to find (optimize) a function h(n), such that it fulfills:
-**Attenuation**: The **stopband attenuation** should be high after the neighboring band to **minimize aliasing (stop band)**:

$$|H(\omega)| \approx 0 \; for \; 1.5\frac{\pi}{N} < |\omega| < \pi$$

(passband: 0..0.5pi/N, transition band: 0.5pi/N..1.5pi/N)
-**Unity condition (overlap region of 2 subbands)**: Sum of magnitude squared frequency responses of 2 **neighboring bands** should be close to to a **constant**, here 2N^2, to achieve **near perfect reconstruction**:

(pos freq. + flipped version)

$$|H(\omega)|^2 + \left|H\left(\frac{\pi}{N} - \omega\right)\right|^2 \approx 2 \cdot N^2 \; for \; 0 \leq \omega < \frac{\pi}{N}$$

## Python Example Optimization

To fulfill these requirements, we now have an **optimization problem**. Python has powerful optimization libraries to find a solution. Take a very simple example: **find the minimum** of the function of 2 variables $f(x_{1,} x_2) = \sin(x_1) + \cos(x_2)$

In Python we write it as a function in file `functionexamp.py`:

```
#function example with several unknowns (variables) for optimization
#Gerald Schuller, Nov. 2016
import numpy as np

def functionexamp(x):
  #x: array with 2 variables

  y=np.sin(x[0])+np.cos(x[1])
  return y
```

## Python Example Optimization

Next we use the library scipy.optimize to find a minimum, and use its function “minimize”. We save it for instance as `optimizationExample.py`

```
#Optimization example, see also:
#https://docs.scipy.org/doc/scipy-0.18.1/reference/optimize.html
#Gerald Schuller, Nov. 2016
#run it with "python optimizationExample.py" in a termina shell
#or type "ipython" in a termina shell and copy lines below:

import numpy as np
import scipy.optimize as optimize
from functionexamp import functionexamp

#Example for 2 unknowns, args: function-name, starting point, method:
from functionexamp import *
xmin=optimize.minimize(functionexamp,[-1.0,-3.0], method='CG')
print xmin
```

And call it with

```
python optimizationExample.py
```

Observe: We indeed obtain the minimium at $x_1=-\pi/2, x_2=-\pi$

## PQMF Optimization, Python Example, Optimization Function

```
import numpy as np
import scipy as sp
import scipy.signal as sig

def optimfuncQMF(x):
    """Optimization function for a PQMF Filterbank
    x: coefficients to optimize (first half of prototype h because of symmetry)
    err: resulting total error"""

    N=4  #4 subbands
    h = np.append(x,np.flipud(x));
    f,H_im = sig.freqz(h)
    H=np.abs(H_im) #only keeping the magnitude

    posfreq = np.square(H[0:512/N]);
    #Negative frequencies are symmetric around 0:
    negfreq = np.flipud(np.square(H[0:512/N]))
    #Sum of magnitude squared frequency responses should be close to unity (or N)
    unitycond = np.sum(np.abs(posfreq+negfreq - 2*(N*N)*np.ones(512/N)))/512;
    #plt.plot(posfreq+negfreq);
    #High attenuation after the next subband:
    att = np.sum(np.abs(H[1.5*512/N:]))/512;

    #Total (weighted) error:
    err = unitycond + 100*att;
    return err
```

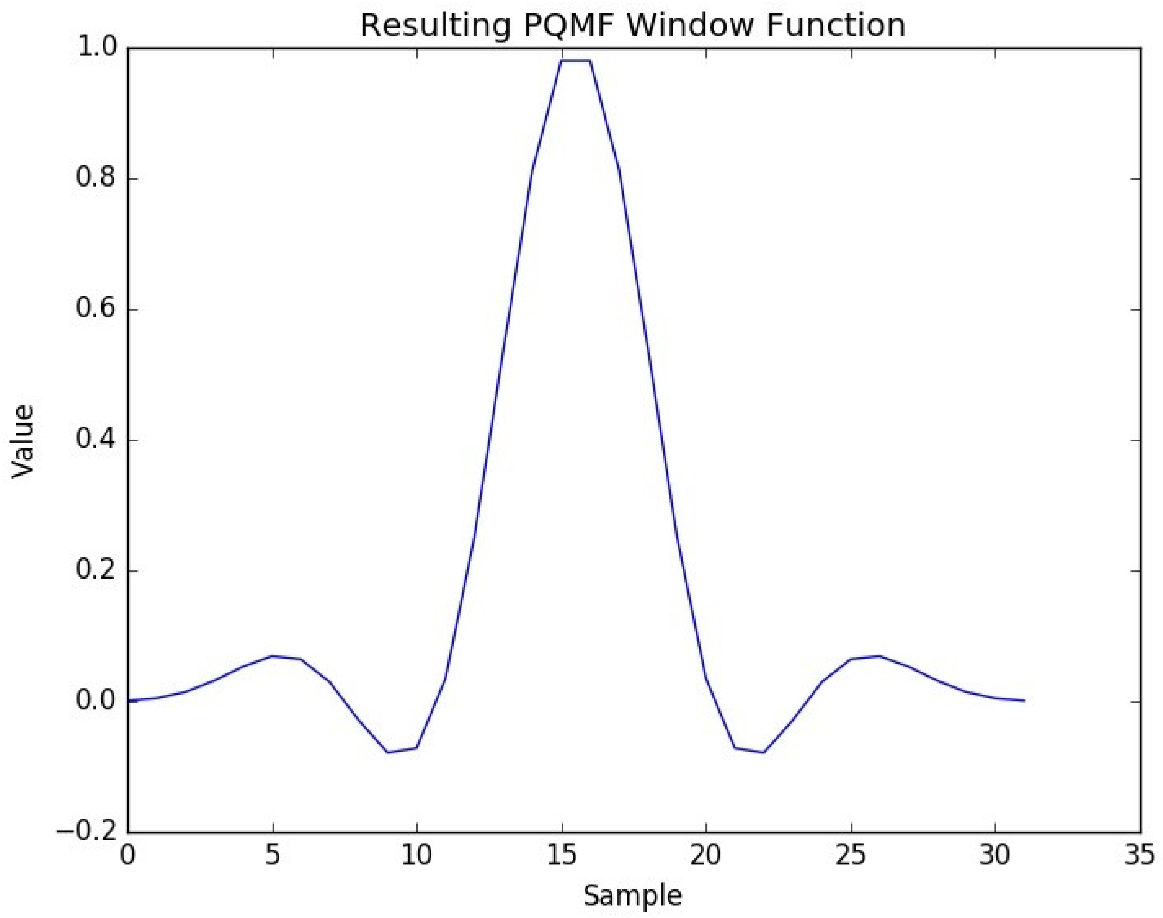
## PQMF Optimization, Python Example, Optimizer

Now we have a function to minimize, and we can use Pythons powerful optimization library to minimize this function:

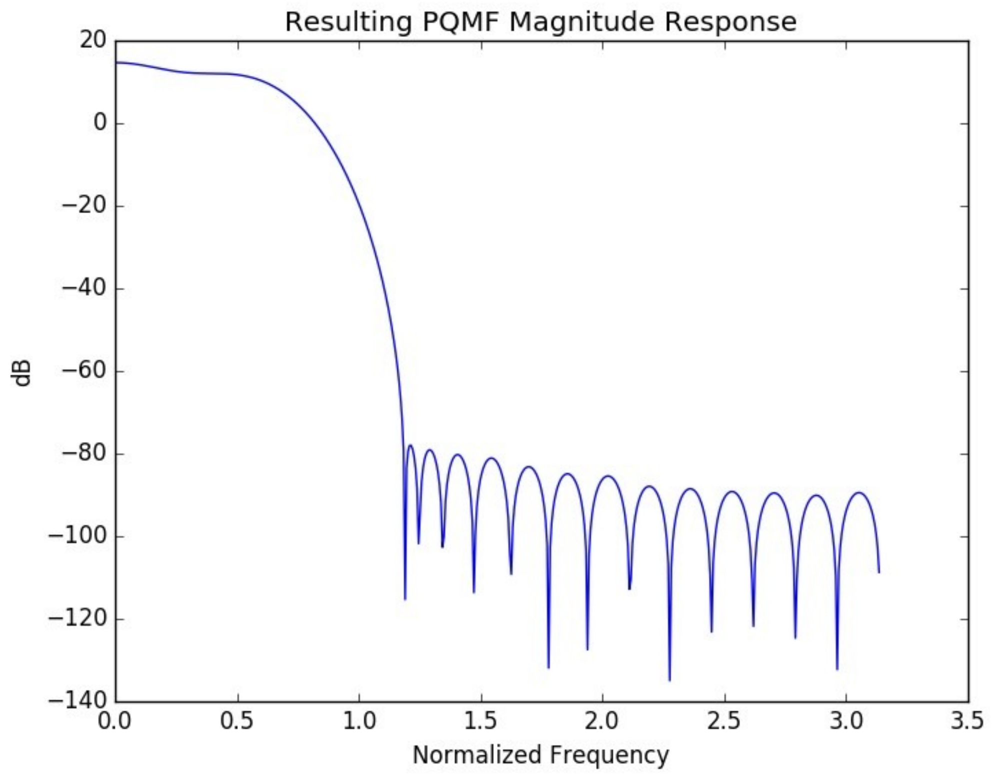
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt
import scipy.signal as sig
from optimfuncQMF import optimfuncQMF
#optimize for 16 filter coefficients:
xmin = opt.minimize(optimfuncQMF,16*np.ones(16),method='SLSQP')
xmin = xmin["x"]

#Restore symmetric upper half of window:
h = np.concatenate((xmin,np.flipud(xmin)))

plt.plot(h)
plt.show()
f,H = sig.freqz(h)
plt.plot(f,20*np.log10(np.abs(H)))
plt.show()
```

Let it run with: `python PQMF_optimization.py`

## PQMF Optimization, Python Example, Optimized Results



# PQMF Optimization, Python Example, Attenuation Condition



**Observe**: We get almost 100 dB Stopband attenuation, much more than with the MDCT!

Prof. Dr.-Ing. K. Brandenburg, karlheinz.brandenburg@tu-ilmenau.de Prof. Dr.-Ing. G. Schuller, gerald.schuller@tu-ilmenau.de

## PQMF Optimization, Python Example, Unity Condition

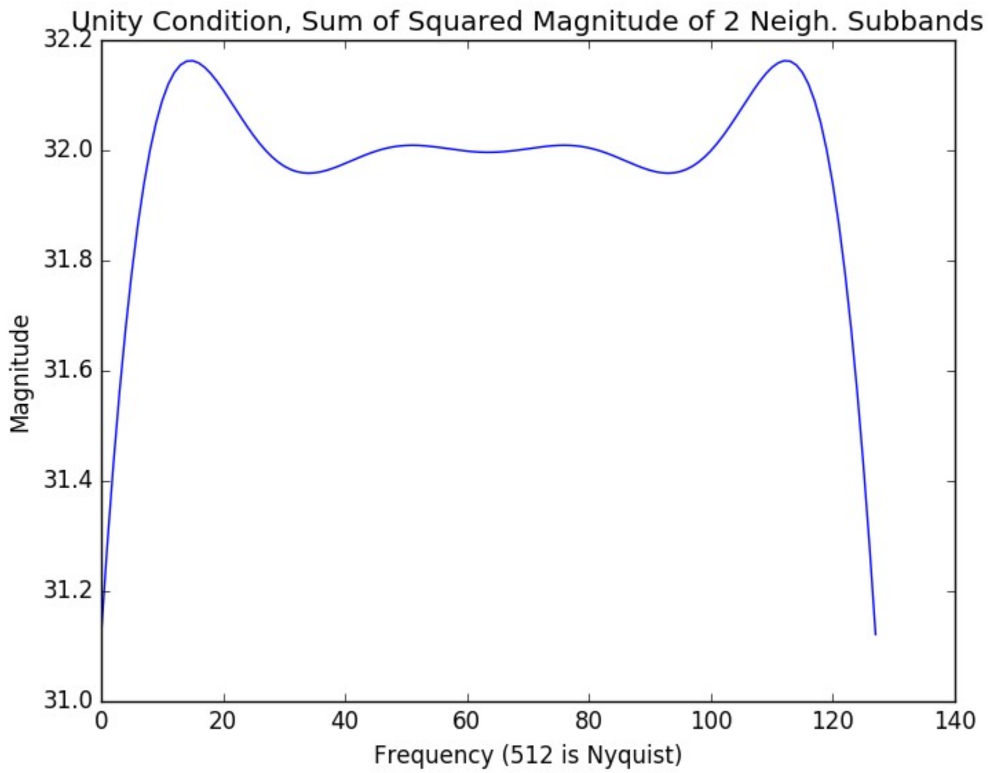
We can test the PQMF Unity condition (slide 19)

$$\left|H(\omega)^2\right| + \left|H(\omega+\pi/N)^2\right| \approx 2 \cdot N^2 \; for \; 0 \le \omega < \frac{\pi}{N}$$

(2N^2=32), with the following Python code,

```
f,H = sig.freqz(h)
posfreq = np.square(H[0:512/N]);
negfreq = np.flipud(np.square(H[0:512/N]))
plt.plot(np.abs(posfreq)+np.abs(negfreq))
plt.xlabel('Frequency (512 is Nyquist)')
plt.ylabel('Magnitude')
plt.title('Unity Condition, Sum of Squared
Magnitude of 2 Neigh. Subbands')
plt.show()
```

## PQMF Optimization, Python Example, Unity Condition



**Observe**: We get indeed a curve close to 2N^2=32, but with some Deviation, which shows that we get indeed only “near” Perfect Reconstruction!

## PQMF Optimization, Python Example, Optimized Results

- Observe: We obtain a 4-band filter bank with filter length of 32 taps, hence 8 times overlap.
- The stopband attenuation reaches almost 100 dB, almost right after the passband, much more than with the MDCT!

# PQMF Optimization, Polyphase Implementation, Analysis

- The advantage of having the same modulation function as the MDCT is that we can take the same type of filter matrices to implement it efficiently. The analysis filter matrix F_a is now:

$$\mathbf{F_a}(z) = \begin{bmatrix} & & H_{2N-1}^{\downarrow 2N}(-z^2)z^{-1}, \quad H_{N-1}^{\downarrow 2N}(-z^2) & & \\ & & & \ddots & \\ H_{N+N/2}^{\downarrow 2N}(-z^2)z^{-1} & & & & H_{N/2}^{\downarrow 2N}(-z^2) \\ H_{N+N/2-1}^{\downarrow 2N}(-z^2)z^{-1} & & & & -H_{N/2-1}^{\downarrow 2N}(-z^2) \\ & \ddots & & & \\ & & H_{N}^{\downarrow 2N}(-z^2)z^{-1}, \quad -H_{0}^{\downarrow 2N}(-z^2) & & \end{bmatrix}$$

$$H_n^{\downarrow 2N}(z) := \sum_{m=0}^{\infty} h(m2N+n)z^{-m}$$

## PQMF Optimization, Polyphase Implementation, Synthesis

- The synthesis filter matrix F_s is:

$$\mathbf{F_s}(z) = \left[\begin{array}{cccc} & H'^{\downarrow 2N}_{N/2-1}(-z^2), & H'^{\downarrow 2N}_{N/2}(-z^2) & \\ & & & \ddots \\ H'^{\downarrow 2N}_{0}(-z^2) & & & H'^{\downarrow 2N}_{N-1}(-z^2) \\ H'^{\downarrow 2N}_{N}(-z^2)z^{-1} & & & -H'^{\downarrow 2N}_{2N-1}(-z^2)z^{-1} \\ \ddots & & & \\ & H'^{\downarrow 2N}_{N+N/2-1}(-z^2)z^{-1}, & -H'^{\downarrow 2N}_{N+N/2}(-z^2)z^{-1} & \end{array}\right]$$

$$H'^{\downarrow 2N}_{n}(z) := \sum_{m=0}^{\infty} g(m2N+n)z^{-m}$$

## PQMF Optimization, Polyphase Implementation, Analysis and Synthesis

All together we obtain for the analysis filter bank (see also slides Filter Banks 1, with **T** the DCT transform matrix):

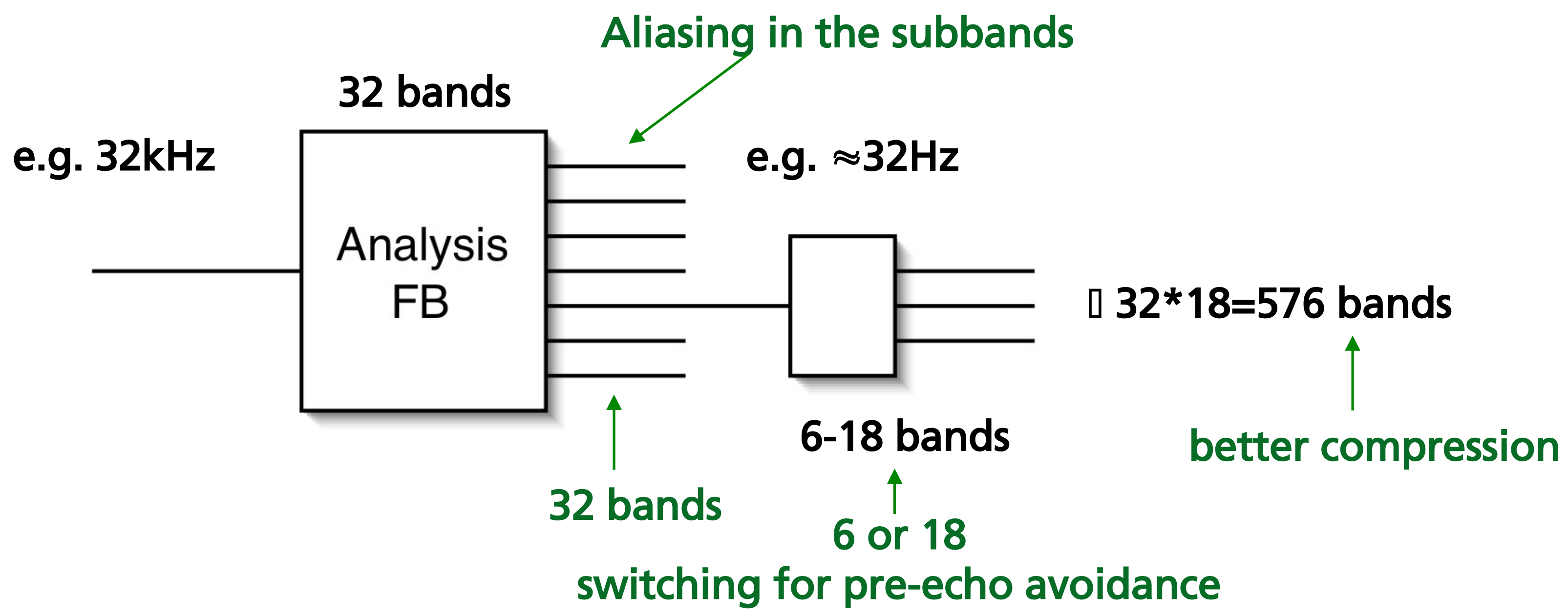
$$\underline{\boldsymbol{Y}}(z) = \underline{\boldsymbol{X}}(z) \cdot \underline{\underline{\boldsymbol{F}_a}}(z) \cdot \underline{\underline{\boldsymbol{T}}}$$

For the reconstruction of synthesis filter bank we got:

$$\underline{\hat{\boldsymbol{X}}}(z) = \underline{\boldsymbol{Y}}(z) \cdot \underline{\underline{\boldsymbol{T}^{-1}}} \cdot \underline{\underline{\boldsymbol{F}_s}}(z)$$

This formulation can also be used for the implementation of the PQMF filter bank.

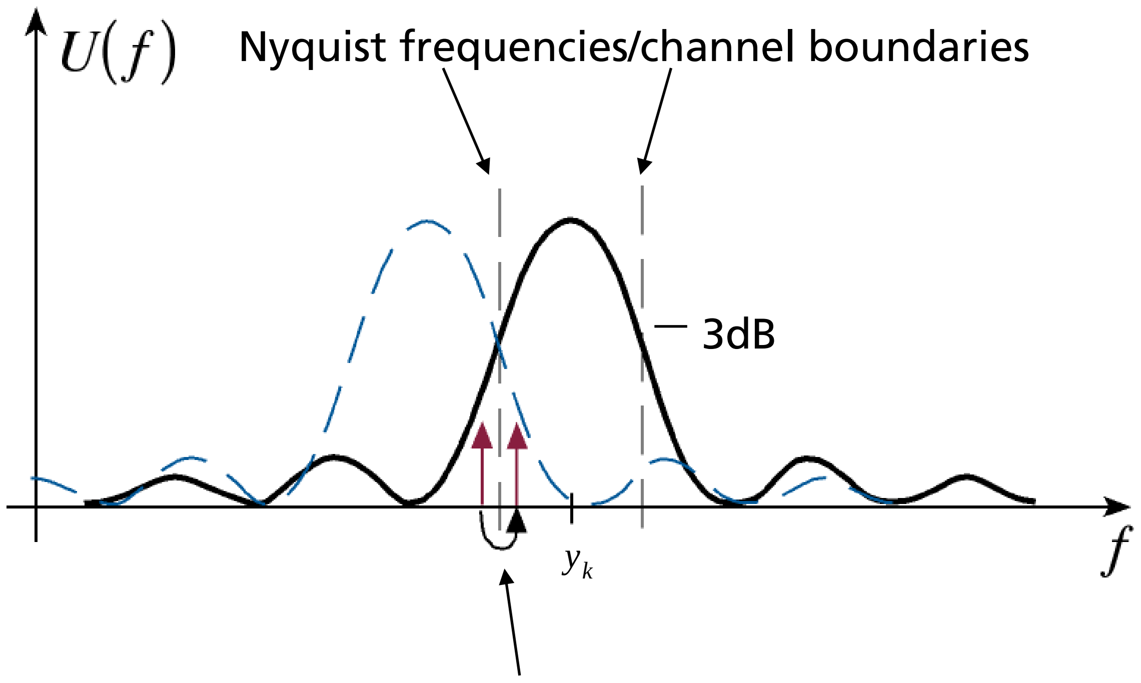
# Hybrid Filter Bank & Aliasing (1)



Filter bank is critically sampled

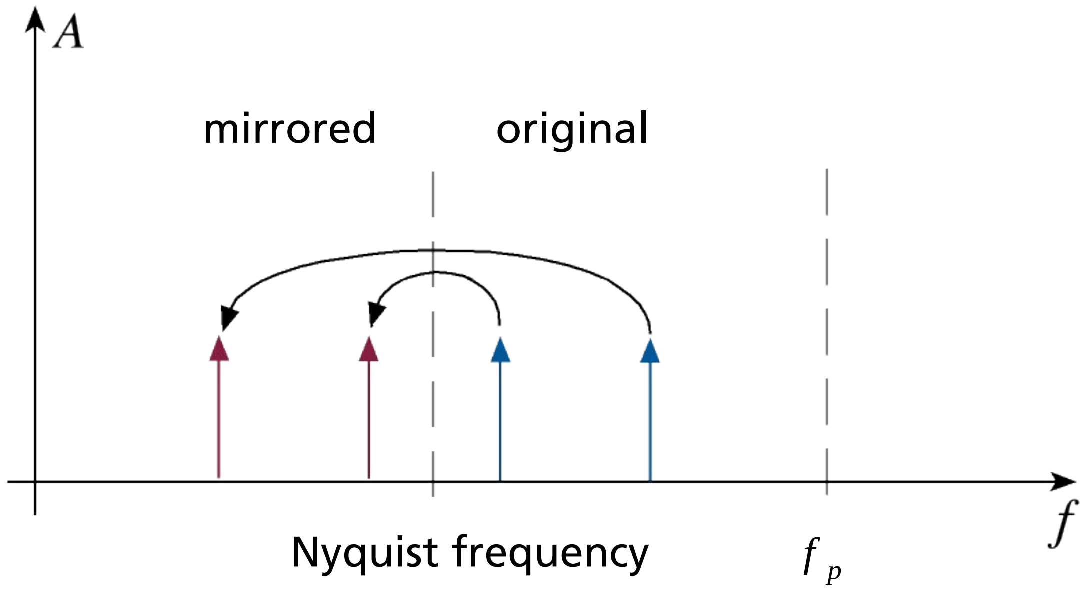
Problem of aliasing in the analysis filter bank

## Hybrid Filter Bank & Aliasing (2)



signal mirrored inside
aliasing cannot be neglected

## Hybrid Filter Bank & Aliasing (3)

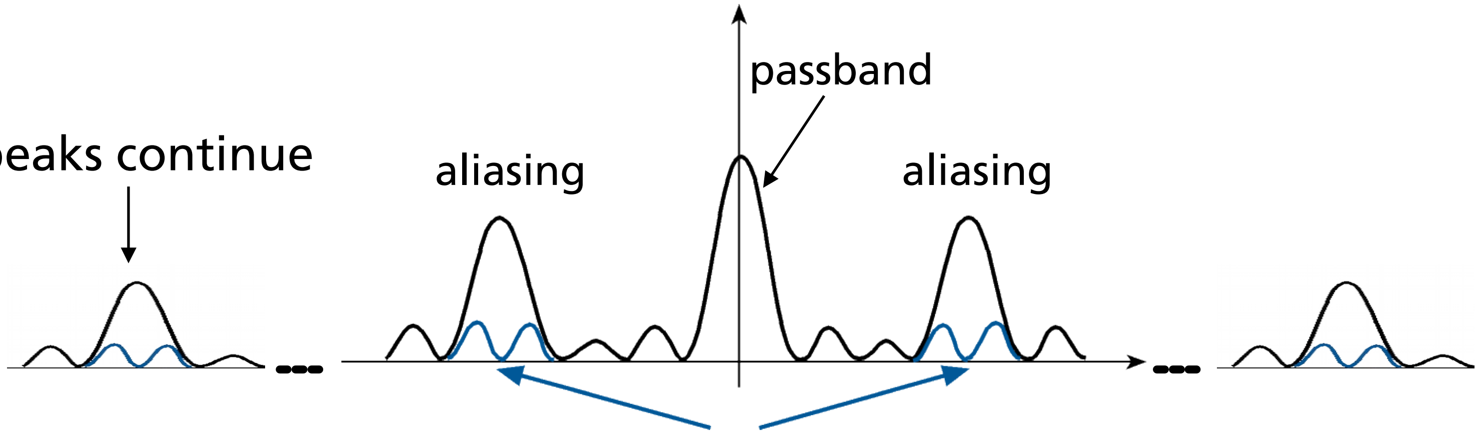


Mirroring of the original signals greater than the Nyquist frequency

- occurs on downsampling

# Problem of Aliasing in a Cascaded Filter Bank (1)

Frequency response contains peaks from the aliasing:



Reduced by alias reduction

not just one passband, but several, only slightly attenuated

Signal in many subbands

Must be coded in many subbands

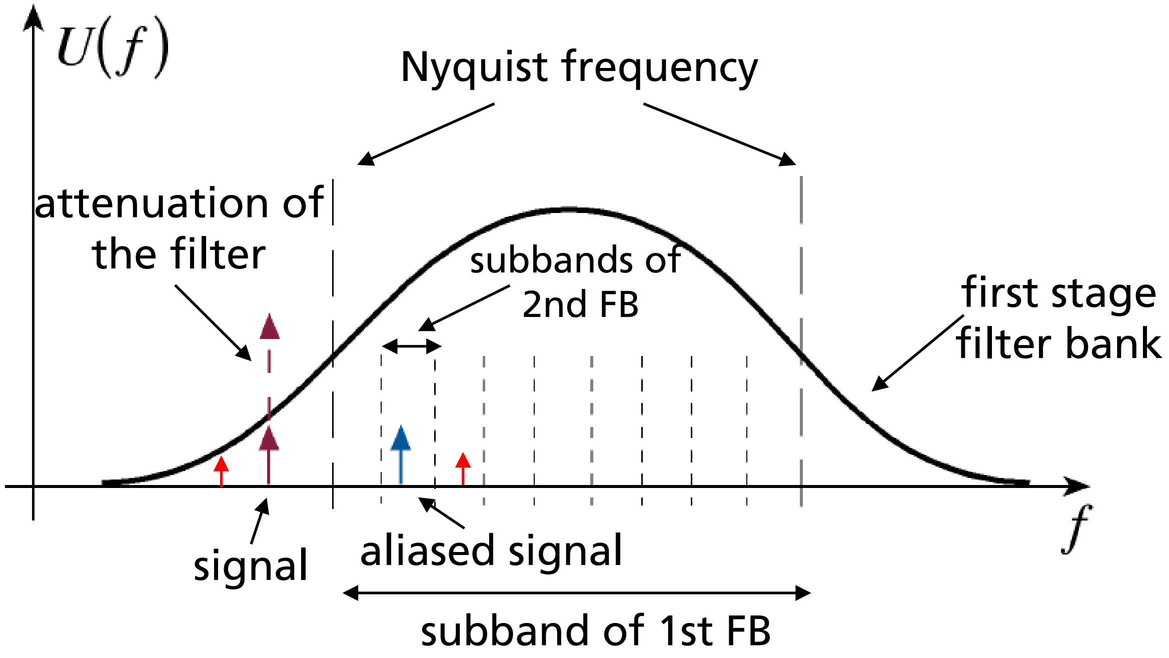
- Reduced coding efficiency

# Problem of Aliasing in a Cascaded Filter Bank (2)

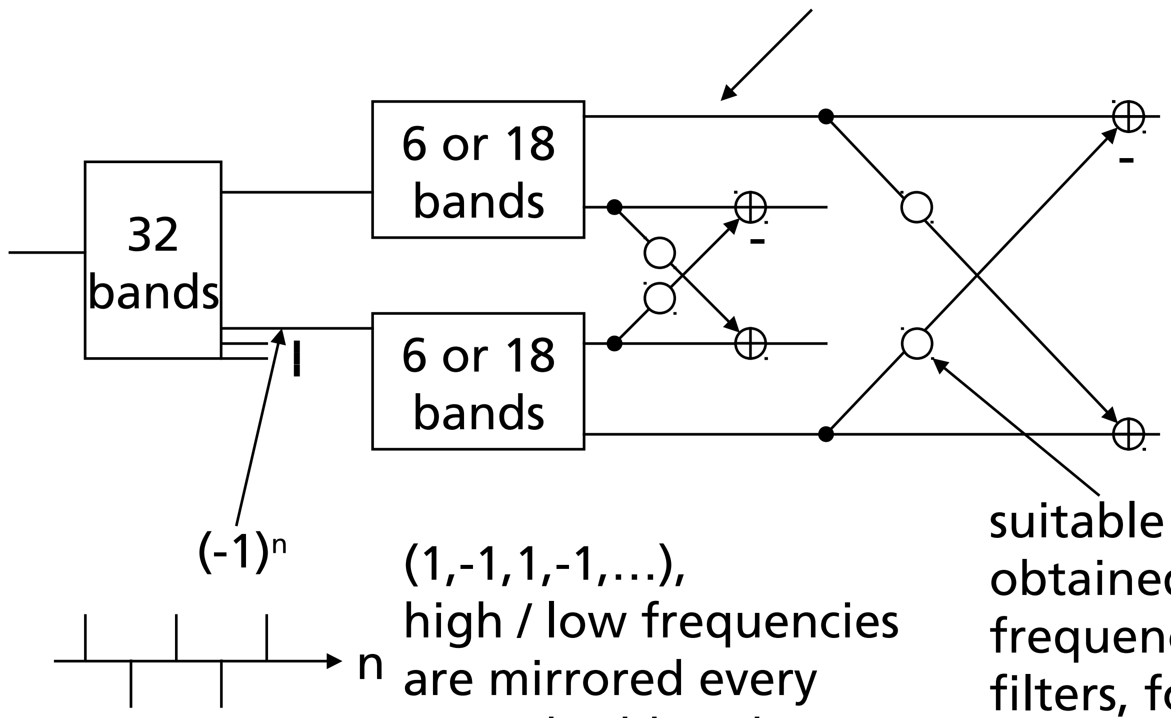
Result of cascading:

- Far off frequencies are aliased into the SB of the 2nd filter bank

Observe:
Aliasing spreads over several subbands of second stage! Not the case for only single stage FB (practically)



# Aliasing Reduction Structure (MP3)



similar approach as synthesis FB uses to reduce/cancel aliasing

(1,-1,1,-1,...), high / low frequencies are mirrored every second subband

suitable factors (fixed), obtained from frequency response of filters, for equal level of aliasing

## MPEG Audio - Layer-3: Bitstream

Organization of the bit streams

Fixed length of bytes: 17 at mono, 32 at stereo, independent of the bitrate

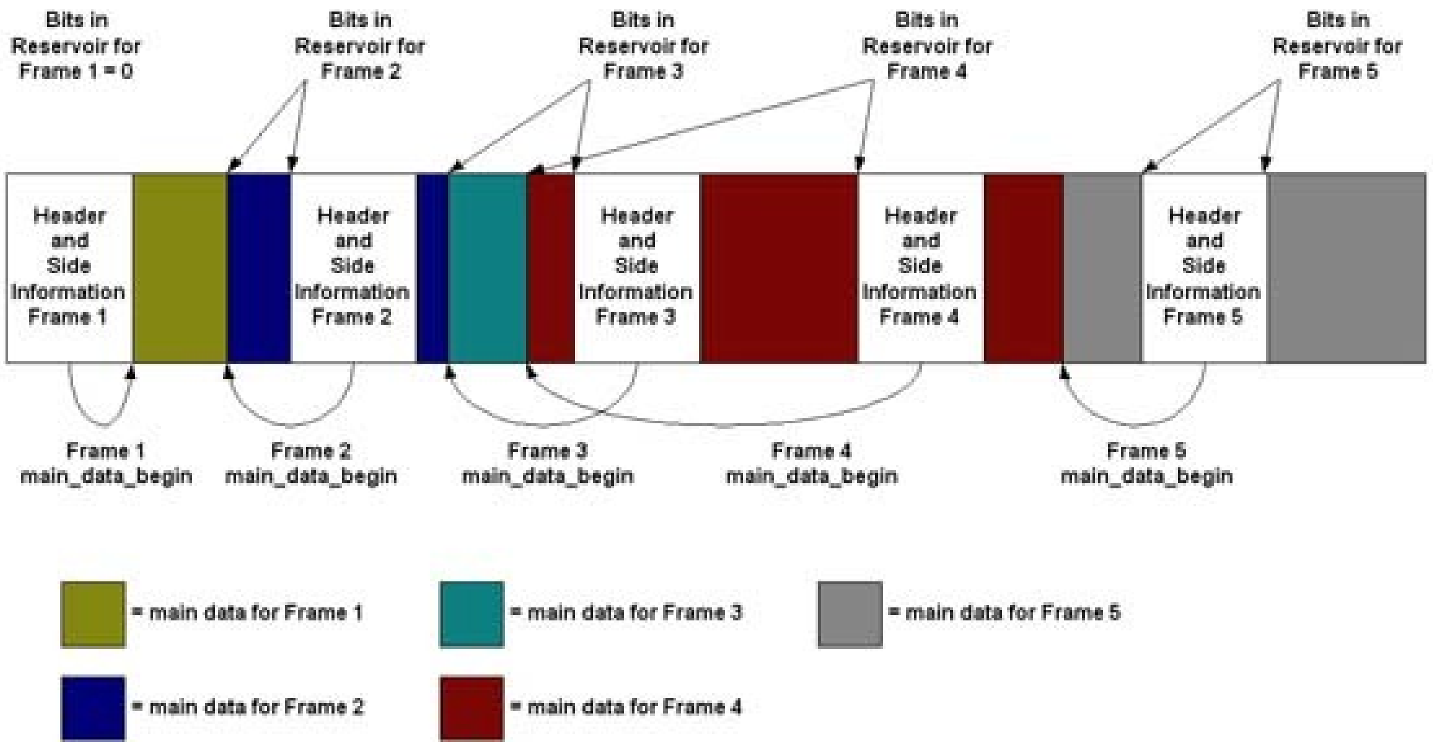
Constant Section

- Header (ISO Standard, like with Layer-1 and -2)
- Additional information for a frame (e.g. Pointer to the variable section)
- Additional information for each “granule” (e.g. Number of the Huffman-Code table)

Variable Section

- Scalefactors
- Huffman-coded frequency lines
- Additional Data

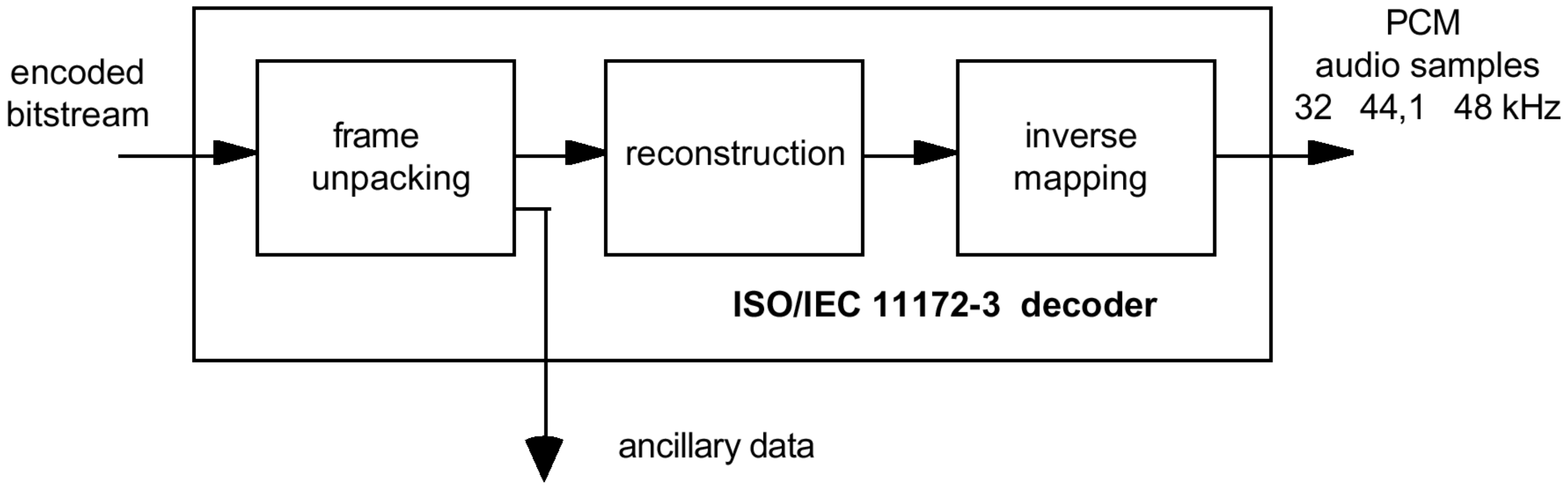
# MPEG Audio - Layer-3: Bitstream (2)


Bits in Reservoir for Frame 1 = 0
Bits in Reservoir for Frame 2
Bits in Reservoir for Frame 3
Bits in Reservoir for Frame 4
Bits in Reservoir for Frame 5
Header and Side Information Frame 1
Header and Side Information Frame 2
Header and Side Information Frame 3
Header and Side Information Frame 4
Header and Side Information Frame 5
Frame 1 main_data_begin
Frame 2 main_data_begin
Frame 3 main_data_begin
Frame 4 main_data_begin
Frame 5 main_data_begin
= main data for Frame 1
= main data for Frame 3
= main data for Frame 5
= main data for Frame 2
= main data for Frame 4

## MPEG-1 Audio Decoder

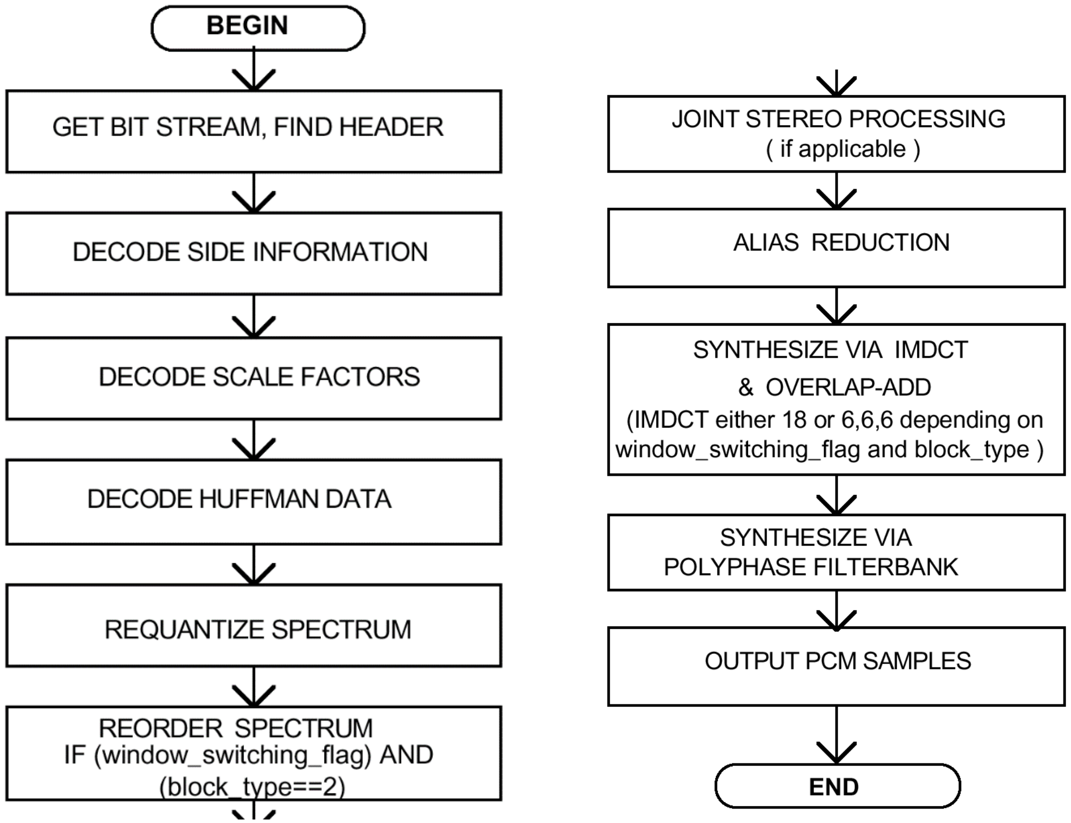
Prof. Dr.-Ing. K. Brandenburg, karlheinz.brandenburg@tu-ilmenau.de Prof. Dr.-Ing. G. Schuller, gerald.schuller@tu-ilmenau.de

# MPEG Audio – General Decoder Structure

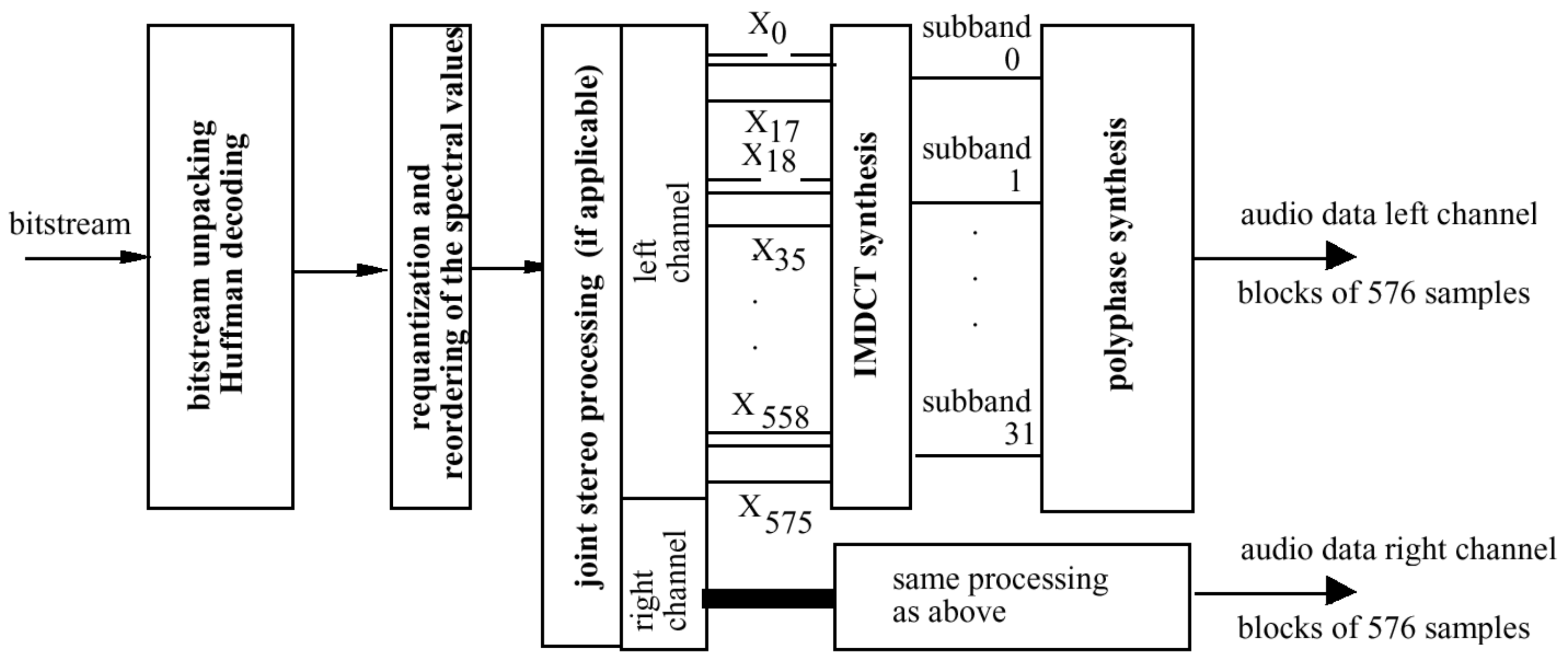


Prof. Dr.-Ing. K. Brandenburg, karlheinz.brandenburg@tu-ilmenau.de Prof. Dr.-Ing. G. Schuller, gerald.schuller@tu-ilmenau.de

## MPEG - Audio Decoder Process (1) Layer-3 Decoder flow chart


BEGIN
GET BIT STREAM, FIND HEADER
DECODE SIDE INFORMATION
DECODE SCALE FACTORS
DECODE HUFFMAN DATA
REQUANTIZE SPECTRUM
REORDER SPECTRUM IF (window_switching_flag) AND (block_type==2)
JOINT STEREO PROCESSING ( if applicable )
ALIAS REDUCTION
SYNTHESIZE VIA IMDCT & OVERLAP-ADD (IMDCT either 18 or 6,6,6 depending on window_switching_flag and block_type )
SYNTHESIZE VIA POLYPHASE FILTERBANK
OUTPUT PCM SAMPLES
END

# MPEG - Audio Decoder Process (2) Layer-3 Decoder Diagramm



## Annex: Abbreviations and Companies

## Abbreviations and Companies (1)

- **AAC:** Advanced Audio Coding
- **ASPEC:** Adaptive Spectral Perceptual Entropy Coding
- **AT&T:** American Telephone and Telegraph Company
- **CCETT:** Centre Commun d'Etudes de Télédiffusion et Télécommunication
- **CNET:** Research and Development Center of France Télécom
- **FhG-IIS:** Fraunhofer Gesellschaft/Institut für Integrierte Schaltungen (Erlangen)

## Abbreviations and Companies (2)

- **IRT:** Institut für Rundfunktechnik GmbH, München, Research and Development Institute of ARD, ZDF, DLR, ORF and SRG
- **ITU-R:** International Telecommunication Union – Radio Communication Sector
- **MASCAM:** Masking-pattern Adapted Subband Coding and Multiplexing AT&T:American Telephone and Telegraph Company
- **MUSICAM:** Masking-pattern Universal Subband Integrated Coding and Multiplexing

## Abbreviations and Companies (3)

- **NTT:** Nippon Telegraph and Telephone Corp./Human Interface Laboratories
- **Thomson:** Thomson, Telefunken, Saba, RCA, GE, ProScan
- **TwinVQ:** Transform-domain Weighted Interleave Vector Quantization