Frequency-domain processing: FFT, filter banks.

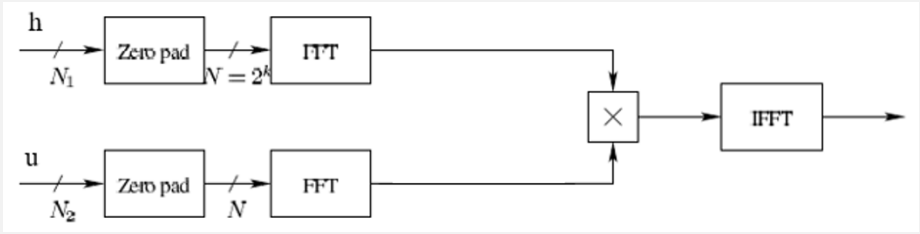
Signal Processing Systems Fall 2025

Lecture 11 (Monday 1.12.)

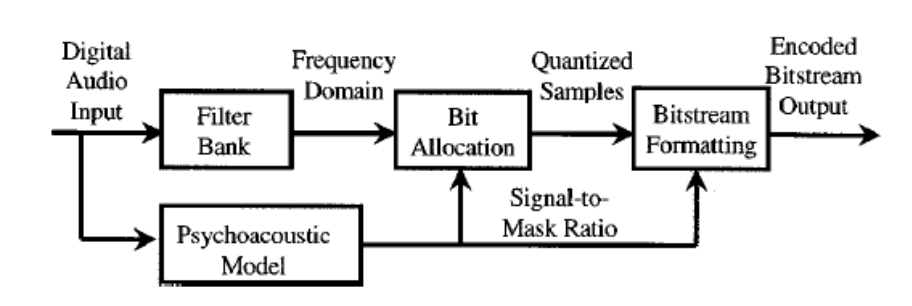
Outline

- Discrete Fourier Transform
  - Definition & basics of FFT: Cooley-Tukey algorithm, butterfly diagrams,
  - Computational complexity
- Convolution: in time or frequency domain?
- Filter banks
  - Modulated filter bank: multirate implementation

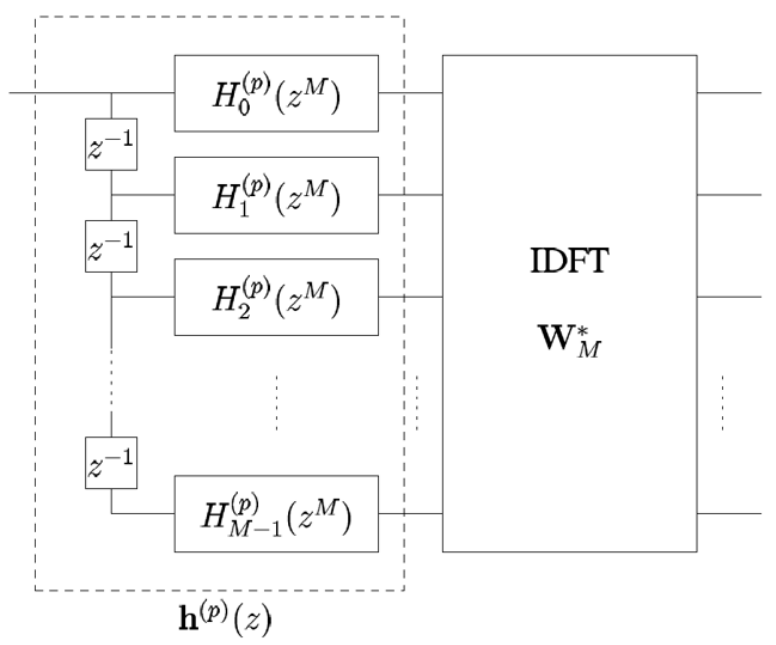
1. Discrete Fourier Transform



- In previous lectures, DFT seen in various contexts
  - Implementing convolution/correlation in frequency domain
  - Analysis based on spectral features. E.g. psychoacoustic models in audio coding require mapping of audio signal to frequency domain
- Today in implementation of modulated filter banks



- Due to its importance, many fast techniques has been developed for DFT computation
  - Cooley-Tukey algorithm
  - Winograd algorithm
  - etc.



DFT definition

- Convenient way is to define it in terms of matrix multiplication
  - DFT matrix : each element is a rotation
- Matrix for the inverse DFT (IDFT) is constructed similarly
  - Using the coefficient $W_M = e^{j2\pi/M}$
  - As a result, we get complex-conjugate $\mathbf{W}_M^*$
- In definitions, scaling factor can be added
  - If no scaling factors, then

$$\mathbf{x}' = \boldsymbol{W}_M^* \boldsymbol{W}_M \mathbf{x} \quad \text{gives} \quad \mathbf{x}' = M\mathbf{x} \ (M \text{ is scalar})$$

$$\mathbf{y} = \mathbf{W}_M \mathbf{x}$$

Signal samples:

$$\mathbf{x} = [x_0, x_1, \ldots, x_{M-1}]^T$$

Transform coefficients:

$$\mathbf{y} = [y_0, y_1, \ldots, y_{M-1}]^T$$

DFT matrix

$$\mathbf{W}_M = \begin{bmatrix} W_{0,0} & W_{0,1} & \cdots & W_{0,M-1} \\ W_{1,0} & W_{1,1} & \cdots & W_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ W_{M-1,0} & W_{M-1,1} & \cdots & W_{M-1,M-1} \end{bmatrix}$$

$$W_{m,n} = W_M^{mn}$$

$$W_M = e^{-j2\pi/M} = \cos(2\pi/M) - j\sin(2\pi/M)$$

Cooley-Tukey algorithm

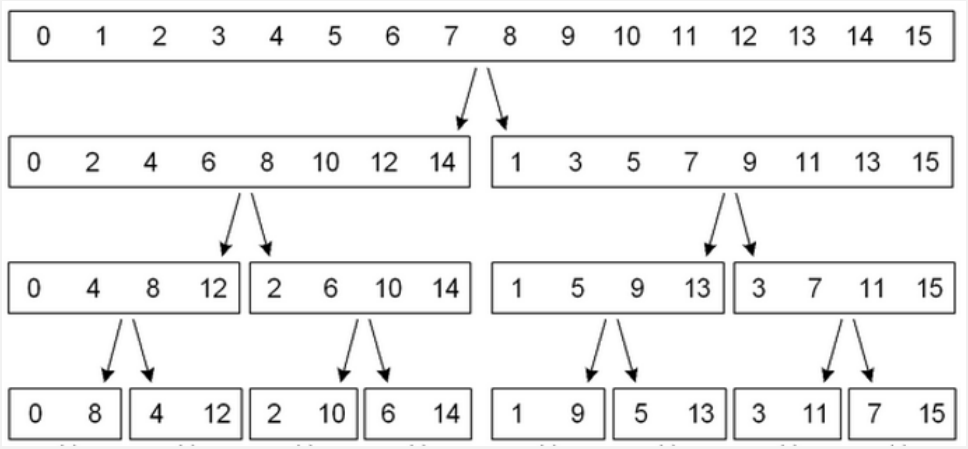
- Invented by Gauss (1805), popularized by Cooley and Tukey (1965)

- Radix-N recursivity
  - Radix-2: M-point FFT is computed using two M/2-point FFTs
  - Radix-4: M-point FFT is computed using four M/4-point FFTs

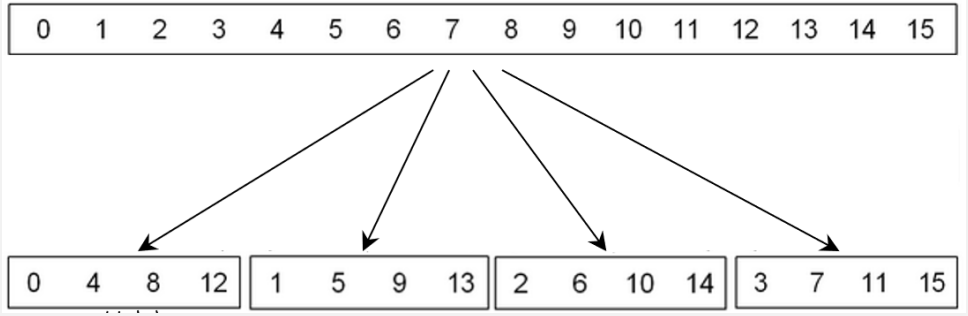
*Example (radix-4). 64 point FFT can be computed using **four** 16-point FFTs, which can be computed using **four** 4-point FFTs*

- Reordering of either input or output samples (decimation)
  - Decimation-in-time (DIT): input samples
  - Decimation-in-frequency (DIF): output samples

Radix-2 decimation of 16 samples



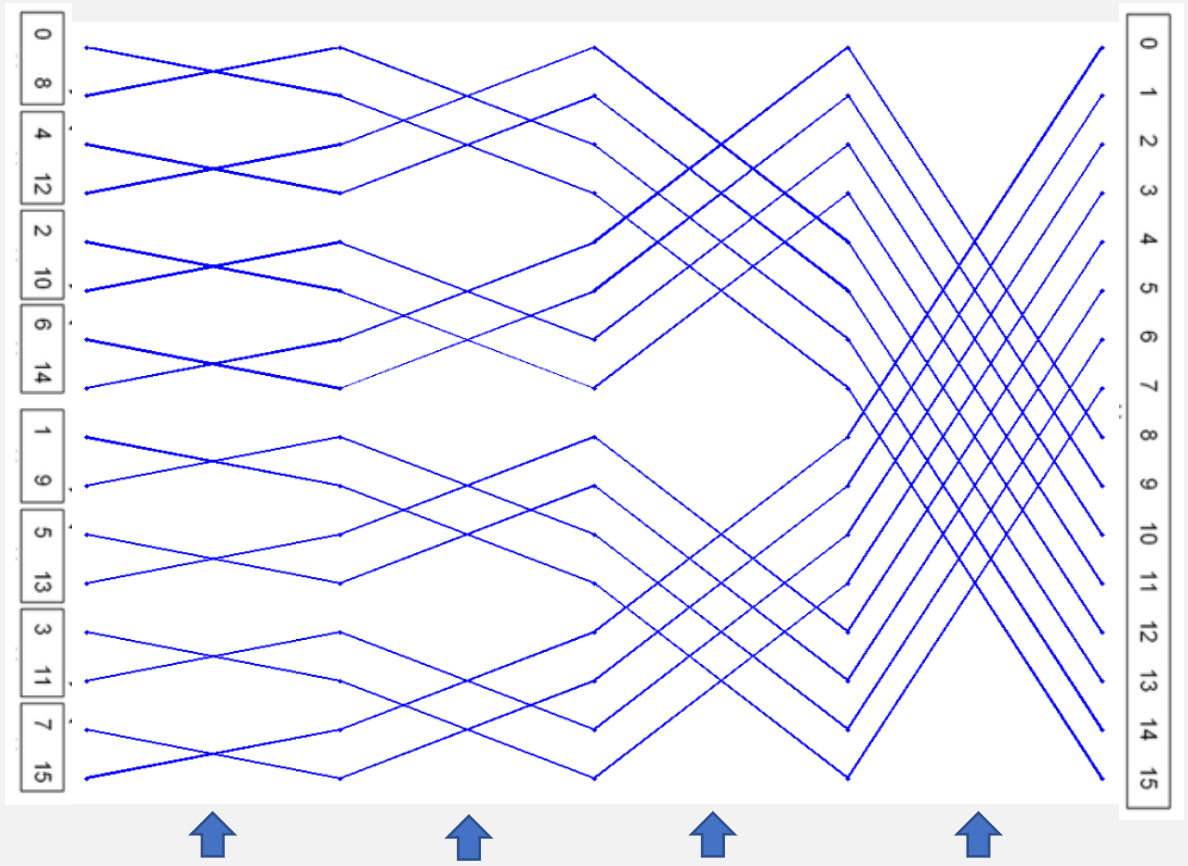
Radix-4 decimation of 16 samples



Sketch of signal flows for 16-point FFT (1)

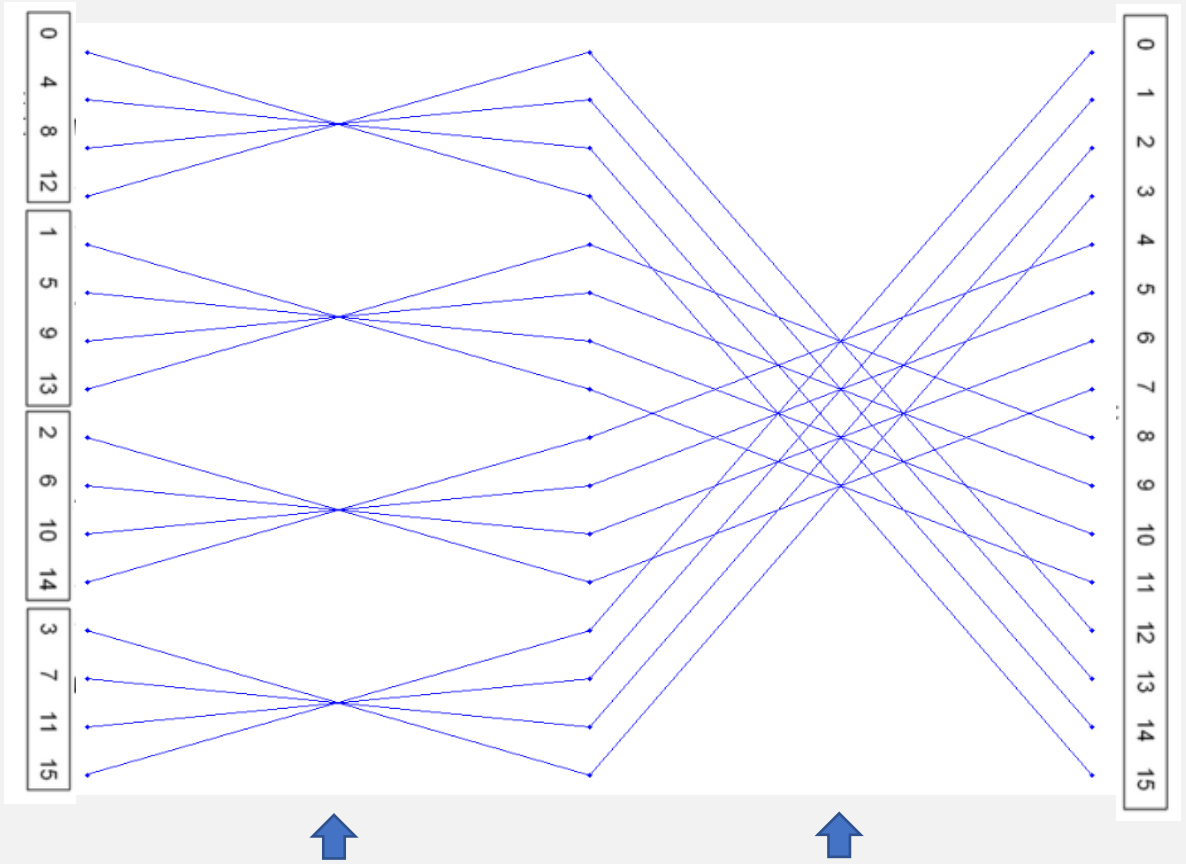
Decimation-in-time

Radix 2



Computations in these crossings

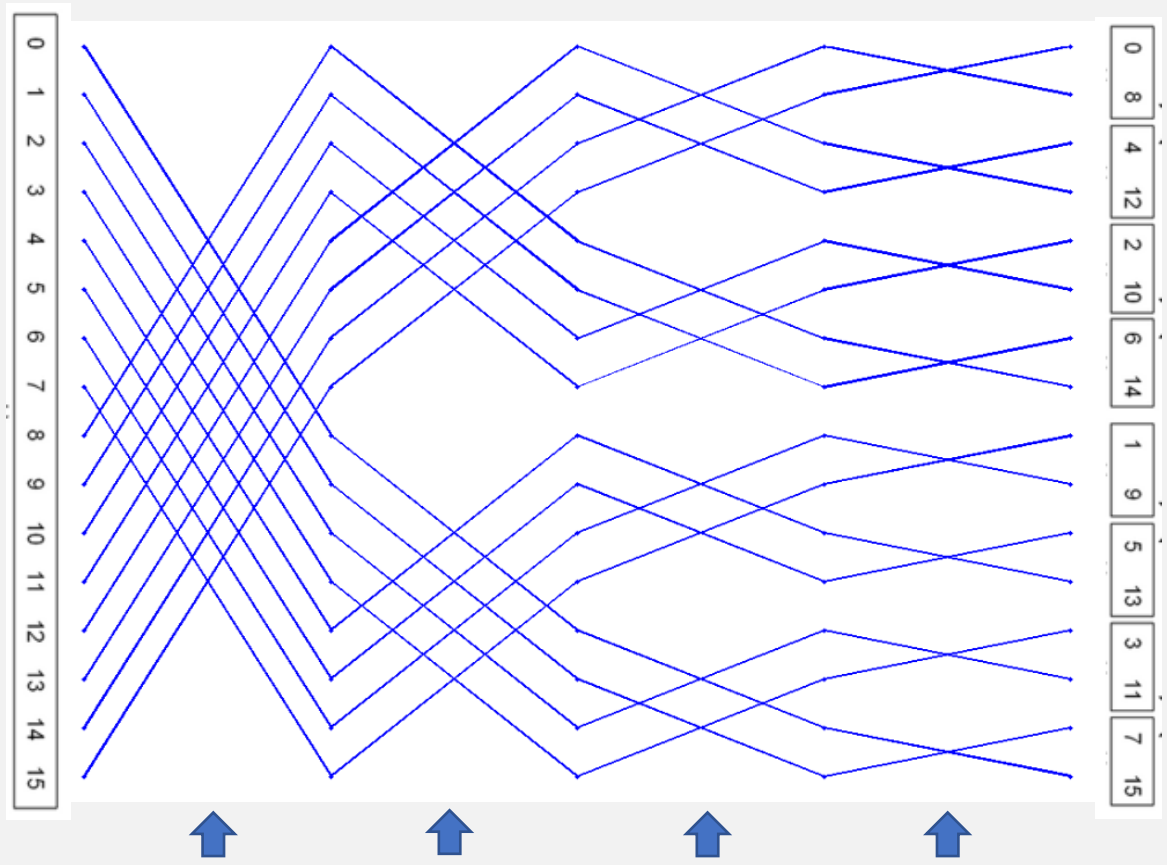
Radix 4



Sketch of signal flows for 16-point FFT (2)

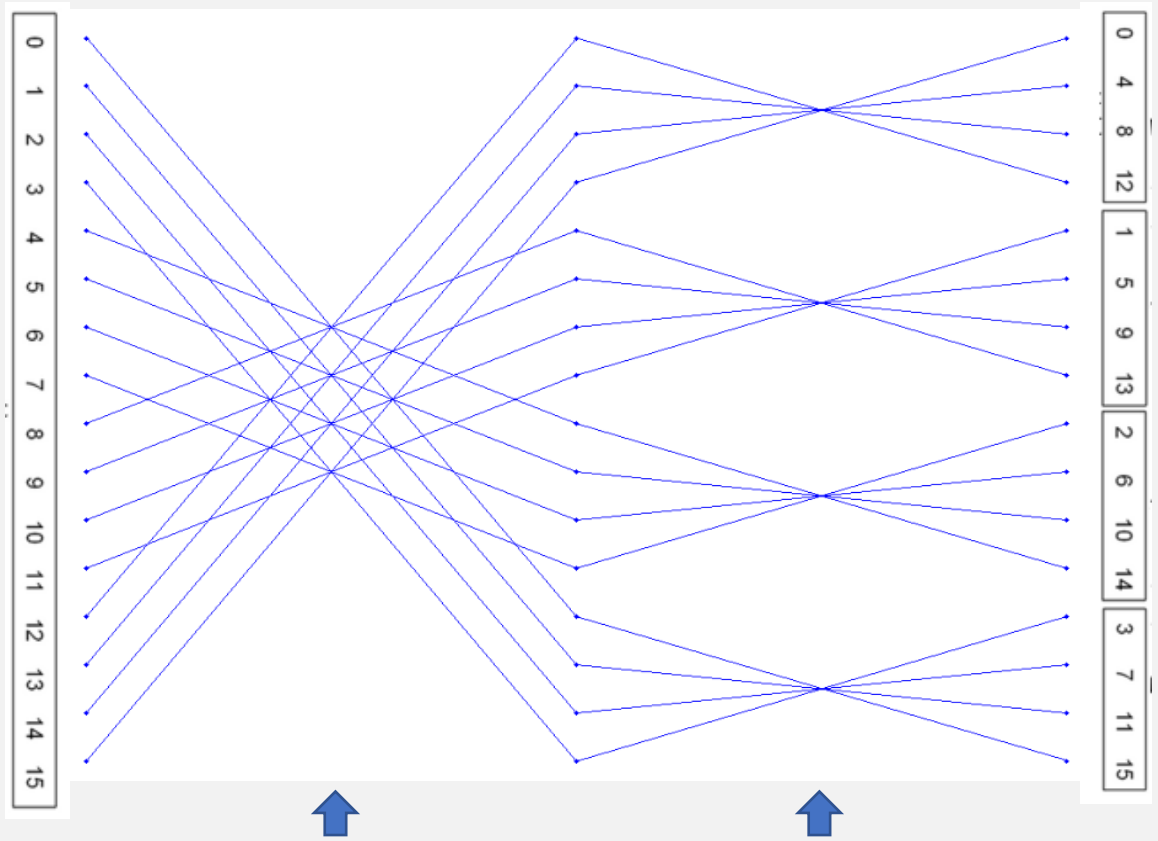
Decimation-in-frequency

Radix 2



Computations in these crossings

Radix 4



Derivation of radix-2 DIT FFT (1)

1. Some redundancies can be pointed out in the DFT matrix. These redundancies are exploited in the steps 2 and 4.

$$\mathbf{W}_M = \begin{bmatrix} W_{0,0} & W_{0,1} & \cdots & W_{0,M-1} \\ W_{1,0} & W_{1,1} & \cdots & W_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ W_{M-1,0} & W_{M-1,1} & \cdots & W_{M-1,M-1} \end{bmatrix}$$

2. Two subsequences are extracted from the input vector, one for even indices and the second for odd indices.

Corresponding column vector elements of the DFT matrix are extracted.

Based on the first redundancy, we can express the multipliers $W_M^{m(2n')}$ as $W_{M/2}^{mn'}$. Similar summation is obtained for even and odd subsequences.

1. Redundancies. The elements of the matrix $\mathbf{W}_M$ are powers of $W_M = e^{-j2\pi/M}$, and some redundancies can be pointed out. First,

$$W_M^2 = W_{M/2} \tag{4}$$

as $W_M^2 = (e^{-j2\pi/M})^2 = e^{-j4\pi/M} = e^{-j2\pi/(M/2)} = W_{M/2}$. Second,

$$W_M^{k+M/2} = -W_M^k \tag{5}$$

as

$$W_M^{k+M/2} = W_M^k W_M^{M/2} = W_M^k (e^{-j2\pi/M})^{M/2} = W_M^k e^{-j\pi}, \text{ and } e^{-j\pi} = -1.$$

2. Splitting. We can split the computation of y_m as

$$\begin{aligned} y_m &= \sum_{n=0}^{M-1} W_M^{mn} x_n \\ &= \underbrace{\sum_{n'=0}^{M/2-1} W_M^{m(2n')} x_{2n'}}_{\text{even sequence}} + \underbrace{\sum_{n'=0}^{M/2-1} W_M^{m(2n'+1)} x_{2n'+1}}_{\text{odd sequence}} \\ &= \sum_{n'=0}^{M/2-1} W_{M/2}^{mn'} x_{2n'} + W_M^m \sum_{n'=0}^{M/2-1} W_{M/2}^{mn'} x_{2n'+1}, \end{aligned} \tag{6}$$

where Eq. 4 has been used in the last step. The even and odd-indexed elements of $\mathbf{x}$ are represented by the vectors $\mathbf{x}_{\text{even}} = [x_0, x_2, \ldots, x_{M-2}]^T$ and $\mathbf{x}_{\text{odd}} = [x_1, x_3, \ldots, x_{M-1}]^T$.

Derivation of radix-2 DIT FFT (2)

Input samples from even and odd subsequences are put into corresponding vectors $\boldsymbol{x}_{\text{even}}$ and $\boldsymbol{x}_{\text{odd}}$.

It is noted that to compute the first terms of the output, ”upper part”, we must multiply $\boldsymbol{x}_{\text{even}}$ and $\boldsymbol{x}_{\text{odd}}$ by the DFT matrix for M/2 points.

The latter result is multiplied by a diagonal matrix and The sum of the results provides the ”upper part”.

For the last terms of the output, ”lower part”, it is found out that same multiplications for $\boldsymbol{x}_{\text{even}}$ and $\boldsymbol{x}_{\text{odd}}$ are needed.

The only difference is that subtraction is performed for the results, not addition.

3. Computing the upper part of $\mathbf{y}$. Let $\mathbf{y}_{0:(M/2-1)} = [y_0, y_1, \ldots, y_{M/2-1}]^T$. According to splitting, we get

$$\mathbf{y}_{0:(M/2-1)} = \underbrace{\mathbf{W}_{M/2}\mathbf{x}_{\text{even}}}_{M/2-\text{point DFT}} + \text{diag}(W_M^0, W_M^1, \ldots, W_M^{M/2-1}) \underbrace{\mathbf{W}_{M/2}\mathbf{x}_{\text{odd}}}_{M/2-\text{point DFT}}, \quad (7)$$

where $\text{diag}(\cdot)$ denotes a diagonal matrix composed of the elements within the parentheses. We observe that M-point DFT can be computed from $M/2$-point DFT results. The Cooley-Tukey algorithm is based on this idea.

4. Computing the lower part of $\mathbf{y}$. Let $\mathbf{y}_{(M/2):(M-1)} = [y_{M/2}, y_1, \ldots, y_{M-1}]^T$. We note first that

$$W_{M/2}^{mn'} = W_{M/2}^{(m-M/2+M/2)n'} = W_{M/2}^{(m-M/2)n'} W_{M/2}^{(M/2)n'} = W_{M/2}^{m'n'},$$

where $m' = m - M/2$ (i.e. $m' \in \{0, 1, ..., M/2-1\}$). So, the summations of Eq. 6 are the same for *both* the upper and lower parts of $\mathbf{y}$. We get

$$\begin{aligned} \mathbf{y}_{(M/2):(M-1)} &= \mathbf{W}_{M/2}\mathbf{x}_{\text{even}} + \text{diag}(W_M^{M/2}, W_M^{M/2+1}, \ldots, W_M^{M-1})\mathbf{W}_{M/2}\mathbf{x}_{\text{odd}} \\ &= \underbrace{\mathbf{W}_{M/2}\mathbf{x}_{\text{even}}}_{M/2-\text{point DFT}} - \underbrace{\text{diag}(W_M^0, W_M^1, \ldots, W_M^{M/2-1})}_{\text{appears also in Eq. 7}} \underbrace{\mathbf{W}_{M/2}\mathbf{x}_{\text{odd}}}_{M/2-\text{point DFT}}, \quad (8) \end{aligned}$$

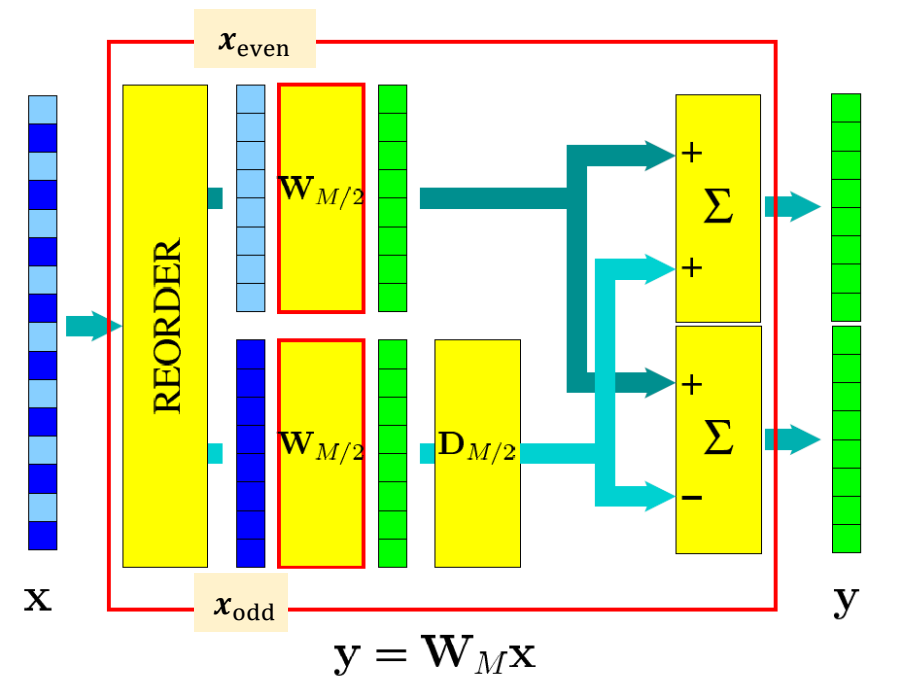
where the second equality follows from Eq. 5.

Derivation of radix-2 DIT FFT (3)

The results are collected into a matrix-vector expression.

This expression is then rewritten in a form which shows how M-point DFT can be computed from M/2-point DFT results.

We get a building block for recursive composition of FFTs.



5. Putting together. Combining the results, we have

$$\mathbf{y} = \begin{bmatrix} \mathbf{W}_{M/2} & \mathbf{D}_{M/2}\mathbf{W}_{M/2} \\ \mathbf{W}_{M/2} & -\mathbf{D}_{M/2}\mathbf{W}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\text{even}} \\ \mathbf{x}_{\text{odd}} \end{bmatrix}, \tag{9}$$

where $\mathbf{D}_{M/2} = \text{diag}(W_M^0, W_M^1, \ldots, W_M^{M/2-1})$. We can rewrite the equation as

$$\mathbf{y} = \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{I}_{M/2} \\ \mathbf{I}_{M/2} & -\mathbf{I}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{W}_{M/2}\mathbf{x}_{\text{even}} \\ \mathbf{D}_{M/2}\mathbf{W}_{M/2}\mathbf{x}_{\text{odd}} \end{bmatrix}, \tag{10}$$

where $\mathbf{I}_{M/2}$ denotes the $M/2 \times M/2$ identity matrix. To derive the computational structure for M-point DFT, we read this equation from right to left:

1. Data in $\mathbf{x}$ is reordered, which gives $\mathbf{x}_{\text{even}}$ and $\mathbf{x}_{\text{odd}}$.
2. M/2-point DFTs are computed for those sample vectors (multiplications by $\mathbf{W}_{M/2}$).
3. The second DFT output vector is multiplied by $\mathbf{D}_{M/2}$. As $\mathbf{D}_{M/2}$ is a diagonal matrix, the matrix multiplication corresponds to elementwise multiplication of the diagonal and the DFT output (total of $M/2$ multiplications).
4. Addition/subtraction of the vectors gives the output $\mathbf{y}$.

NUMERICAL EXAMPLES OF PREVIOUS EQUATION

Equation 9:

$$\mathbf{y} = \begin{bmatrix} \mathbf{W}_{M/2} & \mathbf{D}_{M/2}\mathbf{W}_{M/2} \\ \mathbf{W}_{M/2} & -\mathbf{D}_{M/2}\mathbf{W}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\text{even}} \\ \mathbf{x}_{\text{odd}} \end{bmatrix}$$

Check that corresponds to $\mathbf{y} = \mathbf{W}_M\mathbf{x}$

• $M = 2$: Because $W_2 = e^{-j\pi} = -1$,

$$\mathbf{W}_2 = \begin{bmatrix} W_2^0 & W_2^0 \\ W_2^0 & W_2^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

As $\mathbf{D}_1 = \text{diag}(W_2^0) = [1]$, and $\mathbf{W}_1 = [W_1^0] = [1]$, the Eq. 9 gives

$$\mathbf{y} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}.$$

$$y = \begin{bmatrix} [1] & [1][1] \\ [1] & [-1][1] \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

• $M = 4$: As $W_4 = e^{-j\pi/2} = -j$,

$$\mathbf{W}_4 = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}.$$

As $\mathbf{D}_2 = \text{diag}(W_4^0, W_4^1) = \begin{bmatrix} 1 & 0 \\ 0 & -j \end{bmatrix}$, the Eq. 9 gives

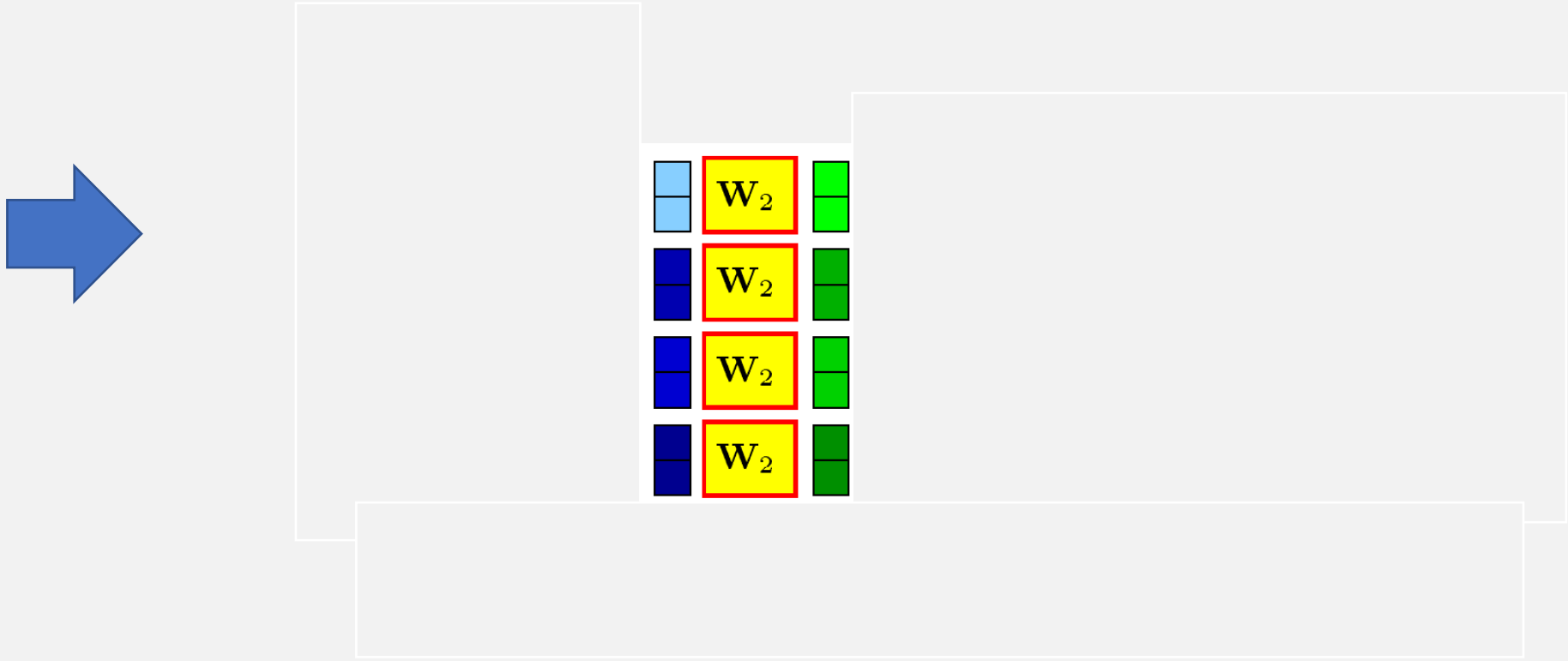
$$\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -j & j \\ 1 & 1 & -1 & -1 \\ 1 & -1 & j & -j \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}.$$

$$y = \left[\begin{array}{cc|cc} [1 & 1] & [1 & 0][1 & 1] \\ [1 & -1] & [0 & -j][1 & -1] \\ \hline [1 & 1] & [-1 & 0][1 & 1] \\ [1 & -1] & [0 & j][1 & -1] \end{array}\right] \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}$$

Note how the 4×4 matrix matches with the matrix $\mathbf{W}_4$: the elements have just been reordered. ■

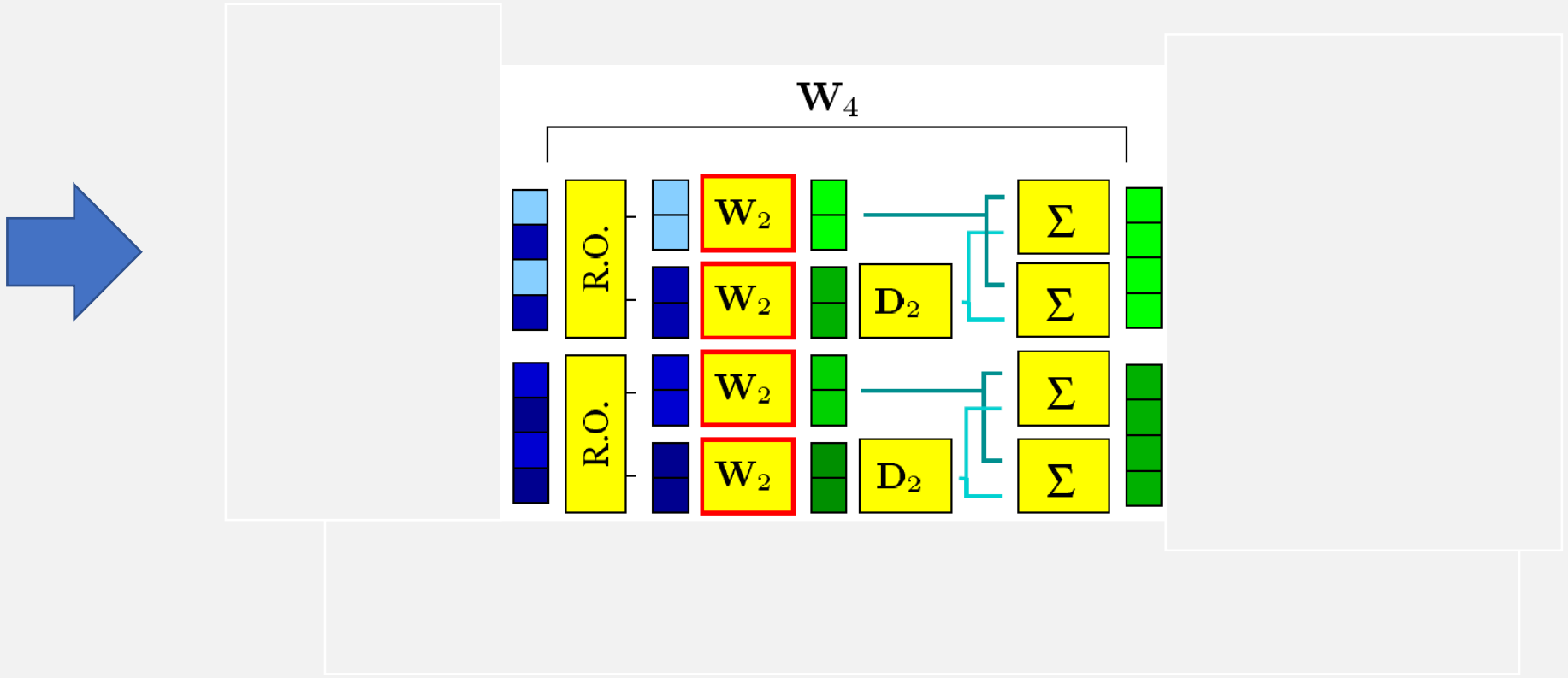
Example. Composing 8-point FFT

- An atomic block for composition is 2-point FFT ($\mathbf{W}_2$). Four needed



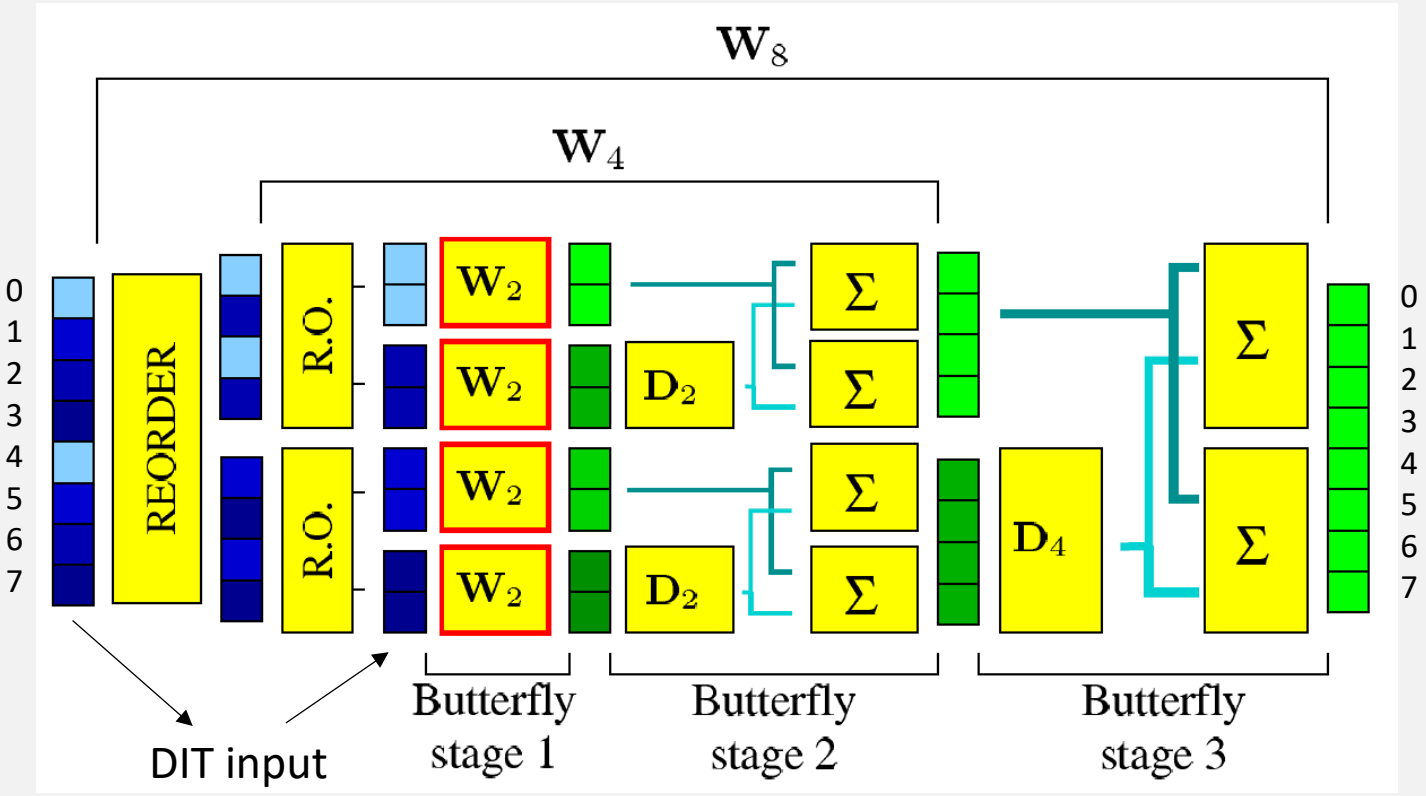
Example. Composing 8-point FFT

- An atomic block for composition is 2-point FFT (**W_2**). Four needed
- Two 4-point FFTs (**W_4**) are built around those four 2-point FFTs

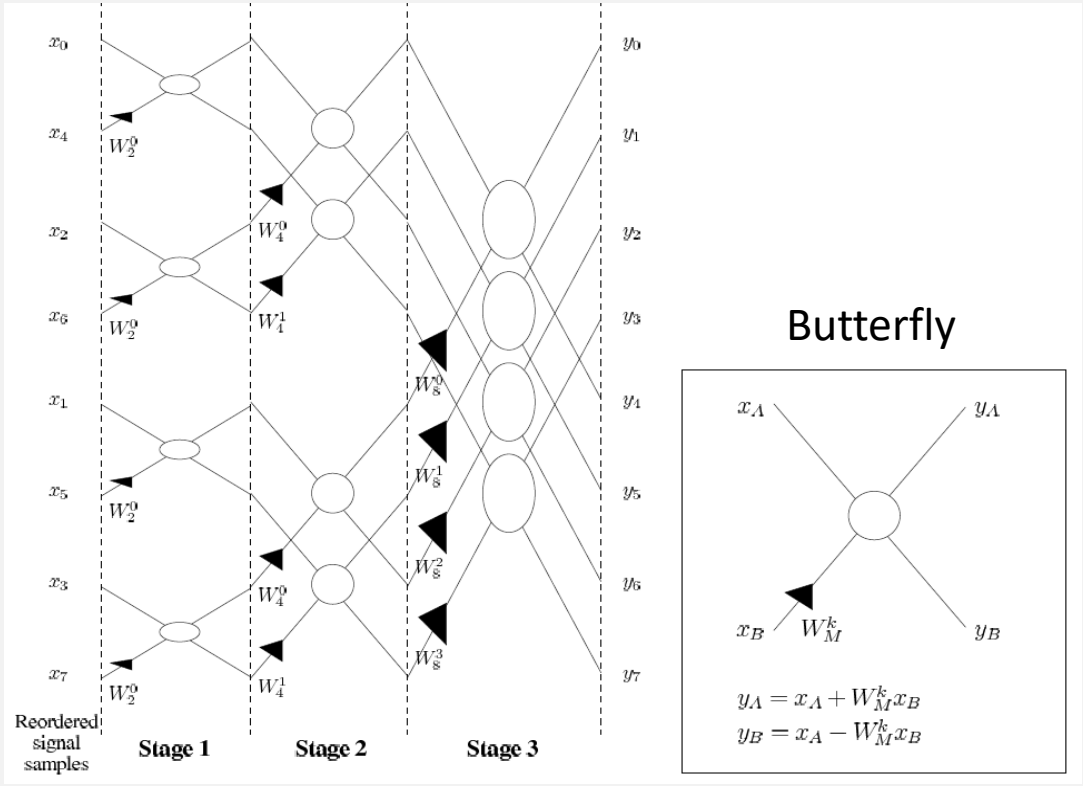


Example. Composing 8-point FFT

- An atomic block for composition is 2-point FFT (W_2). Four needed
- Two 4-point FFTs (W_4) are built around those four 2-point FFTs
- 8-point FFT (W_8) is built around those two 4-point FFTs

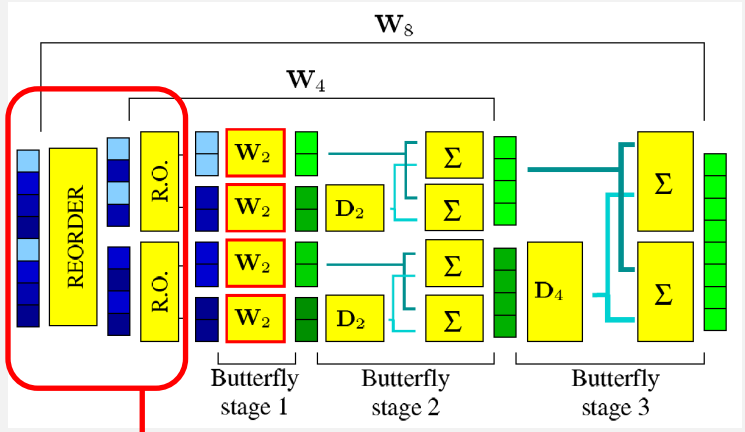


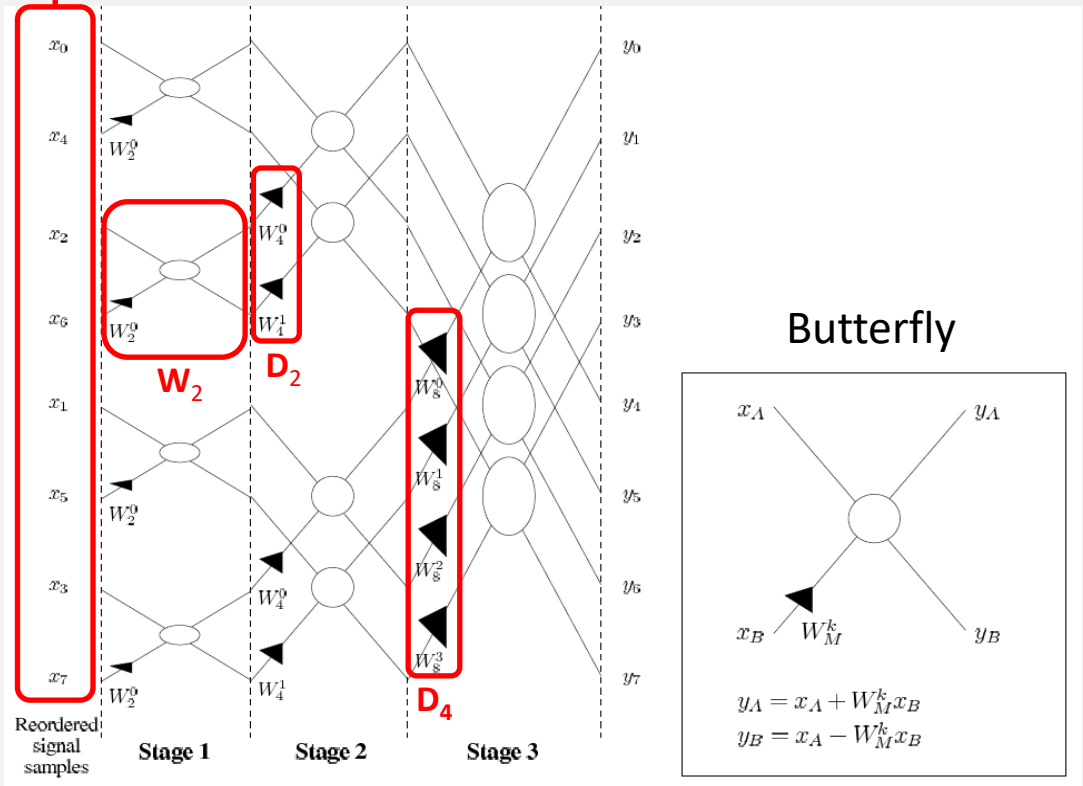
Butterfly graph for 8-point FFT

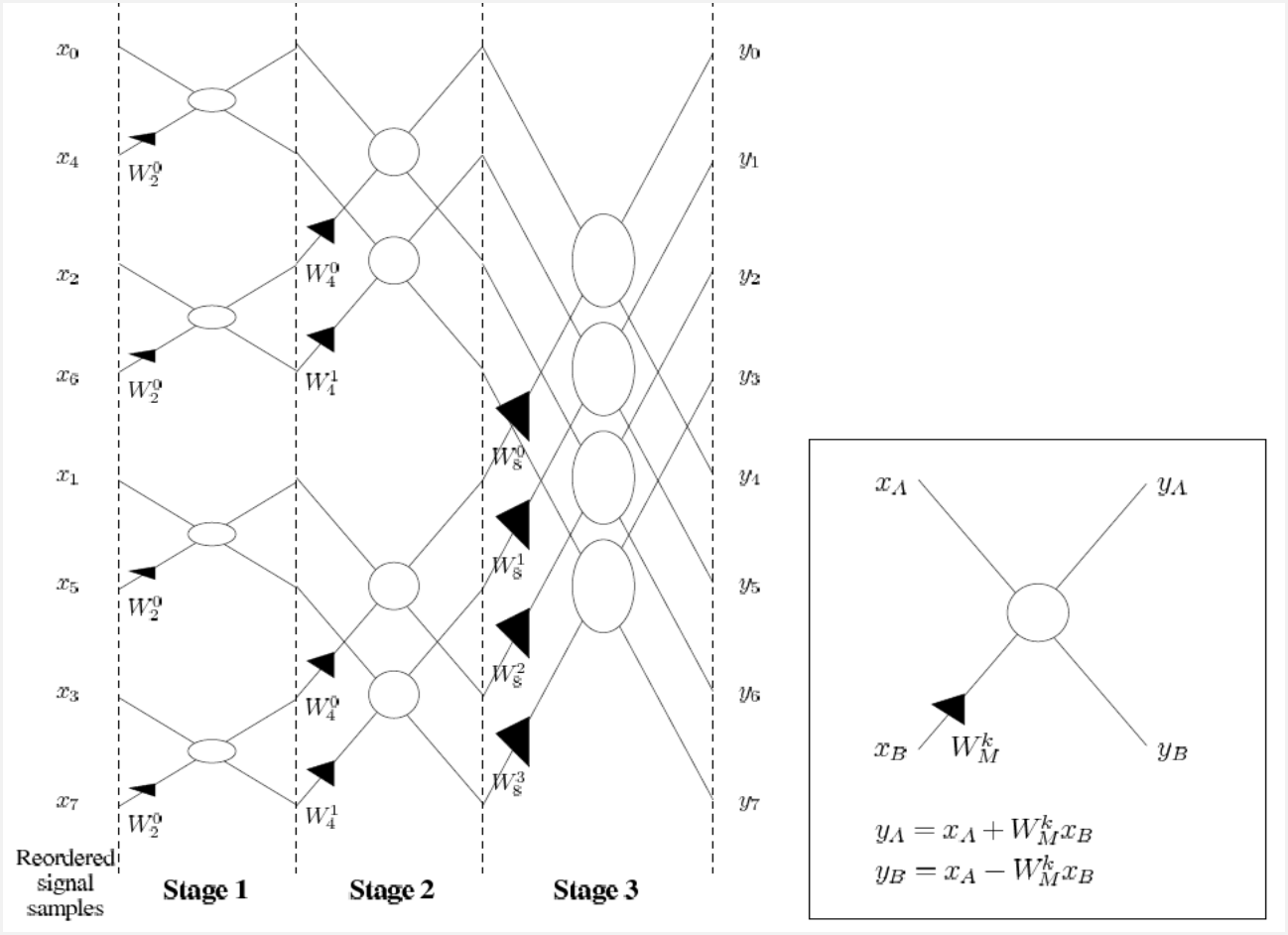


Butterfly graph for 8-point FFT

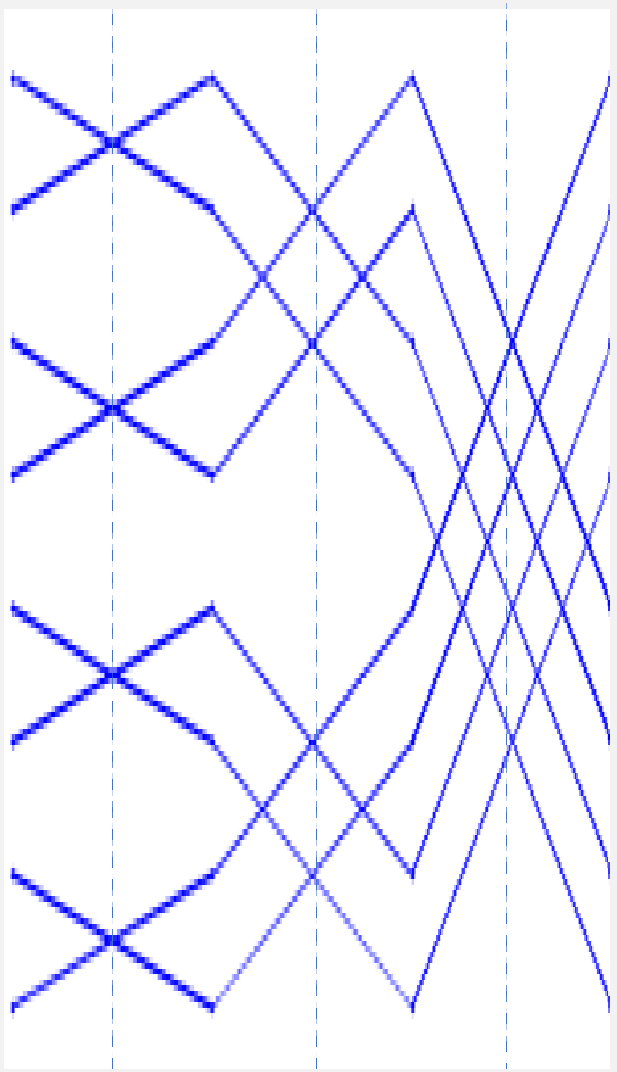
- Reordering steps show up as reordering of input samples
  - Decimation-in-time
- Butterfly stage 1 implements **W_2** operations
- Butterfly stages 1 & 2 (with R.O.) implement **W_4** operations
- Butterfly stages 1, 2 & 3 (+ reordering) implement **W_8** operations





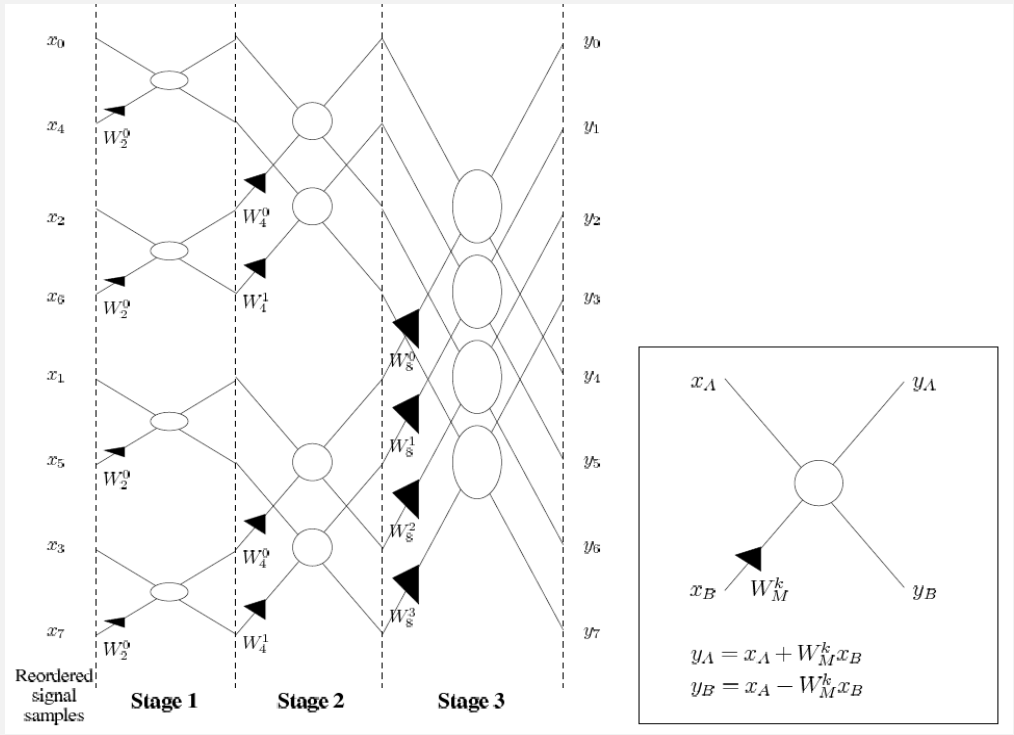


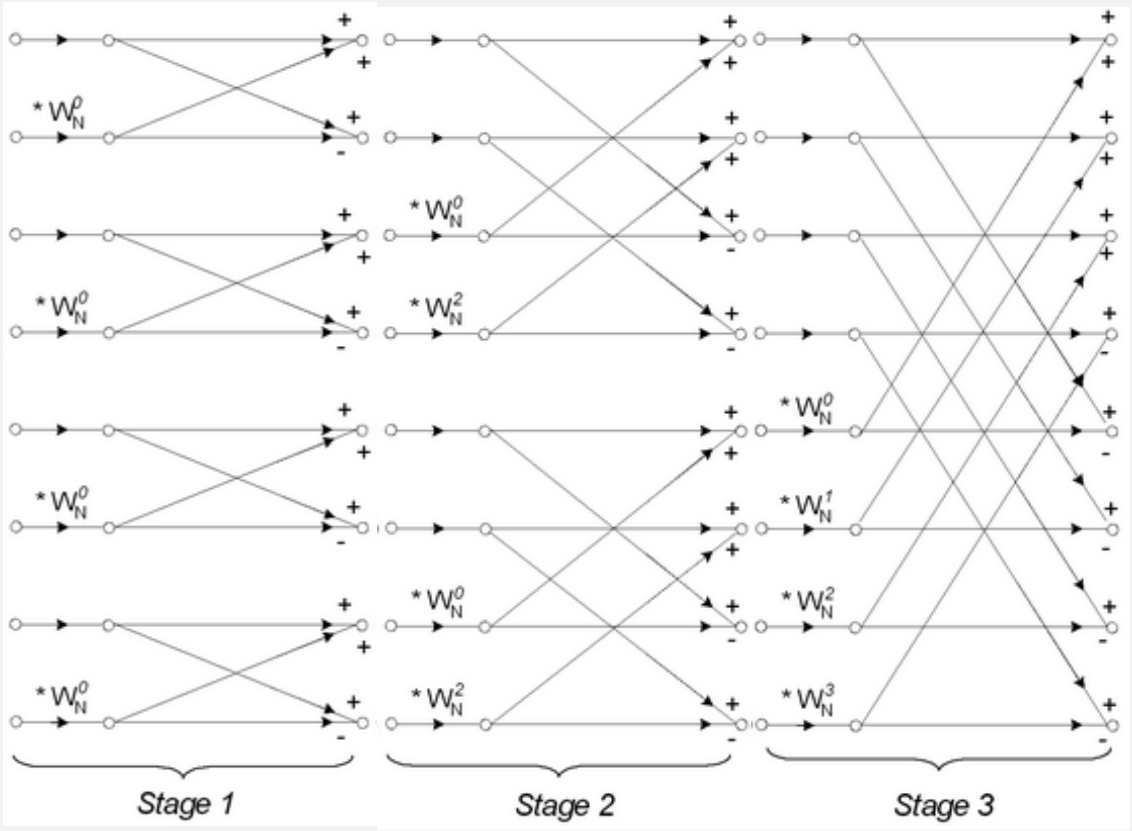
Sketched in previous slides



Note. Another way of drawing the graph

In signal flow graph on the right below, additions and subtractions are represented in a different manner.





Computing operation counts

CHARACTERISTICS OF IMPLEMENTED OPERATION			NUMBER OF **ADD** AND **MPLY** OPERATIONS	
Operation	**Input 1**	**Input 2**	**ADDs**	**MPLYs**
addition	real	real	1	-
multiplication	real	real	-	1
addition	real	complex	1	-
multiplication	real	complex	-	2
addition	complex	complex	2	-
multiplication	complex	complex	2	4
multiplication	complex	complex (fixed)	3	3

For example, if one had FIR filter with 5 complex coefficients and the input signal is real, there would be 5 real-complex multiplications and 4 complex-complex additions. This would require 5 x 2 = 10 MPLYs and 4 x 2 = 8 ADDs.

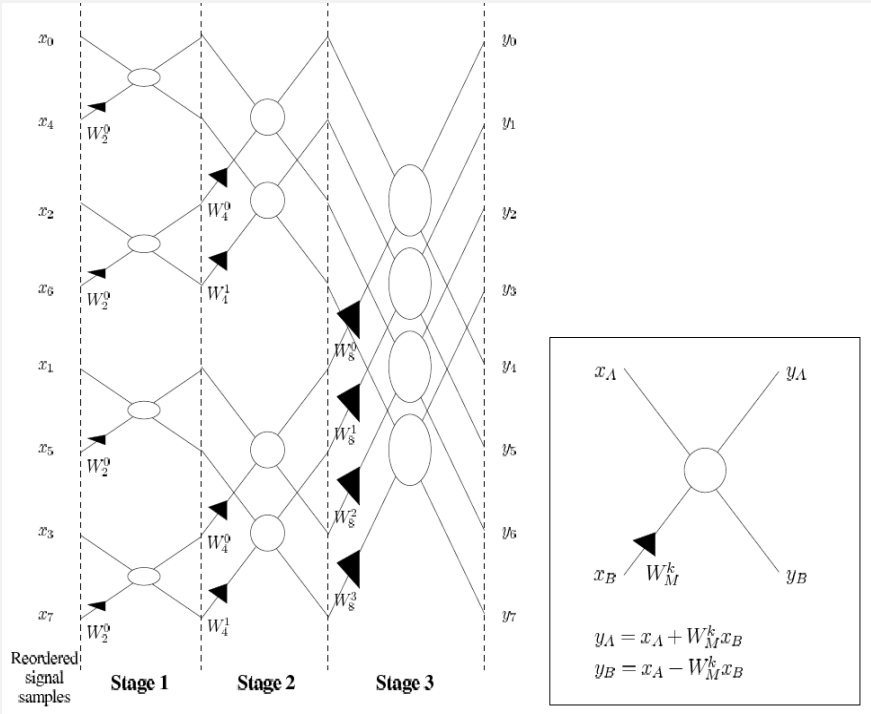
For more information, see **opcounts.pdf**

Computational complexity (8-point FFT)

Analysis for complex-valued input $\mathbf{x} = [x_0, x_1, \ldots, x_{M-1}]^T$

1. For M-point FFT, there are $\log_2 M$ butterfly stages.
2. Each stage contains $M/2$ butterflies.
3. For each butterfly, we must do one complex multiplication by W_M^k, one complex addition, and one complex subtraction.
4. In general, complex multiplication by W_M^k can be implemented by 3 real multiplications and 3 real additions/subtractions.
5. Complex addition (subtraction) can be implemented by 2 real additions (subtractions).

If we consider also subtraction as an addition operation, the implementation of a butterfly requires 3 real multiplications and 7 real additions.



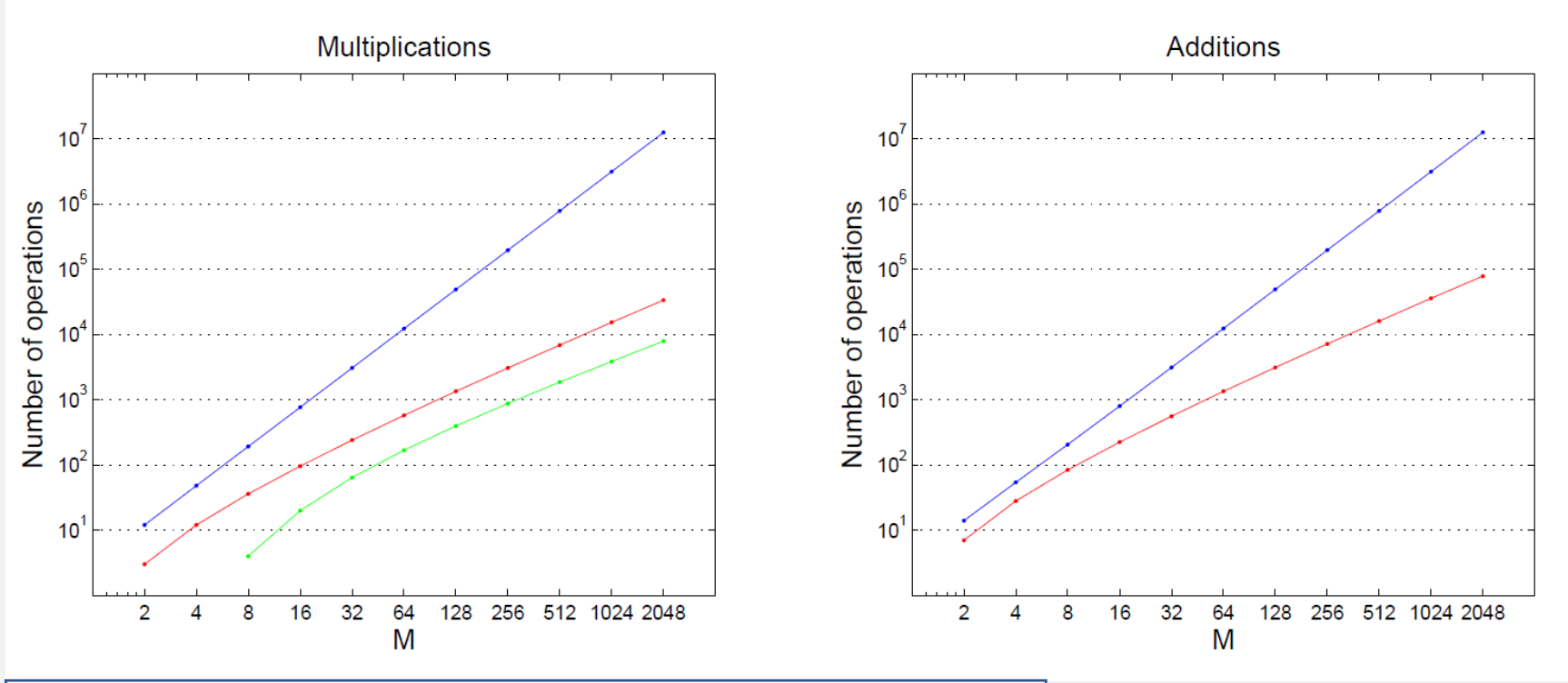
M-point FFT for complex input requires

- $3\left(\frac{M}{2}\right)\log_2 M$ real multiplications (MPLYs)
- $7\left(\frac{M}{2}\right)\log_2 M$ real additions (ADDs)

These formulae give an overestimate of complexity. In practice, even less computations are needed. For example,

- Some complex multiplications are not needed (e.g. first butterfly stage does not need any multiplications)
- If the input is real-valued, we get more simplifications and there are also redundancies in the output coefficients

FFT versus direct matrix-vector multiplication

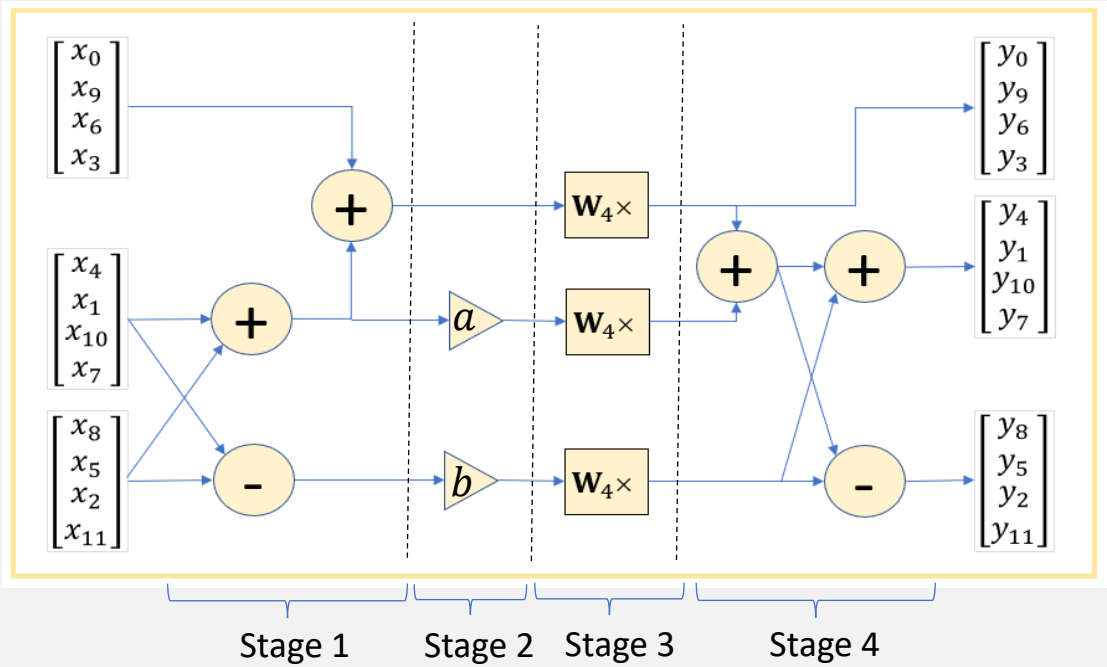


Winograd, S. (1978) On computing the discrete Fourier transform. Math. Computation 32:141–.

Matrix-vector multiplication: $O(M^2)$ complexity
FFT: $O(M \log M)$ complexity

Note. DFT implementation for lengths $N \neq 2^k$

- One can derive efficient structures also for other lengths
- **Example**. 12-point DFT
- Based on Winograd (1978) we can draw the solution shown on the right for 12-point DFT
  - Note interesting ordering of the input and output elements. Based on some number theoretic analysis (see Winograd 1978)
- Complexity analysis for complex inputs (using opcounts.pdf, next slide)
  - **Stage 1**: (3x4 = 12) complex additions = 24 real additions
  - **Stage 2**: (2x4 = 8) real-complex multiplications = 16 real multiplications
  - **Stage 3**: 3 x 4-point DFT. Using additions based on the 4-point DFT matrix without shared calculations. Then, 4x3=12 complex additions per DFT and therefore (3 x 4x3 x 2) = 72 real additions
  - **Stage 4**: (3x4 = 12) complex additions = 24 real additions
  - **TOTAL**: 16 real multiplications, 120 real additions



$$\mathbf{W}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

$$a = \cos 2\pi/3 - 1$$
$$b = i \sin 2\pi/3$$

2. Convolution: in time or frequency domain?

- Let us get some estimates on which approach is better
- Setup:
  - complex-valued input signal
  - filter length = N_h
  - symmetric filter
  - overlap-save based sectioning
  - section length = FFT length = M
- Two parameters: N_h and M

TIME DOMAIN COMPLEXITY

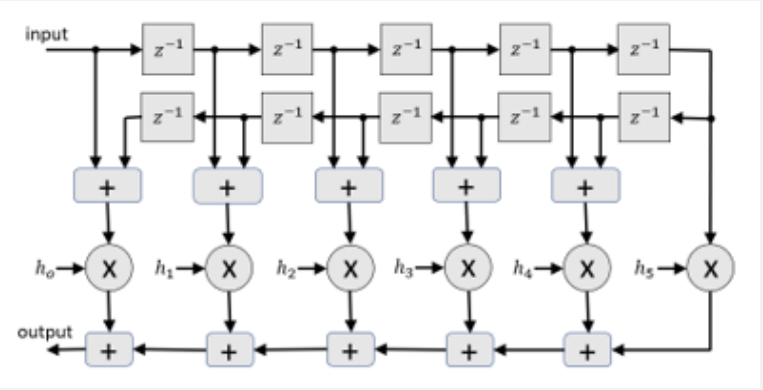
N_h-point FIR filter, real coefficients, complex input, direct implementation:

- $2N_h$ real multiplications
- $2N_h - 2 \approx 2N_h$ real additions

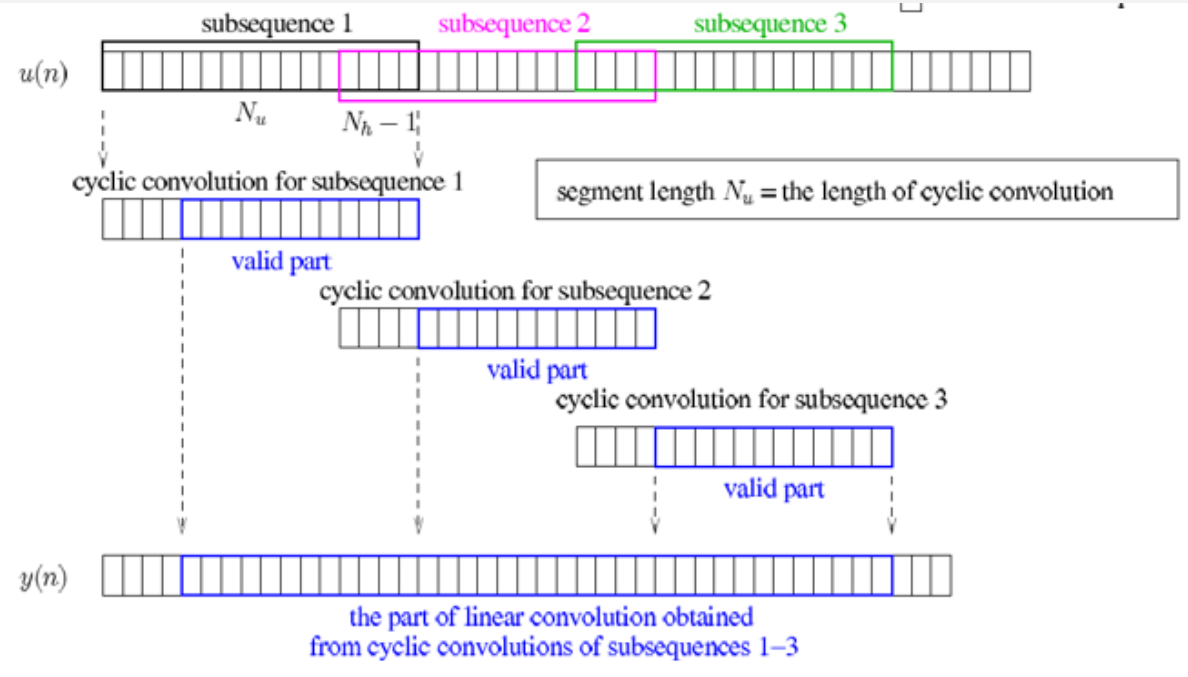
Symmetric filter:

- **$\approx N_h$ multiplications**
- **$\approx 2N_h$ additions** **per input sample**

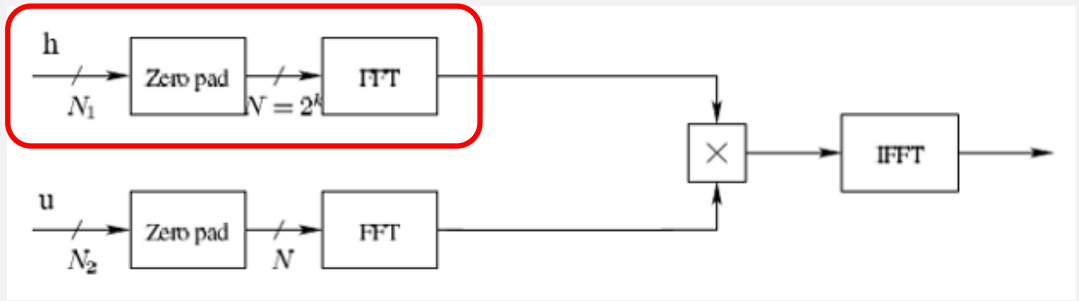
Reduction of multiplications for symmetric case



Complexity: overlap-save + FFT



can be precomputed (as filter coeff.)



Per section: FFT + vector multiplication + IFFT
Section length = FFT length = M
Section overlap = N_h-1
Constraint: $M \geq N_h$
Step between FFT evaluations: $M - N_h + 1$
The number of valid samples per FFT: $M - N_h + 1$

M-point FFT / IFFT for complex input requires

- $3\left(\frac{M}{2}\right)\log_2 M$ real multiplications
- $7\left(\frac{M}{2}\right)\log_2 M$ real additions

FFT, IFFT — FFT output multiplication — overlapped sections

Requires

- $\frac{1}{M-N_h+1}\left[3M + 2\times 3\left(\frac{M}{2}\right)\log_2 M\right]$ real multiplications
- $\frac{1}{M-N_h+1}\left[3M + 2\times 7\left(\frac{M}{2}\right)\log_2 M\right]$ real additions

per input sample

Numbers of operations per input sample

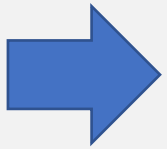
Time domain:

N_h multiplications
$2N_h$ additions

Frequency domain:

$\frac{1}{M-N_h+1}\left[3M + 2\times3\left(\frac{M}{2}\right)\log_2 M\right]$ multiplications
$\frac{1}{M-N_h+1}\left[3M + 2\times7\left(\frac{M}{2}\right)\log_2 M\right]$ additions

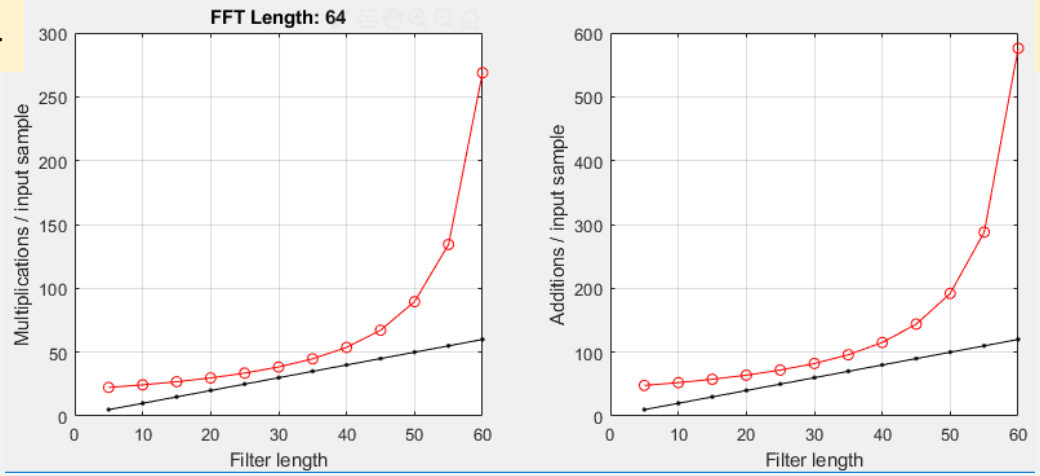
Comparison of complexity with varying filter length N_h and FFT length M



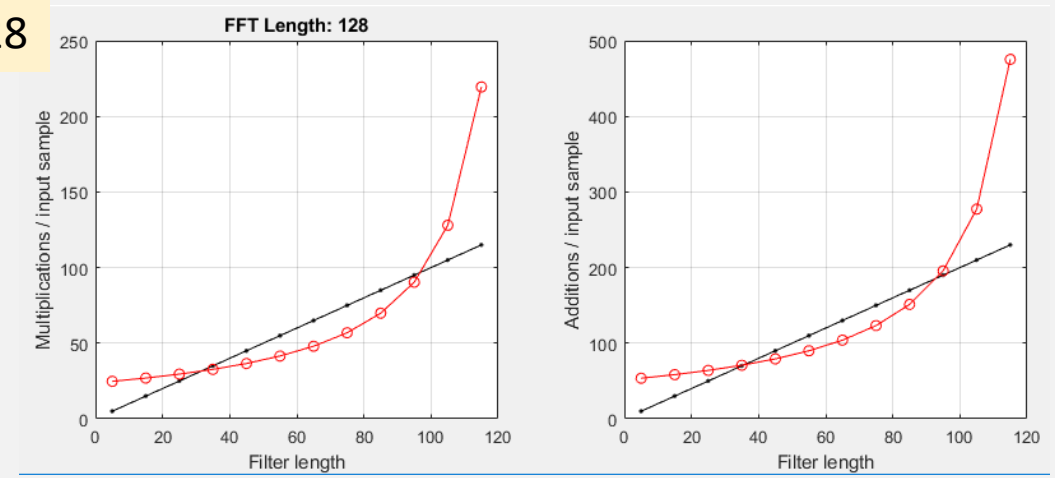
Result

Multiplications on left, additions on right
Black line : time domain, Red curve : frequency domain
X axis = varying filter length

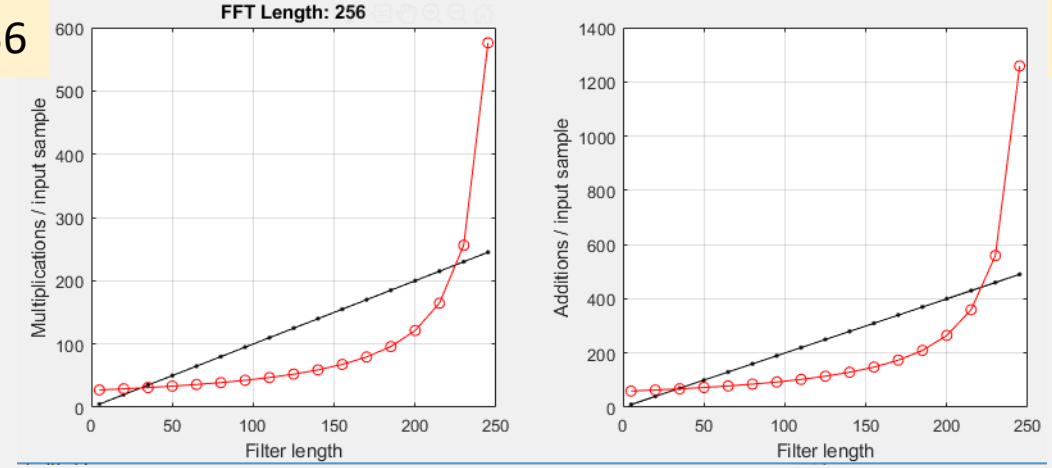
M=64



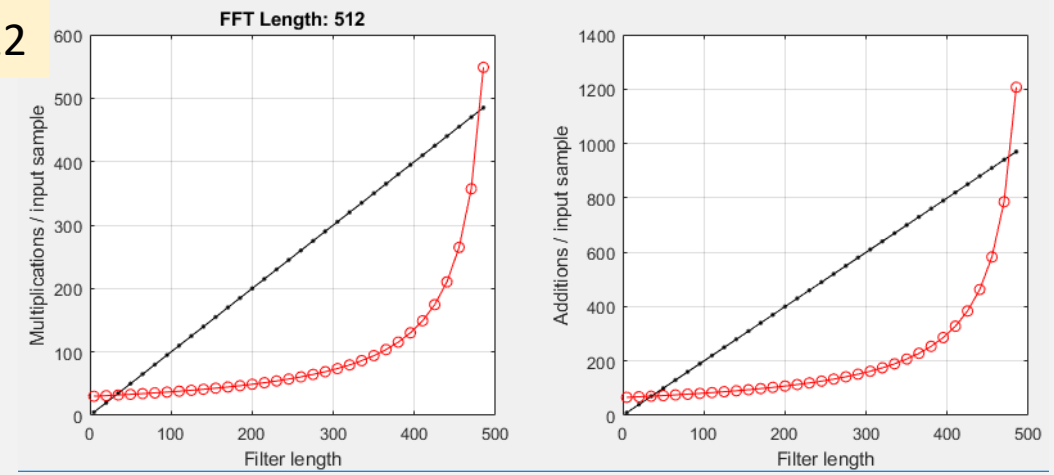
M=128



M=256



M=512



From M = 128 the frequency domain approach starts to have advantage.
Then, the time domain approach is better only for short filter lengths (say, less than 30)
or if FFT length is close to filter length (overlap-save sections are very overlapped).

Frequency domain vs. multirate?

- We did not consider time-domain **multirate** implementation of the filter, which can also give significant advantage in convolution implementation for long *h*

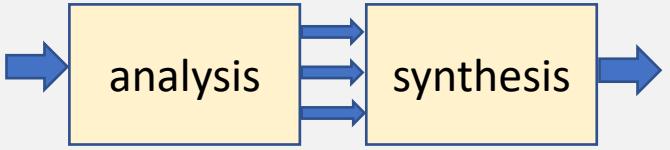
3. Filter banks

A filter bank consists of two or more filters which have directly adjacent frequency bands. There are two kinds of filter banks:

1. **Analysis filter banks** decompose signal spectra in a certain number of frequency bands;

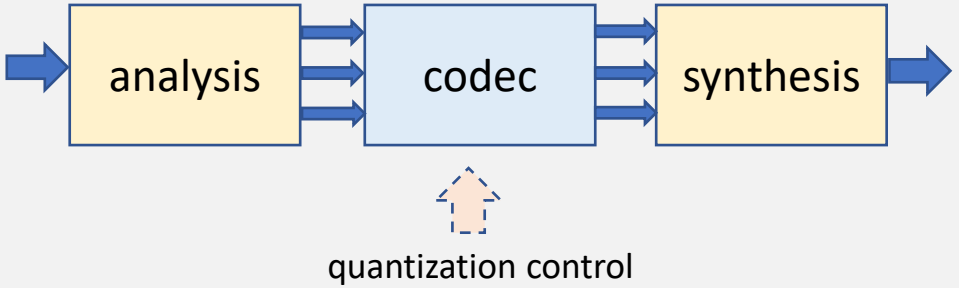
2. **Synthesis filter banks** do the inverse operation reconstructing the signal from the decomposed spectra.

A **subband coding (SBC) filter bank** consist of an analysis filter bank, which is followed by a synthesis filter bank.



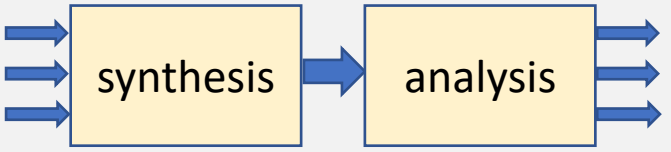
The idea of subband coding

- separate quantization of subband signals
- fine quantization (more bits) for important bands

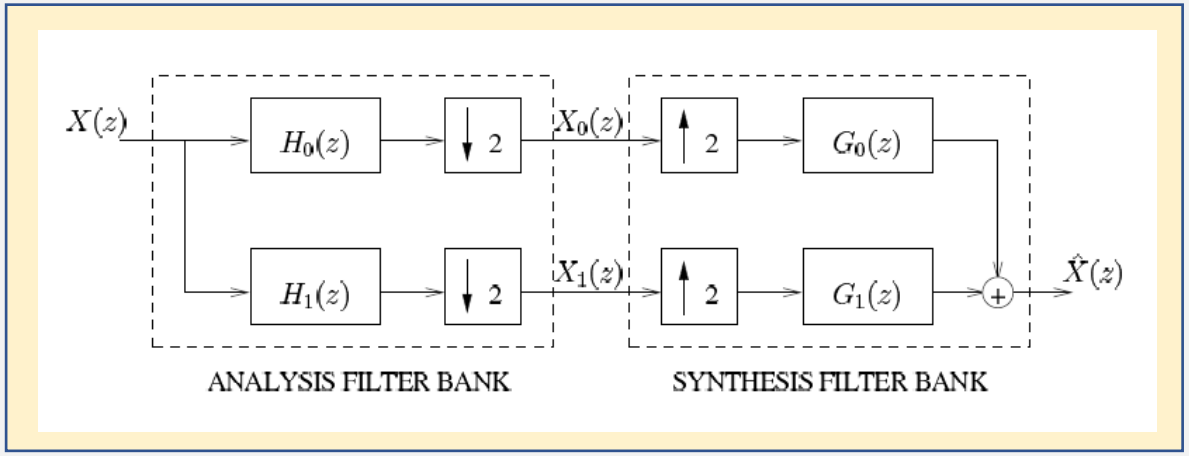


For example, in audio coding psychoacoustic models are used to control quantization.

Transmultiplexer: synthesis followed by analysis (frequency division multiplexing, FDM).



Example. Two-channel SBC filter bank



In the **analysis** filter bank, the input signal is decomposed to two components, which carry low- and high-frequency information, respectively. The filters $H_i(z)$ approximate **anti-alias** filters for decimation by factor 2. Therefore, they can be subsampled in order to maintain the sample rate. This is called **critical sampling**.

In the **synthesis**, the original signal is reconstructed. If alias cancellation is designed carefully, it is even possible to reconstruct the original signal. Then, the SBC filter bank is said to provide **perfect reconstruction**.

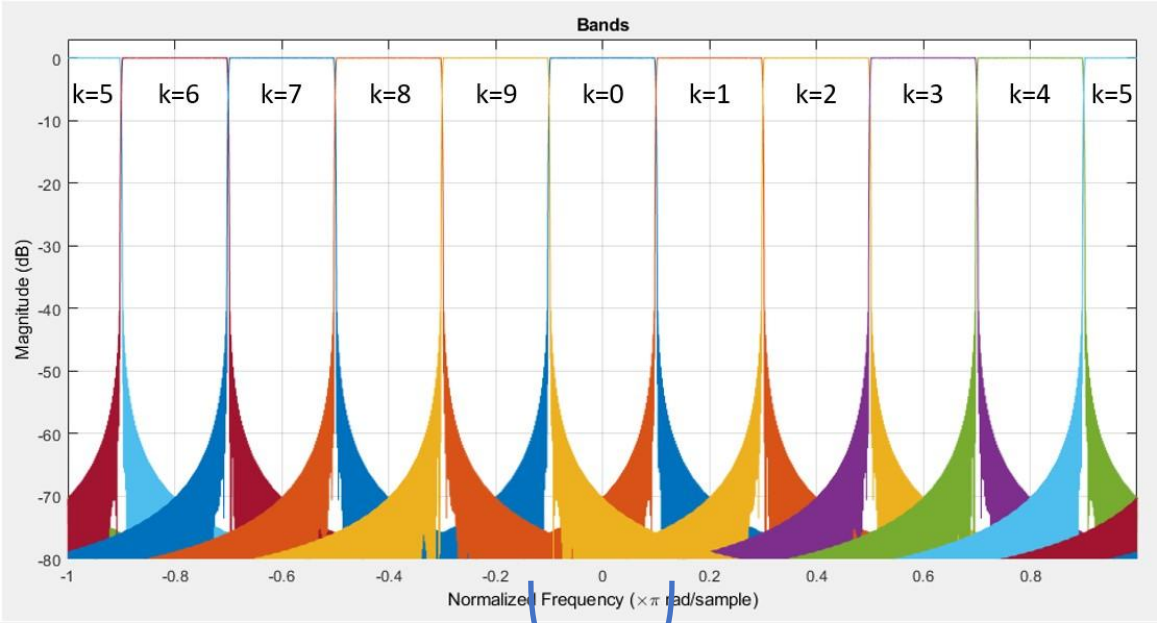
Recall that Nyquist Theorem can be applied to bandpass signals too.

Note: anti-image filter $G_1(z)$ performs modulation in order to provide high-frequency part

Modulated filter banks

- Transfer functions of subband filters can be designed <u>separately</u> in order to reach the goal of perfect reconstruction. However,
  - such design problems can be very complex
  - in addition, computational complexity of the designs can become high, if separate implementation for each subband filter is needed
- Alternative approach is to have a single low-pass **prototype filter** and derive bandpass filters from it **by modulation**. Such designs are called modulated filter banks
  - In general, perfect reconstruction not achievable
  - But leads to computationally efficient designs
  - Let's see how …

An example of a modulated filter bank



Lowpass prototype filter

Modulation of a z-transform

Modulation of a z-transform $X(z)$ (representing a signal or a system) is done by multiplication of the independent variable z with the number W_M^k, where $W_M = \exp(-j2\pi/M)$:

$$X_k^{(m)}(z) = X(zW_M^k).$$

The name "modulation" can be understood by considering the effect of the operation in the time domain. If $X(z)$ is the z-transform of $x(n)$, then $X_k^{(m)}(z)$ is the z-transform of

$$x(n)\cos(2\pi kn/M) + jx(n)\sin(2\pi kn/M).$$

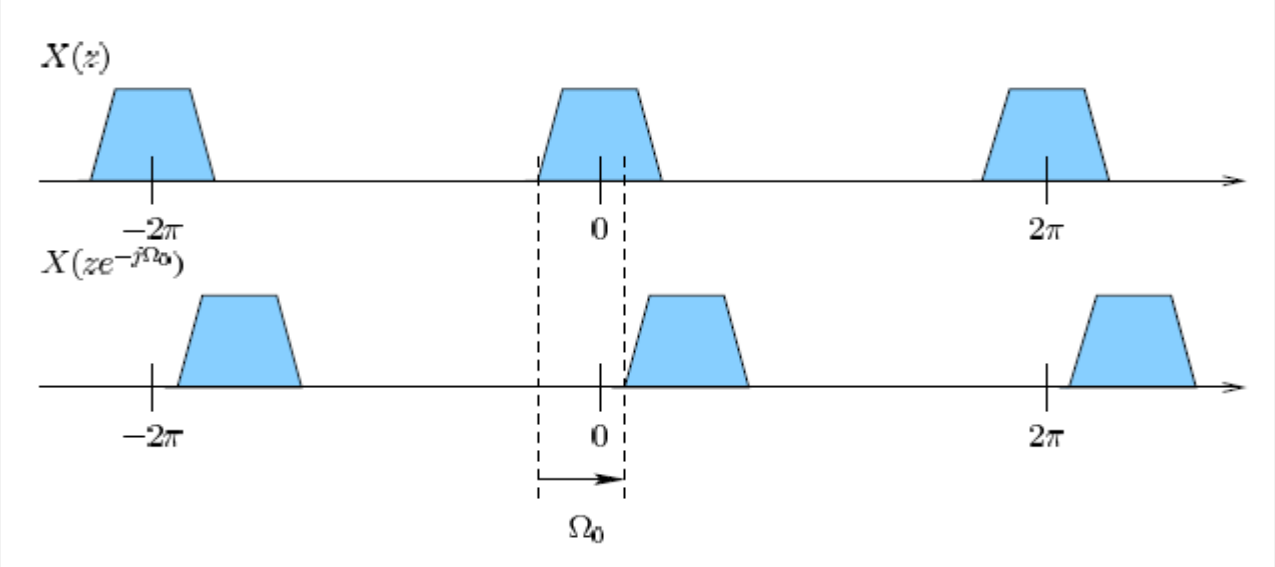
Also, multiplication of the independent variable z changes the signal spectrum

Stacking transforms to a vector

$$\mathbf{x}^{(m)}(z) = \begin{bmatrix} X_0^{(m)}(z) \\ X_1^{(m)}(z) \\ \vdots \\ X_{M-1}^{(m)}(z) \end{bmatrix}$$

Complex modulation

Change of the spectrum, when z is multiplied by $\exp(-j\Omega_0)$, $\Omega_0 = 2\pi k/M$



Example 1. Modulation of a system H(z)

to obtain band filters

$$H(z) = a_0 + a_1 z^{-1} + a_2 z^{-2}$$

$$M = 4 \qquad W_M = e^{-j\pi/2}$$

$H_0^{(m)}(z)$: $W_M^0 = 1$

$$H_0^{(m)}(z) = a_0 + a_1 (z \cdot 1)^{-1} + a_2 (z \cdot 1)^{-2}$$
$$= a_0 + a_1 z^{-1} + a_2 z^{-2}$$

$H_1^{(m)}(z)$: $W_M^1 = e^{-j\pi/2} = -j$

$$H_1^{(m)}(z) = a_0 + a_1 (z \cdot -j)^{-1} + a_2 (z \cdot -j)^{-2}$$
$$= a_0 + j a_1 z^{-1} - a_2 z^{-2}$$

because $(-j)^{-1} = j$

$(-j)^{-2} = -1$

Similarly $H_2^{(m)}(z)$ and $H_3^{(m)}(z)$

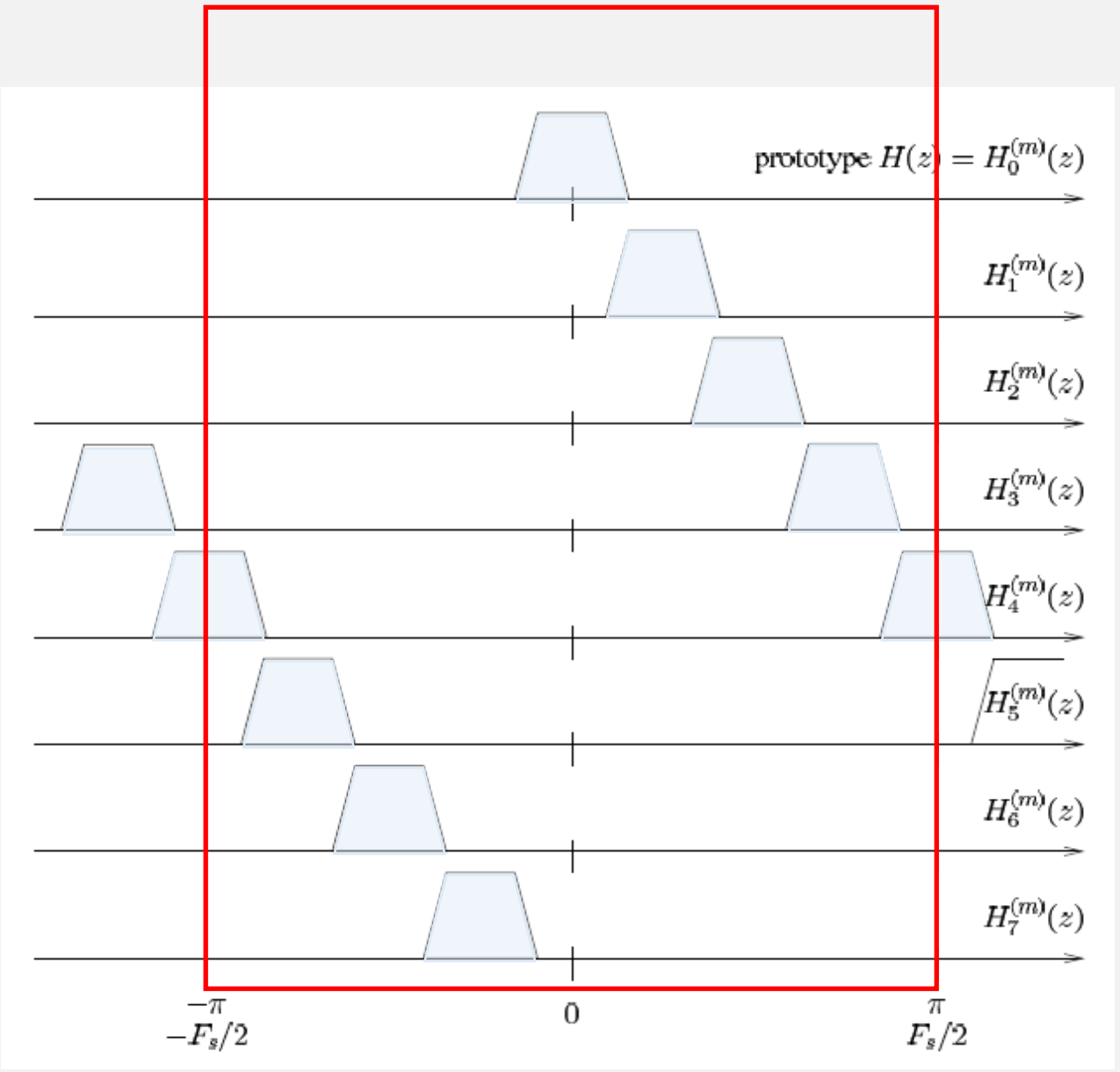
$$h^{(m)}(z) = \begin{bmatrix} a_0 + a_1 z^{-1} + a_2 z^{-2} \\ a_0 + j a_1 z^{-1} - a_2 z^{-2} \\ a_0 - a_1 z^{-1} + a_2 z^{-2} \\ a_0 - j a_1 z^{-1} - a_2 z^{-2} \end{bmatrix}$$

Bands of a complex-modulated filter bank

$$\mathbf{h}^{(m)}(z) = \begin{bmatrix} H_0^{(m)}(z) \\ H_1^{(m)}(z) \\ \vdots \\ H_{M-1}^{(m)}(z) \end{bmatrix}$$

M=8

BANDS ARE SHIFTED ”VERSIONS” OF THE PROTOTYPE



Link to polyphase decomposition

$$H(z) = \sum_{\lambda=0}^{M-1} z^{-\lambda} H_{\lambda}^{(p)}(z^M)$$

Sum terms to vector

$$\mathbf{h}^{(m)}(z) = \begin{bmatrix} H_0^{(m)}(z) \\ H_1^{(m)}(z) \\ \vdots \\ H_{M-1}^{(m)}(z) \end{bmatrix}$$

?

$$\mathbf{h}^{(p)}(z) = \begin{bmatrix} H_0^{(p)}(z^M) \\ z^{-1} H_1^{(p)}(z^M) \\ \vdots \\ z^{-(M-1)} H_{M-1}^{(p)}(z^M) \end{bmatrix}$$

Transfer functions of complex-modulated filter bank bands

Polyphase representation of the **prototype filter**

Example 1 continued

$$H(z) = a_0 + a_1 z^{-1} + a_2 z^{-2}$$

$$M = 4 \qquad W_M = e^{-j\pi/2}$$

Polyphase decomposition

$$H(z) = \sum_{\lambda=0}^{M-1} z^{-\lambda} H_\lambda^{(p)}(z^M)$$

Modulation

$$h^{(m)}(z) = \begin{bmatrix} a_0 + a_1 z^{-1} + a_2 z^{-2} \\ a_0 + j a_1 z^{-1} - a_2 z^{-2} \\ a_0 - a_1 z^{-1} + a_2 z^{-2} \\ a_0 - j a_1 z^{-1} - a_2 z^{-2} \end{bmatrix}$$

?

$$h^{(p)}(z) = \begin{bmatrix} a_0 \\ a_1 z^{-1} \\ a_2 z^{-2} \\ 0 \cdot z^{-3} \end{bmatrix}$$

Link to polyphase decomposition

$$\mathbf{W}_M = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_M^1 & W_M^2 & \cdots & W_M^{M-1} \\ 1 & W_M^2 & W_M^4 & \cdots & W_M^{2(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_M^{M-1} & W_M^{2(M-1)} & \cdots & W_M^{(M-1)(M-1)} \end{bmatrix}$$

where $W_M = \exp(-j2\pi/M)$

$$H(z) = \sum_{\lambda=0}^{M-1} z^{-\lambda} H_\lambda^{(p)}(z^M)$$

Sum terms to vector

Discrete Fourier transform (DFT)

$$\mathbf{h}^{(p)}(z) = \frac{1}{M}\mathbf{W}_M\mathbf{h}^{(m)}(z)$$

$$\mathbf{h}^{(m)}(z) = \mathbf{W}_M^*\mathbf{h}^{(p)}(z)$$

Inverse DFT

$$\mathbf{h}^{(m)}(z) = \begin{bmatrix} H_0^{(m)}(z) \\ H_1^{(m)}(z) \\ \vdots \\ H_{M-1}^{(m)}(z) \end{bmatrix}$$

Transfer functions of complex-modulated filter bank bands

$$\mathbf{h}^{(p)}(z) = \begin{bmatrix} H_0^{(p)}(z^M) \\ z^{-1}H_1^{(p)}(z^M) \\ \vdots \\ z^{-(M-1)}H_{M-1}^{(p)}(z^M) \end{bmatrix}$$

Polyphase representation of the prototype filter

Example 1 continued

$$M = 4 \qquad W_M = e^{-j\pi/2}$$

$$W_M^* = e^{j\pi/2}$$

$$W_M^* = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_M^* & W_M^{*2} & W_M^{*3} \\ 1 & W_M^{*2} & W_M^{*4} & W_M^{*6} \\ 1 & W_M^{*3} & W_M^{*6} & W_M^{*9} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{j\pi/2} & e^{j\pi} & e^{j3\pi/2} \\ 1 & e^{j\pi} & e^{j2\pi} & e^{j3\pi} \\ 1 & e^{j3\pi/2} & e^{j3\pi} & e^{j9\pi/2} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 z^{-1} \\ a_2 z^{-2} \\ 0\, z^{-3} \end{bmatrix}$$

W_M^* $\qquad$ $h^{(p)}(z)$

$$h^{(m)}(z) = \begin{bmatrix} a_0 + a_1 z^{-1} + a_2 z^{-2} \\ a_0 + j a_1 z^{-1} - a_2 z^{-2} \\ a_0 - a_1 z^{-1} + a_2 z^{-2} \\ a_0 - j a_1 z^{-1} - a_2 z^{-2} \end{bmatrix}$$

We see that the product of IDFT and polyphase representation provides required filter bank transfer functions

(SKIPPED IN LECTURE)

Example 2. 2-channel case

1. Prototype: $H(z) = \sum_{i=0}^{6} a_i z^{-i}$
2. Channel count $M = 2$
3. $W_M = \exp(-j2\pi/M) = \exp(-j\pi) = -1$

4. DFT

$$\mathbf{W}_M = \begin{bmatrix} 1 & 1 \\ 1 & W_M \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

5. IDFT

$$\mathbf{W}_M^* = \begin{bmatrix} 1 & 1 \\ 1 & W_M^* \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{h}^{(m)}(z) = \begin{bmatrix} H_0^{(m)}(z) \\ H_1^{(m)}(z) \end{bmatrix} = \begin{bmatrix} H(zW_M^0) \\ H(zW_M^1) \end{bmatrix} = \begin{bmatrix} H(z) \\ H(-z) \end{bmatrix}$$

$$= \begin{bmatrix} a_0 z^0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} + a_5 z^{-5} + a_6 z^{-6} \\ a_0 z^0 - a_1 z^{-1} + a_2 z^{-2} - a_3 z^{-3} + a_4 z^{-4} - a_5 z^{-5} + a_6 z^{-6} \end{bmatrix}$$

$$\mathbf{h}^{(p)}(z) = \begin{bmatrix} H_0^{(p)}(z^2) \\ z^{-1} H_1^{(p)}(z^2) \end{bmatrix} = \begin{bmatrix} a_0 z^0 + a_2 z^{-2} + a_4 z^{-4} + a_6 z^{-6} \\ a_1 z^{-1} + a_3 z^{-3} + a_5 z^{-5} \end{bmatrix}$$

We see easily that $\mathbf{h}^{(m)}(z) = \mathbf{W}_M^* \mathbf{h}^{(p)}(z)$

(SKIPPED IN LECTURE)

Example 3. 3-channel case

- Prototype: $H(z) = \sum_{i=0}^{6} a_i z^{-i}$
- Channel count $M = 3$
- $W_M = \exp(-j2\pi/M) = \exp(-j2\pi/3)$

$$\mathbf{W}_M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W_M & W_M^2 \\ 1 & W_M^2 & W_M^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{-j2\pi/3} & e^{-j4\pi/3} \\ 1 & e^{-j4\pi/3} & e^{-j2\pi/3} \end{bmatrix}$$

$$\mathbf{W}_M^* = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W_M^* & W_M^{*2} \\ 1 & W_M^{*2} & W_M^{*4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{j2\pi/3} & e^{j4\pi/3} \\ 1 & e^{j4\pi/3} & e^{j2\pi/3} \end{bmatrix}$$

$$\mathbf{h}^{(m)}(z) = \begin{bmatrix} H_0^{(m)}(z) \\ H_1^{(m)}(z) \\ H_2^{(m)}(z) \end{bmatrix} = \begin{bmatrix} H(zW_M^0) \\ H(zW_M^1) \\ H(zW_M^2) \end{bmatrix}$$

$$\mathbf{h}^{(p)}(z) = \begin{bmatrix} H_0^{(p)}(z^3) \\ z^{-1}H_1^{(p)}(z^3) \\ z^{-2}H_2^{(p)}(z^3) \end{bmatrix} = \begin{bmatrix} a_0 + a_3 z^{-3} + a_6 z^{-6} \\ a_1 z^{-1} + a_4 z^{-4} \\ a_2 z^{-2} + a_5 z^{-5} \end{bmatrix}$$

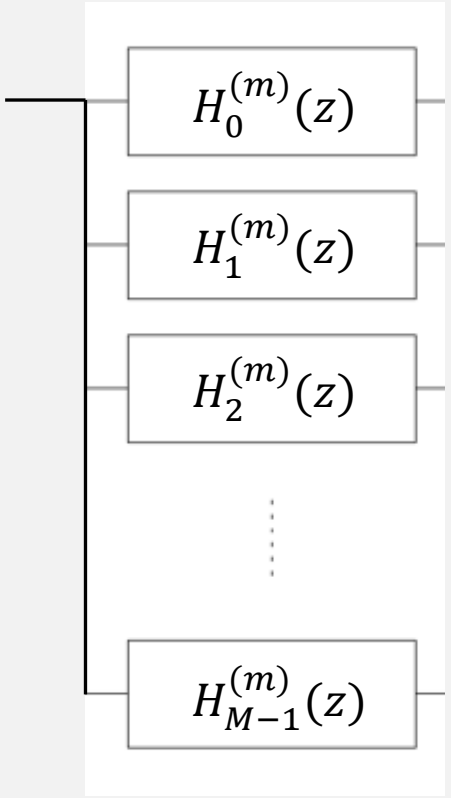
For example, deriving band #1 from polyphase representation:

$$\begin{aligned} H_1^{(m)}(z) = \sum_{i=0}^{6} a_i (zW_M)^{-i} &= a_0 + a_1 W_M^{-1} z^{-1} + a_2 W_M^{-2} z^{-2} + a_3 W_M^{-3} z^{-3} + a_4 W_M^{-4} z^{-4} + a_5 W_M^{-5} z^{-5} + a_6 W_M^{-6} z^{-6} \\ &= 1a_0 + W_M^* a_1 z^{-1} + W_M^{*2} a_2 z^{-2} + 1a_3 z^{-3} + W_M^* a_4 z^{-4} + W_M^{*2} a_5 z^{-5} + 1a_6 z^{-6} \\ &= \begin{bmatrix} 1 & W_M^* & W_M^{*2} \end{bmatrix} \mathbf{h}^{(p)}(z) \end{aligned}$$

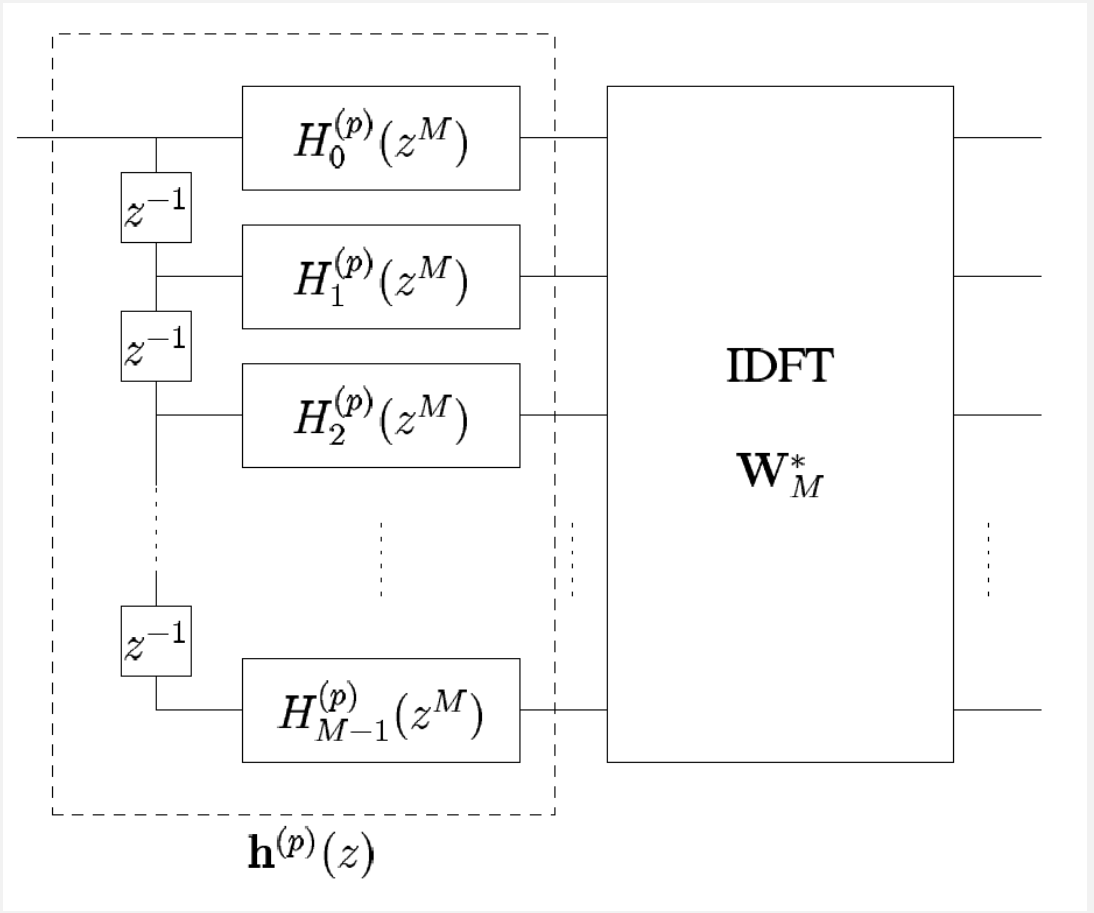
Similarly band #2, $H_2^{(m)}(z)$.

Implementation for complex-modulated filter bank

$$\mathbf{h}^{(m)}(z) = \mathbf{W}_M^* \mathbf{h}^{(p)}(z)$$

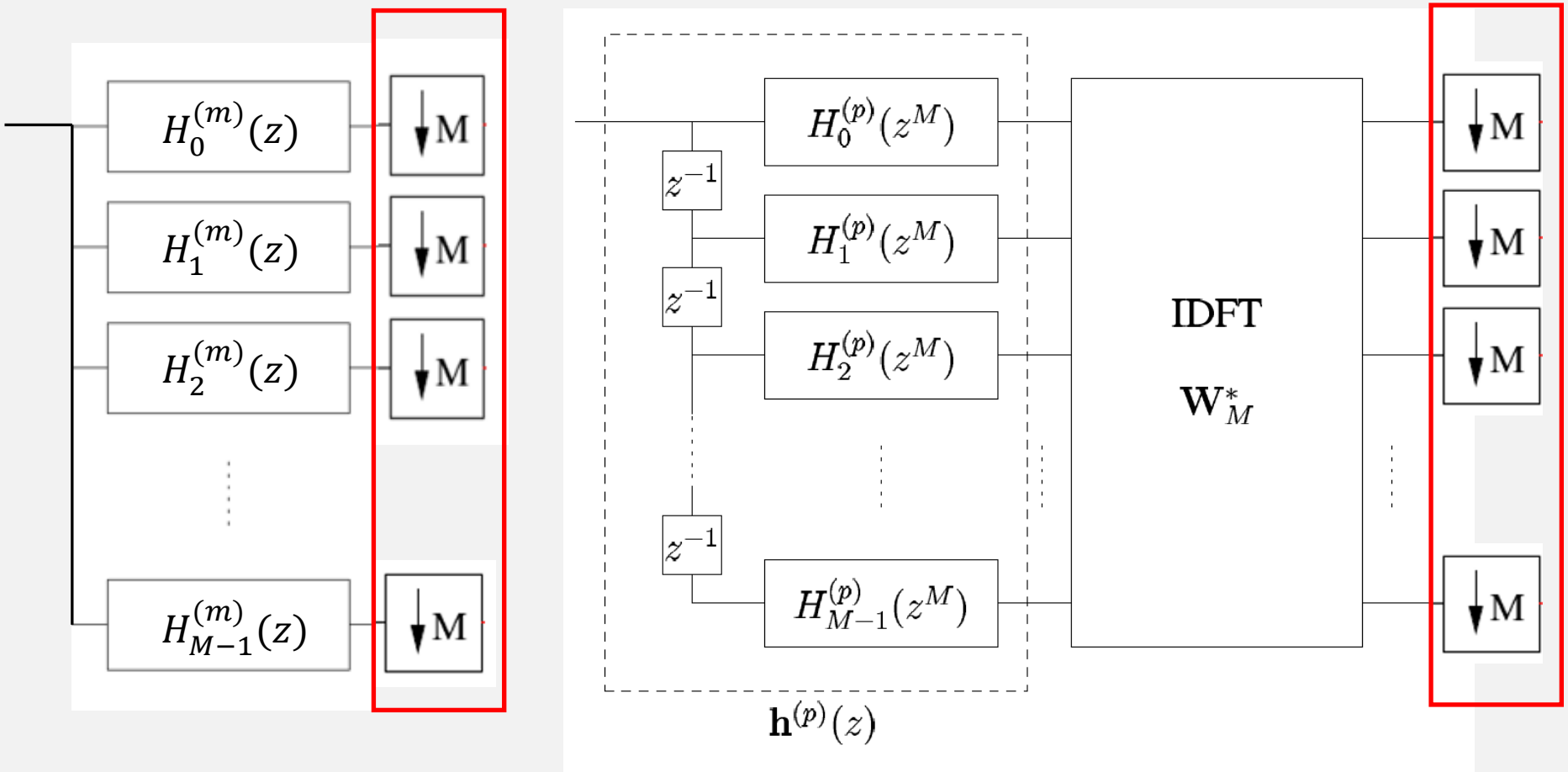


CAN BE IMPLEMENTED AS



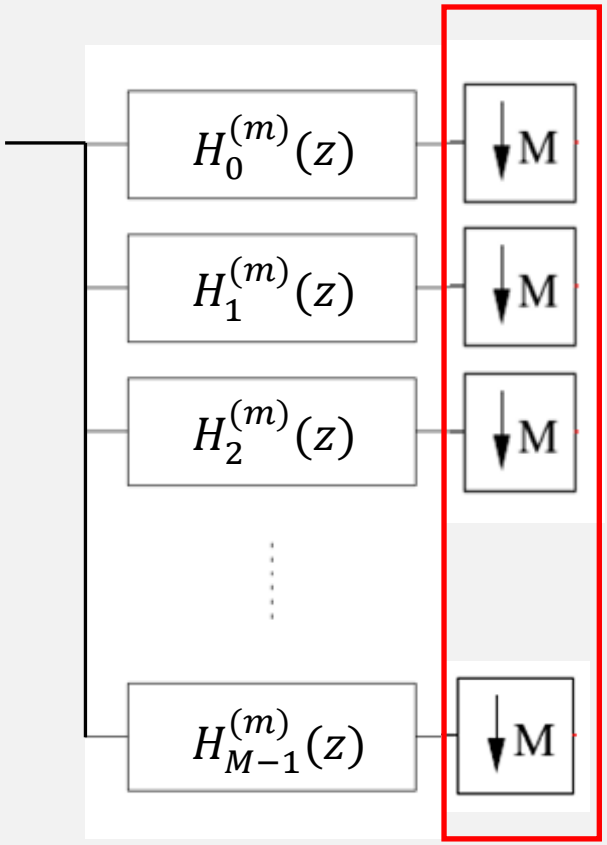
For each input sample, this structure outputs *M* complex coefficients.
But, assuming that aliasing will be tolerable, critical sampling can be applied.
What happens if outputs of the filter bank are downsampled by factor *M*?

Adding critical sampling …



Noble identities can be applied to the systems on the left and right.

1) System on the left (separate channel filters)



(1) Prototype filter $H(z)$ is a FIR filter: $H(z) = \sum_{n=0}^{N-1} a_n z^{-n}$

a_n are the coefficients of the filter (values are real numbers, not complex)

(2) Filter for channel *k* is $H_k^{(m)}(z) = H\left(zW_M^k\right)$ (where $W_M = e^{-j2\pi/M}$)

(3) We get:
$$H_k^{(m)}(z) = H\left(zW_M^k\right)$$
$$= \sum_{n=0}^{N-1} a_n \left(zW_M^k\right)^{-n}$$
$$= \sum_{n=0}^{N-1} a_n W_M^{-kn} z^{-n}$$

(4) That is, the channel filter $H_k^{(m)}(z)$ is a FIR filter: $H_k^{(m)}(z) = \sum_{n=0}^{N-1} b_n z^{-n}$

where $b_n = a_n W_M^{-kn}$ are the coefficients of the filter (complex numbers, for some channels like *k=0* imaginary parts are zero)

Polyphase decomposition with factor *M* can be done for each channel filter and decompositions allow one to apply noble identity 3 to the implementation of each critically sampled channel filter.

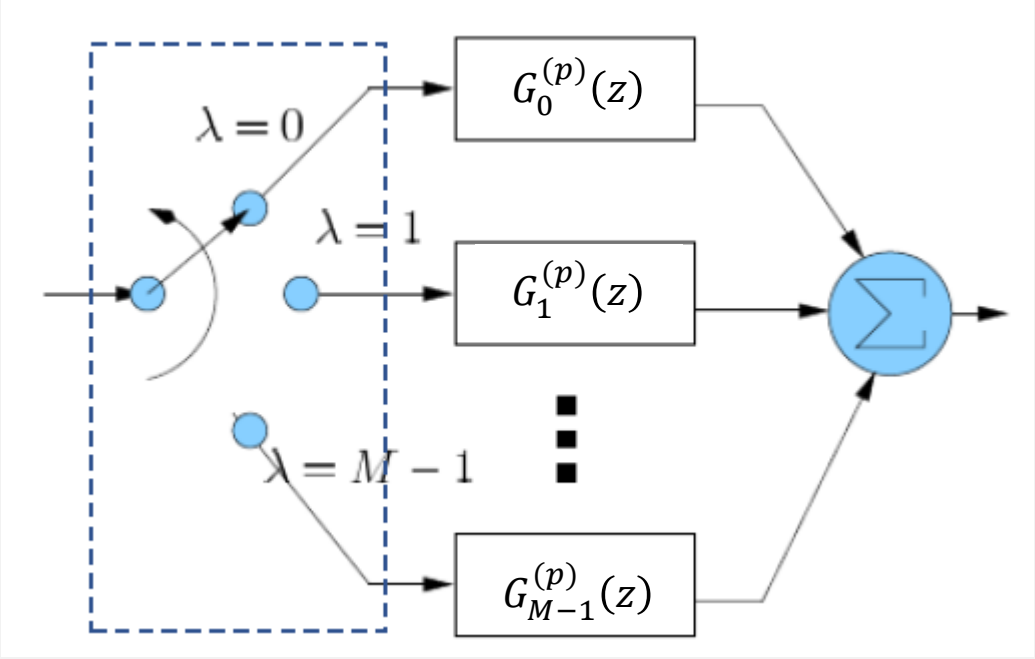
1) System on the left

So, for each channel filter

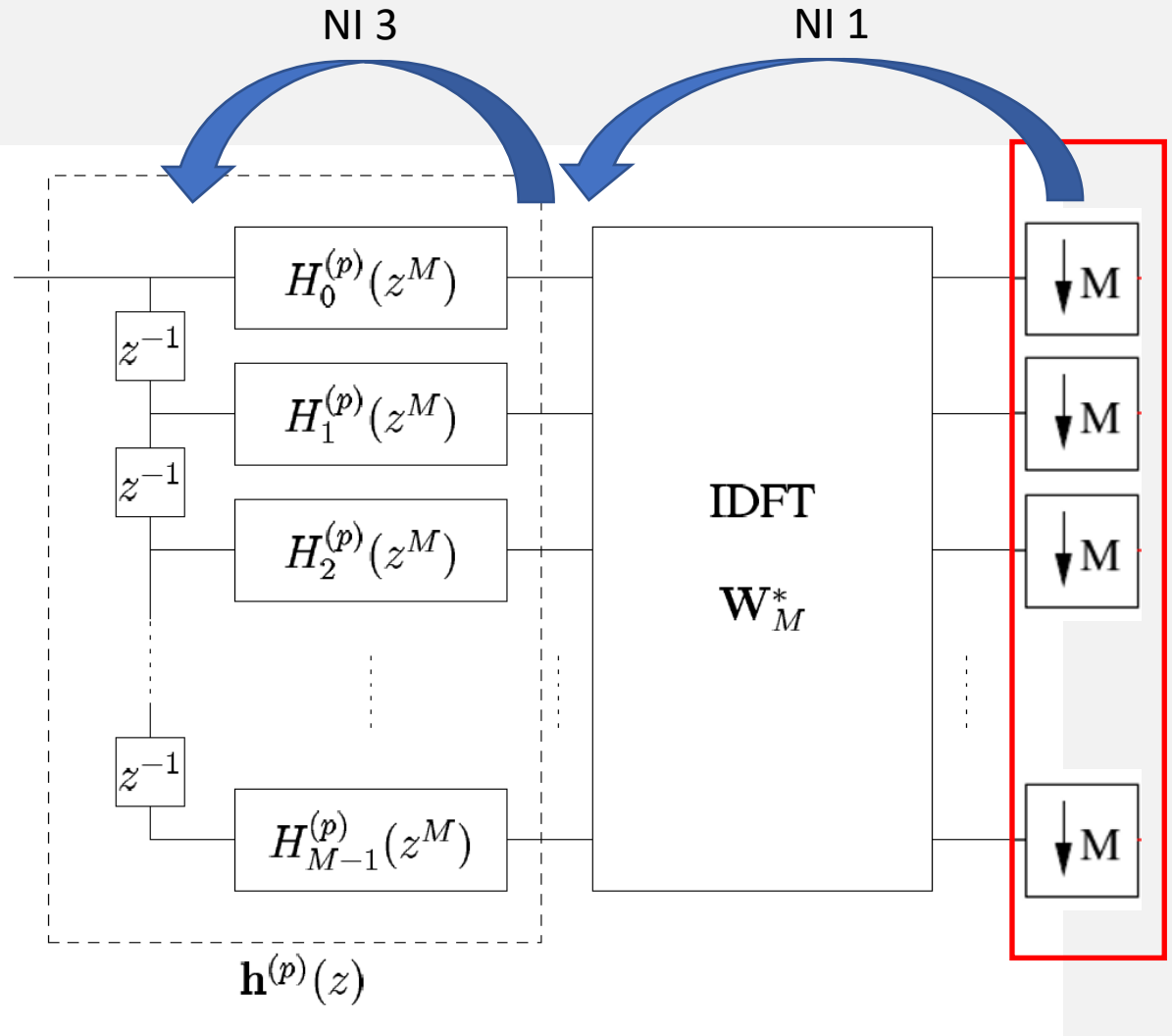
1) Perform polyphase decomposition for it

$$H_k^{(m)}(z) = \sum_{n=0}^{N-1} b_n z^{-n} = \sum_{i=0}^{M-1} z^{-i} G_i^{(p)}(z^M)$$

2) Implement using commutation

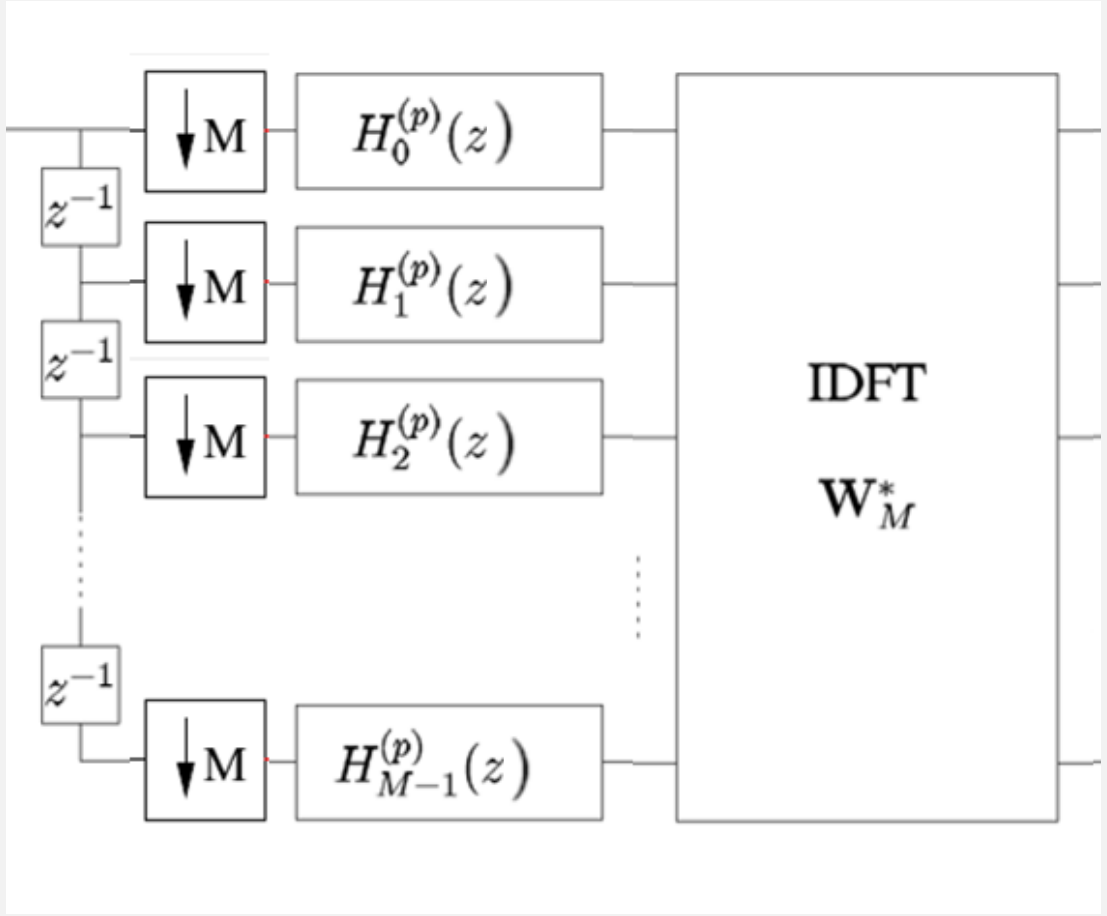


2) System on the right



IDFT contains just arithmetic operations => apply NI 1
Polyphase component filters have non-zero values just for $z^{-nM}(n = 0,1,2, \ldots)$=> apply NI 3

RESULT:



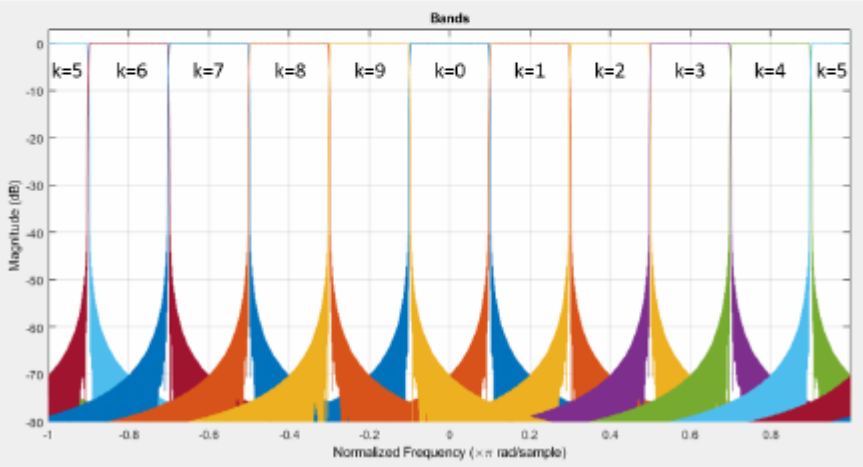
To summarize …

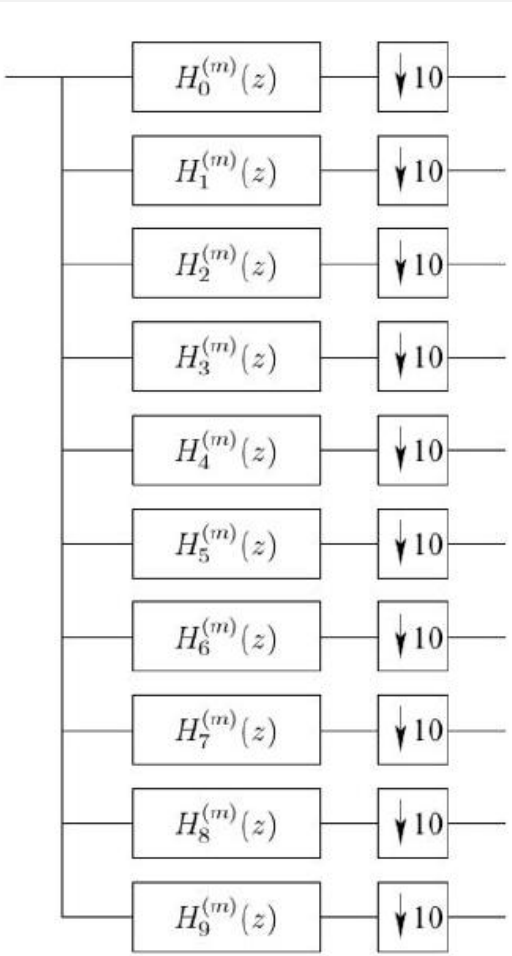
- Efficient implementation of filter banks is based on many ideas:
  - Modulation of a prototype filter
  - Critical sampling for having sufficient coefficient sample rate
  - Relationship between modulation and polyphase decomposition
  - Application of noble identities to resulting structures

  => **Leads to minimization of redundant computations**
- In addition, IDFT seen in the structure has efficient implementations for specific M e.g. when $M = 2^k$ (i.e., Fast Fourier Transform)
- More information in **intro5c.pdf**

About DT5 Problem 2

T2. (2p) Computational complexity of critically sampled modulated filter banks can be reduced using polyphase decomposition techniques and fast transforms. To see how effective the techniques are, let us consider implementation of the critically sampled 10-channel complex-modulated filter bank, whose frequency response for each band k is illustrated below. The prototype filter $H(z)$ is symmetric, has real coefficients, and its length is N. The input to the system is a stream of complex numbers and associated sample rate is f_{in}.





Prototype filter

$H(z) = \sum_{n=0}^{N-1} a_n z^{-n}$

Modulated filter for channel k

$H_k^{(m)}(z) = H(W_{10}^k z) = \sum_{n=0}^{N-1} a_n W_{10}^{-kn} z^{-n}$

where $W_{10} = \exp(-j2\pi/10)$

Determining numbers of multiplications & additions per second in three cases:

(a) Filtering + downsampling afterwards (picture above)
(b) Applying polyphase decomp & noble identities to channel filters (see previous slides #42-43)
(c) IDFT based solution (see previous slide #44)

In DT5 Problem 2, use information from Slide #19

CHARACTERISTICS OF IMPLEMENTED OPERATION			NUMBER OF **ADD** AND **MPLY** OPERATIONS	
Operation	**Input 1**	**Input 2**	**ADDs**	**MPLYs**
addition	real	real	1	-
multiplication	real	real	-	1
addition	real	complex	1	-
multiplication	real	complex	-	2
addition	complex	complex	2	-
multiplication	complex	complex	2	4
multiplication	complex	complex (fixed)	3	3

For more information, see **opcounts.pdf**

Summary of lecture

- N-point DFT for $N = 2^k$ has a systematic implementation visible in butterfly diagrams
- Advantage of frequency domain implementation shows up with FFT lengths $\geq$128 and long filters
- Modulated filter banks can be implemented efficiently using multirate approach