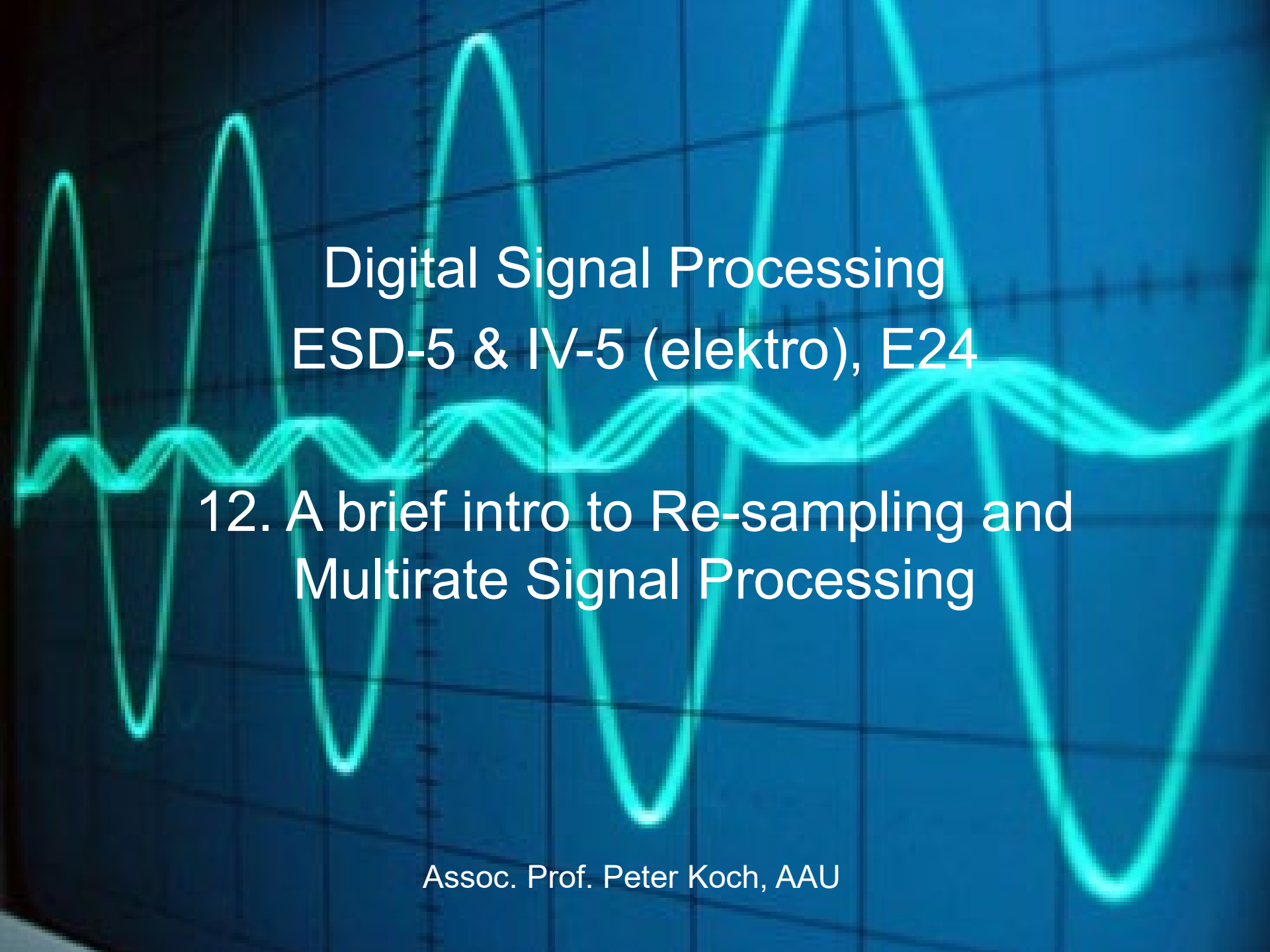


Digital Signal Processing
ESD-5 & IV-5 (elektro), E24

12. A brief intro to Re-sampling and Multirate Signal Processing

Assoc. Prof. Peter Koch, AAU

Setting the scene for today's lecture

- Multirate signal processing is a domain within digital signal processing which has gained significant interest over the past 1-2 decades – this is mainly due to its applicability in wireless and mobile communication.

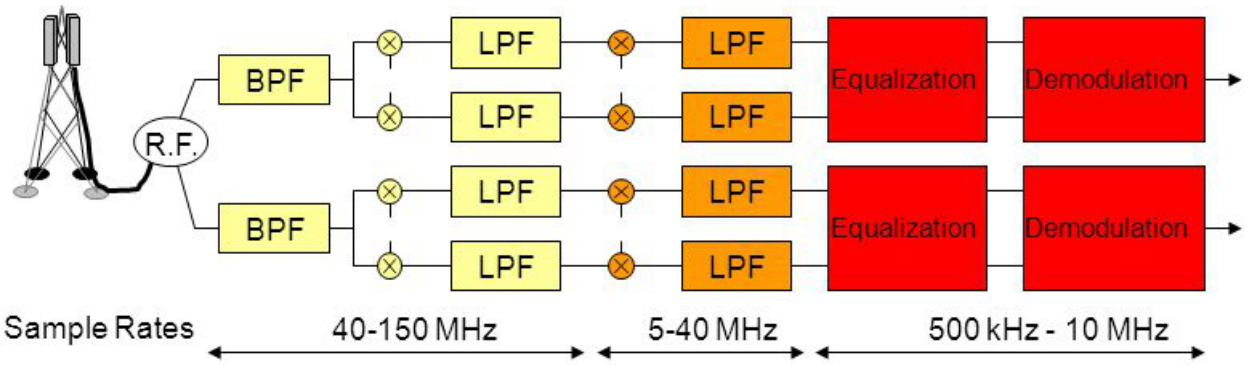
- Multirate signal processing is based on the idea of resampling an already sampled signal, thus in practice the sample rate is changed – typically several times throughout the complete signal chain.

- Using such resampling techniques it is possible to achieve various benefits in the implementation, e.g., reduction of the computational complexity.

- To get a complete overview of the domain, we could easily discuss for a complete 5 ECTS course…!! For that reason, today we will touch only on some few very basic topics…

An example of a system with multiple different sample rates

- Looking at a typical wireless base-station receiver application, we can see that filters can form a major part of the DSP functionality
- Because the object of such a system is to down convert from high frequencies to lower frequencies, however, this is also coupled with many *different sample rates*…



…and here is another one

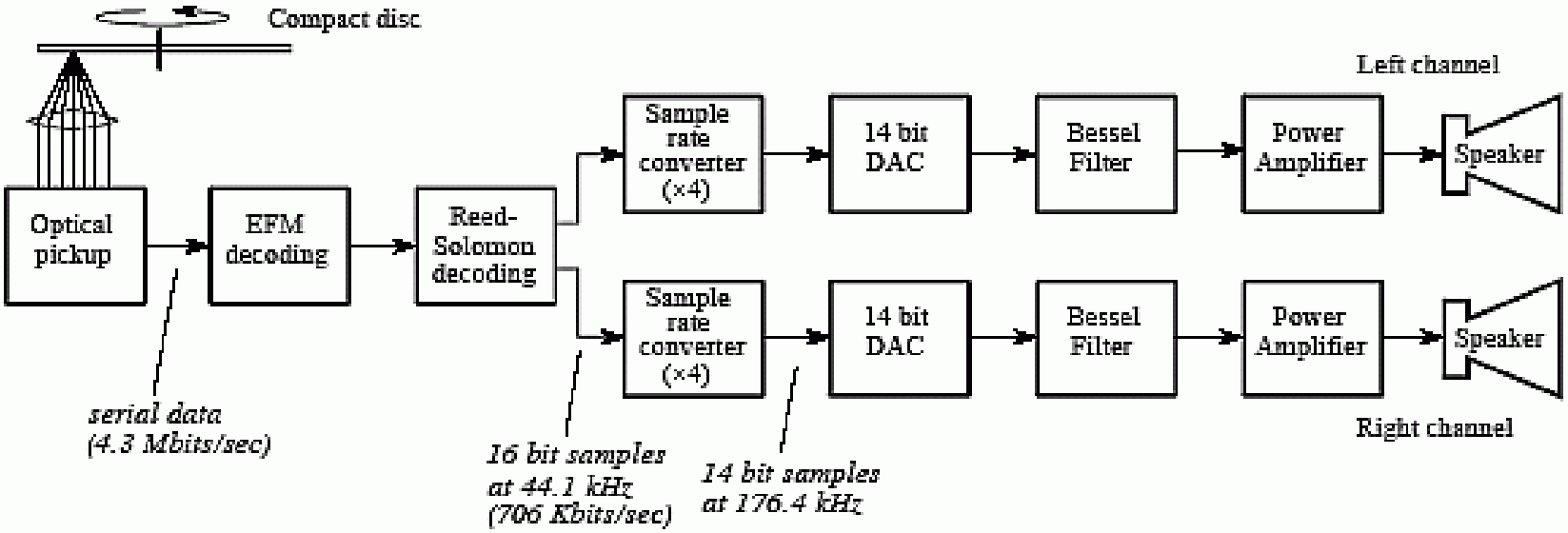
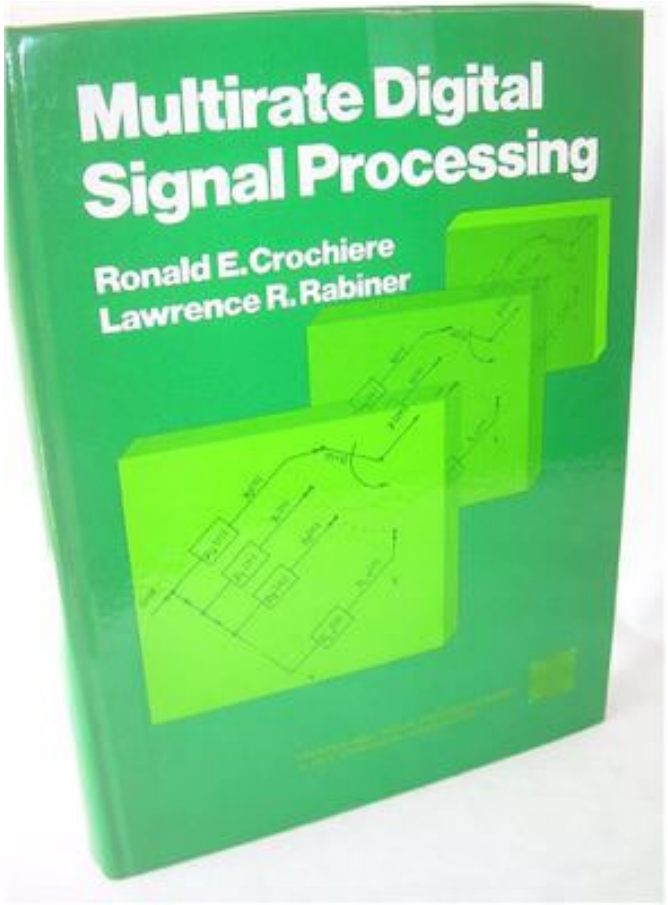
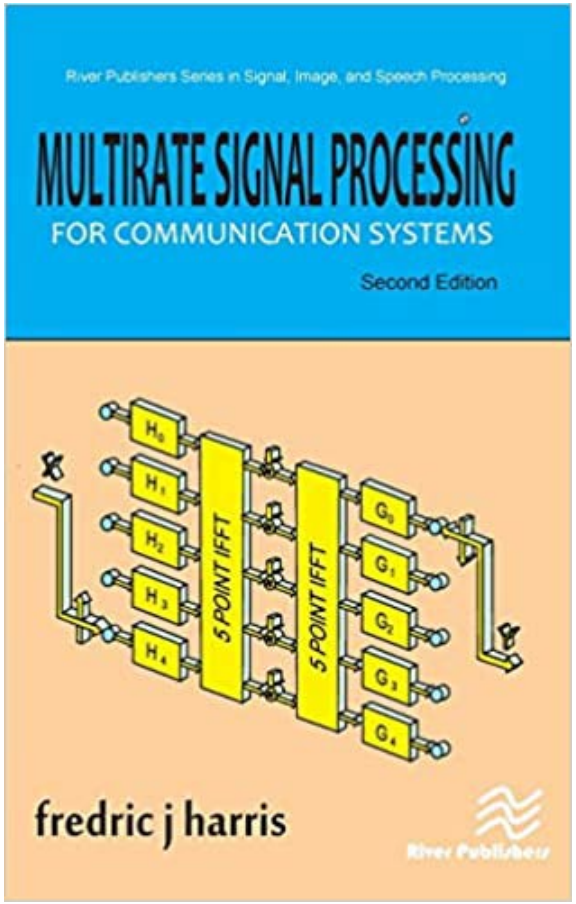


FIGURE 22-6
Compact disc playback block diagram. The digital information is retrieved from the disc with an optical sensor, corrected for EFM and Reed-Solomon encoding, and converted to stereo analog signals.

The best text books on Multirate Signal Processing…





…and here is a journal paper which is also worth reading

Interpolation and Decimation of Digital Signals—A Tutorial Review

RONALD E. CROCHIERE, SENIOR MEMBER, IEEE, AND LAWRENCE R. RABINER, FELLOW, IEEE

Invited Paper

Abstract—The concepts of digital signal processing are playing an increasingly important role in the area of multirate signal processing, i.e. signal processing algorithms that involve more than one sampling rate. In this paper we present a tutorial overview of multirate digital signal processing as applied to systems for decimation and interpolation. We first discuss a theoretical model for such systems (based on the sampling theorem) and then show how various structures can be derived to provide efficient implementations of these systems. Design techniques for the linear-time-invariant components of these systems (the digital filter) are discussed, and finally the ideas behind multistage implementations for increased efficiency are presented.

I. Introduction

ONE OF THE MOST fundamental concepts of digital signal processing is the idea of sampling a continuous process to provide a set of numbers which, in some sense, is representative of the characteristics of the process being sampled. If we denote a continuous function from the process being sampled as $x_C(t)$, $-\infty \leqslant t \leqslant \infty$ where x_C is a continuous function of the continuous variable t (t may represent time, space, or any other continuous physical variable), then we can define the set of samples as $x_D(n)$, $-\infty \leqslant n \leqslant \infty$ where the correspondence between t and n is essentially specified by the sampling process, i.e.,

$$n = q(t) \tag{1a}$$

Many types of sampling have been discussed in the literature [1]–[3] including nonuniform sampling, uniform sampling, and multiple function uniform sampling. The most common form of sampling, and the one which we will refer to throughout this paper is uniform (periodic) sampling in which

$$q(t) = t/T = n \tag{1b}$$

i.e., the samples $x_D(n)$ are uniformly spaced in the dimension t, occurring nT apart. For uniform sampling we define the sampling period as T and the sampling rate as

$$F = 1/T \tag{2}$$

It should be clear from the above discussion that $x_C(t)$ can be sampled with any sampling period T. However, for a unique correspondence between the continuous function $x_C(t)$ and the discrete sequence $x_D(n)$, it is necessary that the sampling period T be chosen to satisfy the requirements of the Nyquist sampling theorem. This concept of a unique analog waveform corresponding to a digital sequence will often be used in the course of our discussion to provide greater intuitive insights into the nature of the processing algorithms that we will be considering.

The sampling period T is a fundamental consideration in many signal processing techniques and applications. It often determines the convenience, efficiency, and/or accuracy in which the signal processing can be performed. In some cases an input signal may already be sampled at some predetermined sampling period T and the goal is to convert this sampled signal to a new sampled signal at a different sampling period T' such that the resulting signal corresponds to the same analog function. In other cases it may be more efficient or convenient to perform different parts of a processing algorithm at different sampling rates in which case it may be necessary to convert the sampling rates of the signals in the system from one rate to another.

The process of digitally converting the sampling rate of a signal from a given rate $F = 1/T$ to a different rate $F' = 1/T'$ is called *sampling rate conversion*. When the new sampling rate is higher than the original sampling rate, i.e.,

$$F' > F \tag{3a}$$

or

$$T' < T \tag{3b}$$

the process is generally called *interpolation* since we are creating samples of the original physical process from a reduced set of samples. Historically the mathematical process of interpolation, or "reading between the lines," has received widespread attention from mathematicians who were interested in the problem of tabulating useful mathematical functions. The question was how often a given function had to be tabulated (sampled) so that someone could use a simple interpolation rule to obtain accurate values of the function at any higher sampling rate [4]. Not only did this early work lead to an appreciation of the sampling process, but it also led to several interesting classes of "interpolation functions" which could provide almost arbitrarily high accuracy in the interpolated values, provided that sufficient tabulated values of the function were available.

The process of digitally converting the sampling rate of a signal from a given rate F to a lower rate F', i.e.,

$$F' < F \tag{4a}$$

Manuscript received June 25, 1980.
The authors are with Bell Laboratories, Murray Hill, NJ 07974.

So, what is MRSP all about…?

Well, basically it is a matter of changing the sample rate of an already sampled signal, i.e., given a sequence $x[n]$ which is sampled with f_s, then we want to either decrease the sample frequency to f_s/M, down-sampling, or to increase it to Lf_s, up-sampling.

Changing the sample frequency is a process which is tightly coupled to ordinary sampling, i.e.,

1) compliance with Shannon's sample theorem is necessary, and

2) sampling will concequently lead to periodicity in the frequency domain

Initially, we will study (briefly) the concept of down- and up-sampling, and discuss the impact to be seen in the frequency domain.

Down-sampling, or sample-rate reduction

In down-sampling we ”pick out” every M’th sample from the sequence $x[n]$.

$$x_d[n] = x[nM] = x_c(nMT)$$

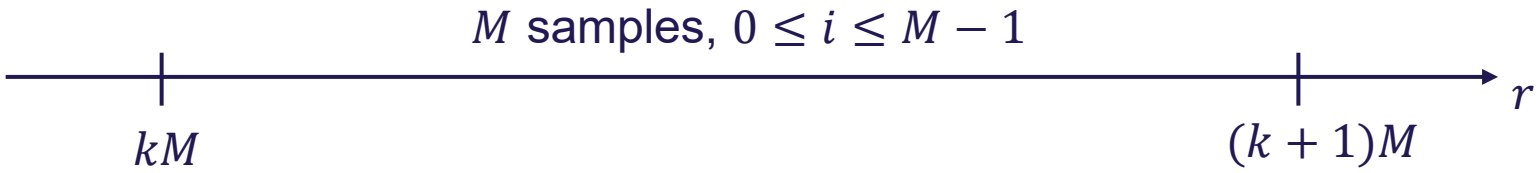
From the sample theorem we remember that the DTFT of $x[n]$ is

$$X(e^{j\omega}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left[j\left(\frac{\omega}{T} - \frac{2\pi k}{T}\right)\right]$$

If $x[n]$ is down-sampled with M, then the sample period becomes $T_d = MT$, and thus

$$X_d(e^{j\omega}) = \frac{1}{T_d}\sum_{r=-\infty}^{\infty} X_c\left[j\left(\frac{\omega}{T_d} - \frac{2\pi r}{T_d}\right)\right] = \frac{1}{MT}\sum_{r=-\infty}^{\infty} X_c\left[j\left(\frac{\omega}{MT} - \frac{2\pi r}{MT}\right)\right]$$

The summation index r is now reformulated; $r = i + kM$, where $k, i \in \mathbb{Z}$



Now, we use $r = i + kM$ in

$$X_d(e^{j\omega}) = \frac{1}{MT}\sum_{r=-\infty}^{\infty} X_c\left[j\left(\frac{\omega}{MT} - \frac{2\pi r}{MT}\right)\right]$$

i.e., the sum is split into two sums (one over i, and one over k), and r is inserted here

$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1}\left\{\frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left[j\left(\frac{\omega}{MT} - \frac{2\pi k}{T} - \frac{2\pi i}{MT}\right)\right]\right\}$$

Now, the term in the "Tuborg" brackets is quite similar to the DTFT of $x[n]$;

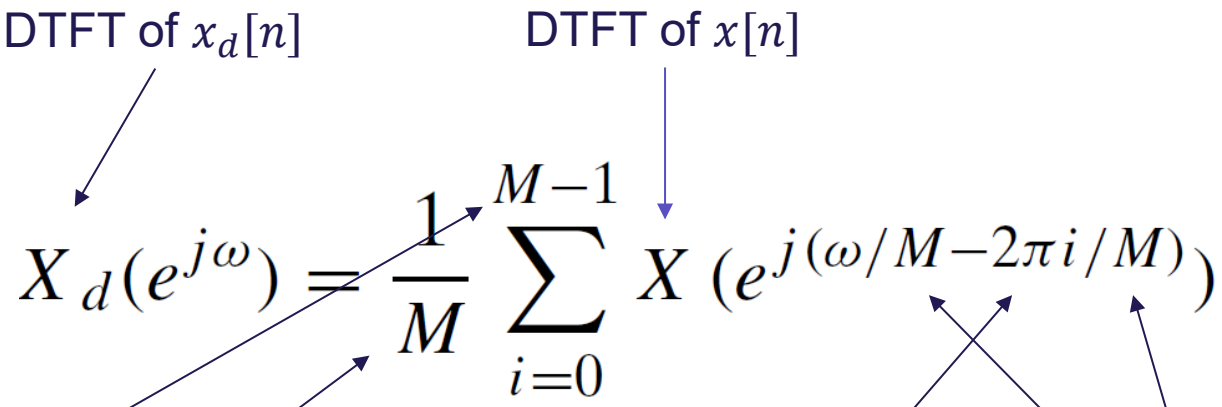
$$X(e^{j\omega}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left[j\left(\frac{\omega}{T} - \frac{2\pi k}{T}\right)\right]$$

but now the frequency is $\frac{\omega - 2\pi i}{MT}$ rather than $\frac{\omega}{T}$, and thus;

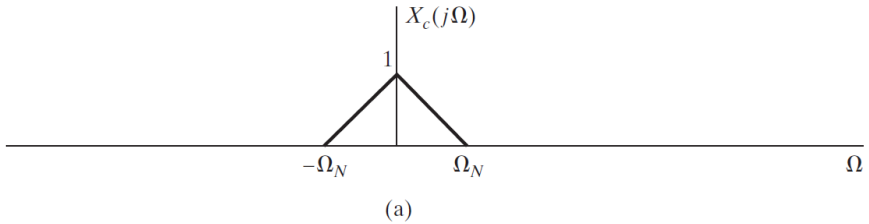
$$X(e^{j(\omega-2\pi i)/M}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left[j\left(\frac{\omega - 2\pi i}{MT} - \frac{2\pi k}{T}\right)\right]$$ which leads to;

$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} X(e^{j(\omega/M - 2\pi i/M)})$$

So, how should we interpret this equation…?

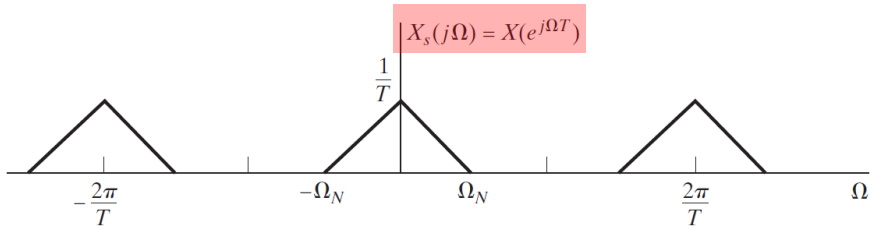


M amplitude-scaled copies of the periodic DTFT $X(e^{j\omega})$, which are frequency-scaled by M and shifted by integer multiples of 2π.



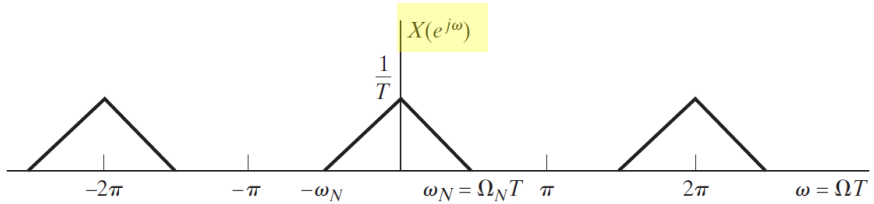
(a)

The FT of the continuous-time signal $x_c(t)$



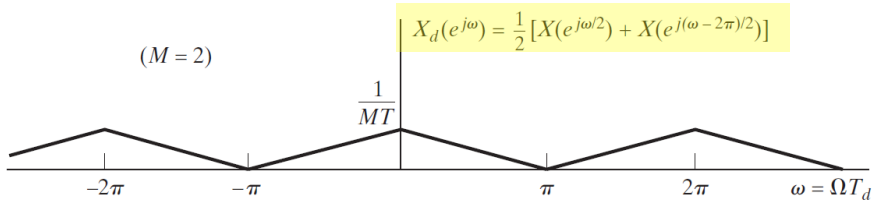
(b)

The DTFT of the discrete-time signal $x[n]$ as a function of Ω. The signal is sampled with $f_s = {}^1/_T$.



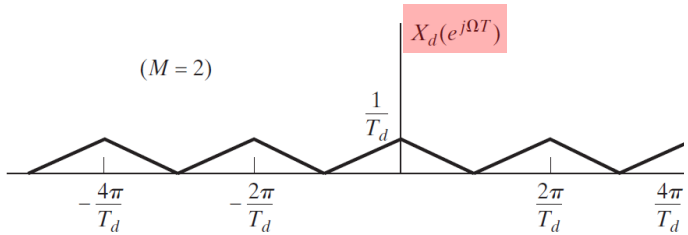
(c)

The DTFT of the discrete-time signal $x[n]$ as a function of ω. Note that $\omega_N = {}^\pi/_2$.



(d)

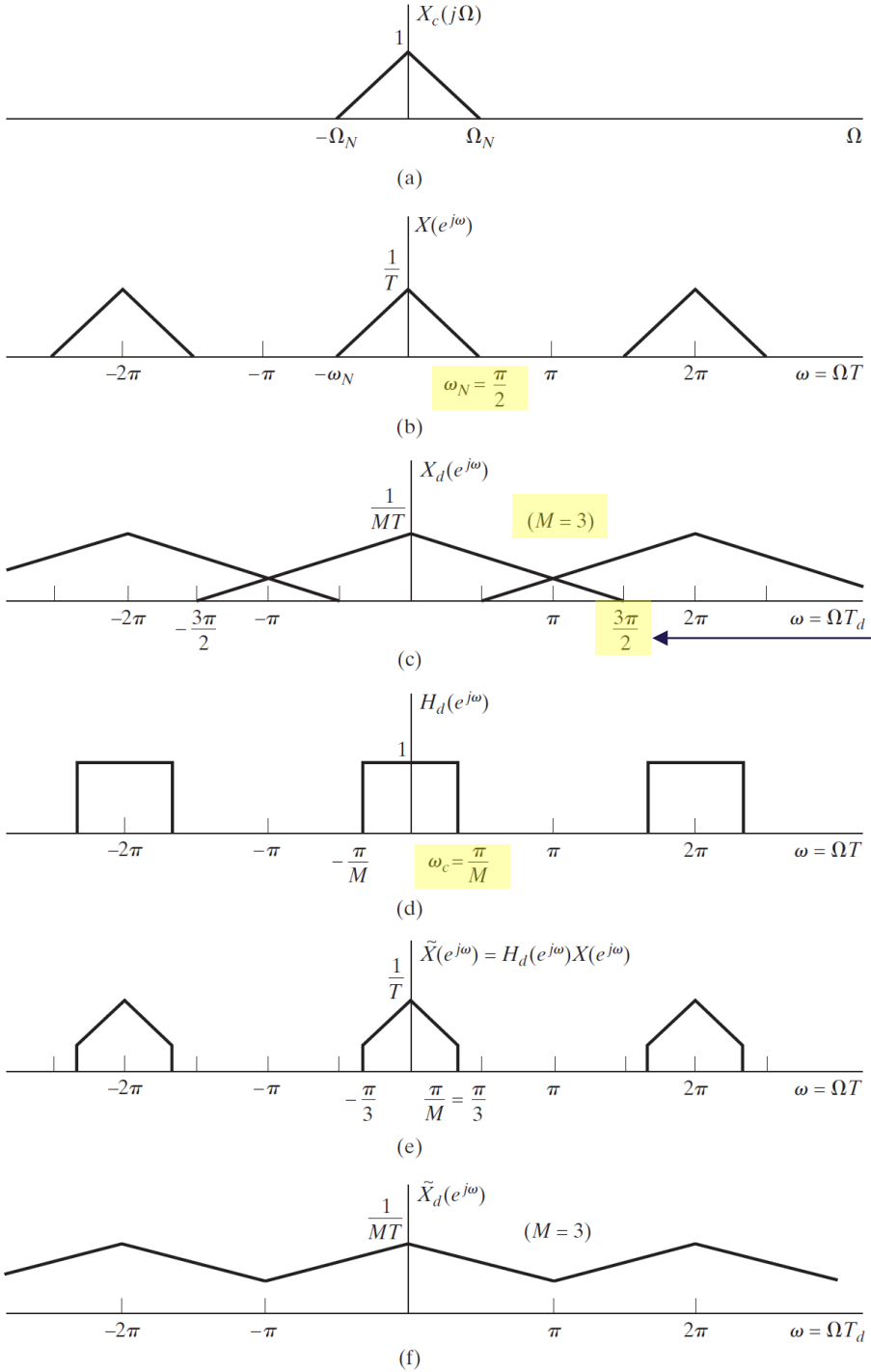
The DTFT of the down-sampled signal $x_d[n]$ as a function of ω. The down-sampling factor is $M = 2$, i.e., the new sample freq. is now ${}^1/_{T_d} = {}^1/_{MT} = {}^{f_s}/_M$.
Since the bandwidth of the signal is unchanged, the Nyquist freq. is now $\omega_N = \pi$



(e)

What we can learn from these considerations and the example is that things can go wrong, i.e., the down-sampled sequence $x_d[n]$ may suffer from ALIASING, if M is chosen too large as compared to the initial sample period T, and the bandwidth of the signal $x_c(t)$.

For that reason, we may want to apply a discrete-time anti-aliasing filter prior to down-sampling. See next slide…



Now there is aliasing…!!

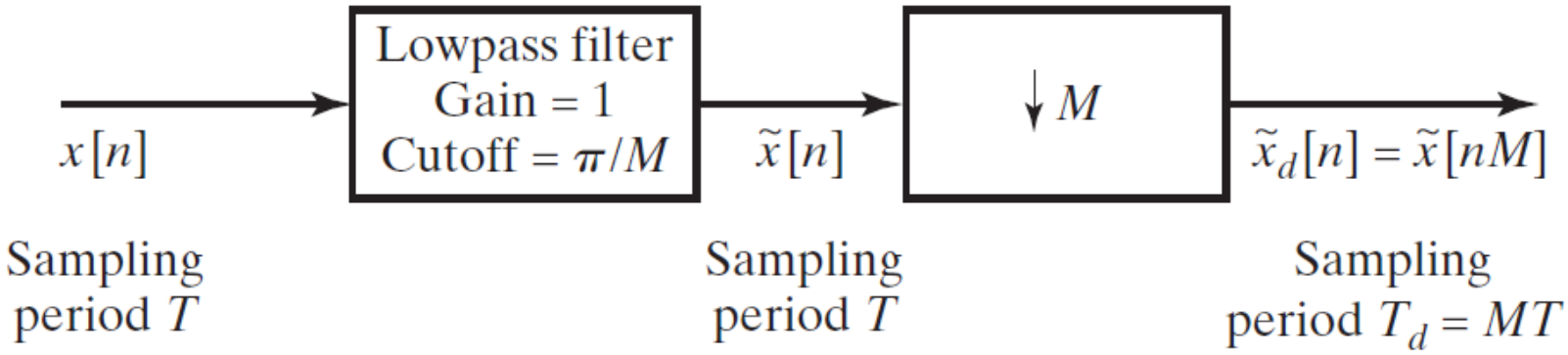
$3\omega_N$

To avoid aliasing, the sequence $x[n]$ is lowpass filtered with $\omega_c = {}^{\pi}/_{M}$.

This figure represents (b) multiplied with (d).

Due to the lowpass filtering, the down-sampled signal is no longer identical to the original signal, but there is no aliasing.

The general Down-sampling system – called the Decimator



Up-sampling, or sample-rate increase

The idea now is to derive a sequence, $x_i[n]$, which is characterized by a sample frequency which is increased by L as compared to the original sequence $x[n]$.

$$x_i[n] = x[{}^{n}/_{L}] = x_c({}^{nT}/_{L})$$

To do so, we need to create a new sequence with L times the number of samples which are in $x[n]$.

Or put another way – for every sample in $x[n]$, there should be L samples in the up-sampled sequence. Let's call this sequence $x_e[n]$.

$$x_e[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \ldots, \\ 0, & \text{otherwise}, \end{cases}$$

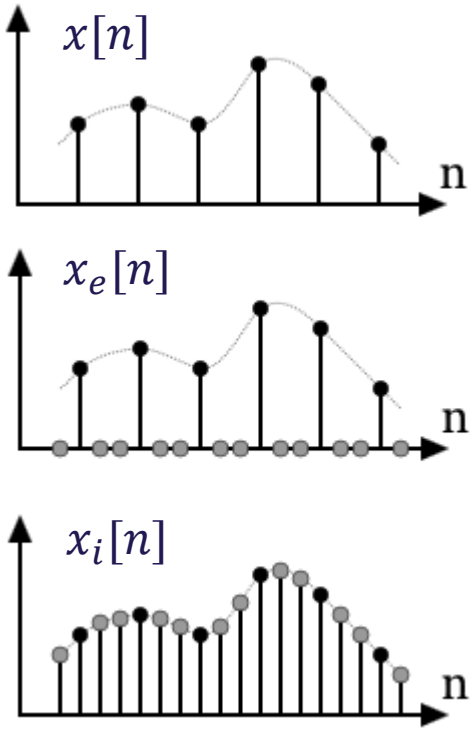
In up-sampling, $L > 0$, but we need the negative indices to manage non-causal signals…

Basically, what we do is to insert $L - 1$ zeros (0) in between every sample of $x[n]$.

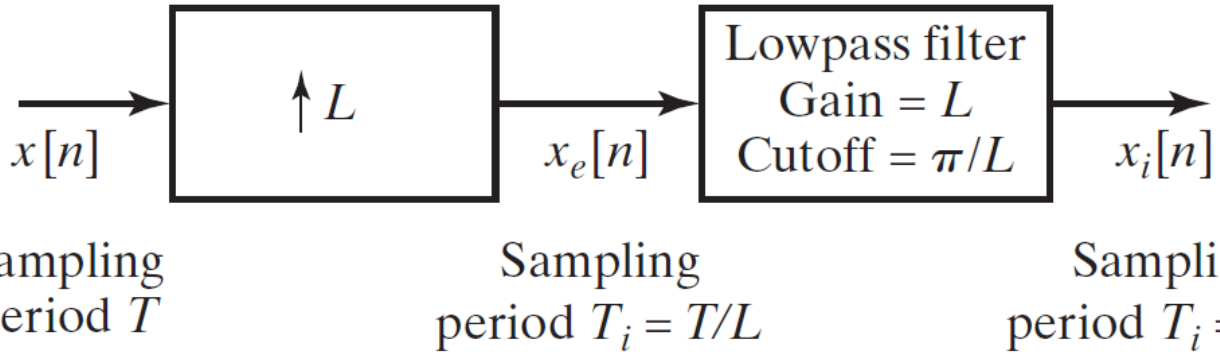
This new sequence can also be written as

$$x_e[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - kL]$$

The problem now is the following; how do we manipulate all the inserted zeros such that we obtain the sequence $x_i[n]$ representing an up-sampled version of $x[n]$.



The immediate idea is to ”smoothen” the sequence $x_e[n]$, which can be done by low-pass filtering.



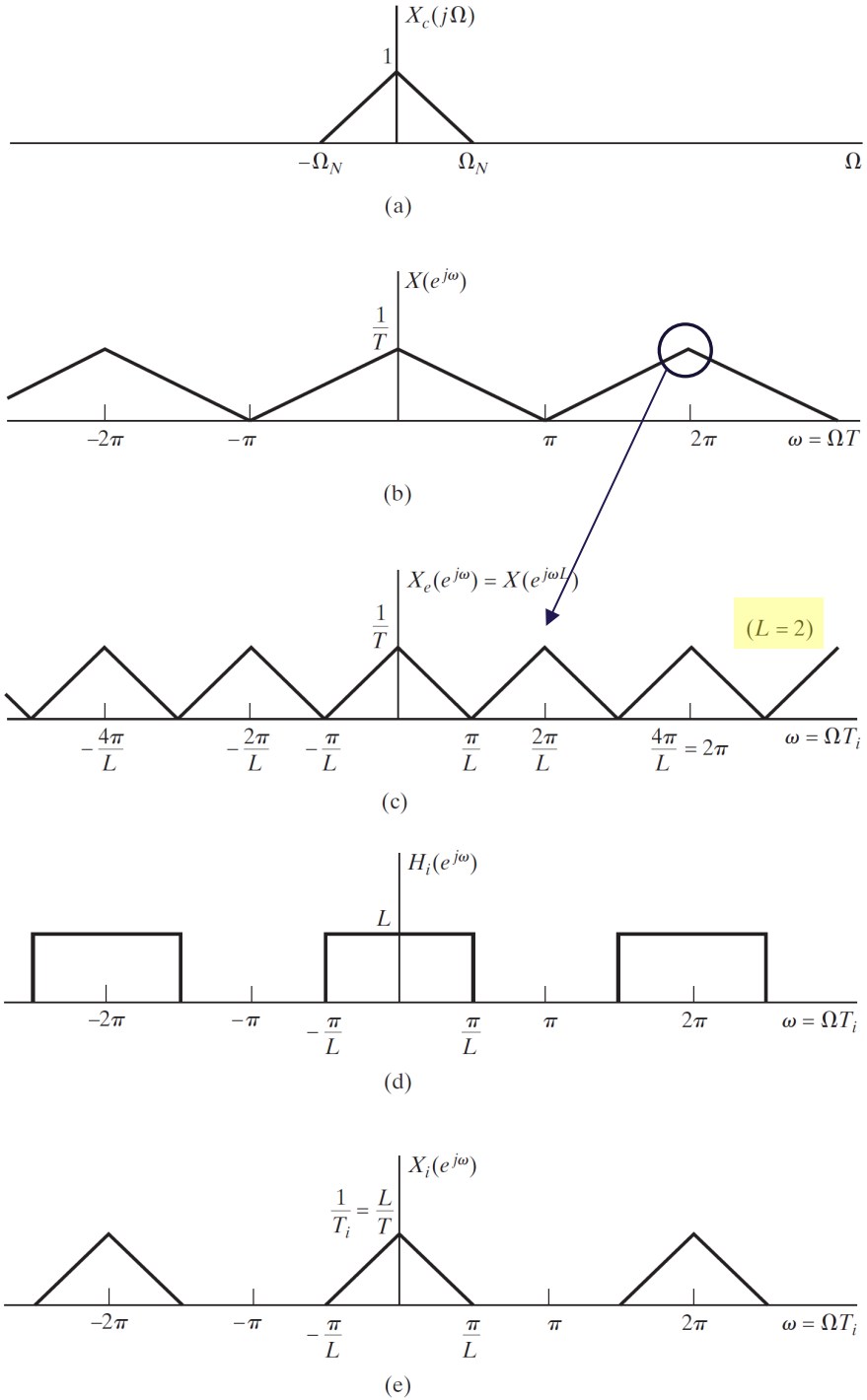
This system is now discussed in the frequency domain. Since we have an expression for the sequence $x_e[n]$, we can easily derive the DTFT…

$$X_e(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x[k]\delta[n-kL] \right) e^{-j\omega n}$$

$$= \sum_{k=-\infty}^{\infty} x[k]e^{-j\omega L k} = X(e^{j\omega L}).$$

This is an interesting result which shows that the Fourier transform of the expanded sequence $x_e[n]$ is a frequency-scaled version of the Fourier transform of the original sequence $x[n]$.

Here ”frequency-scaled” means ”compressed” with a factor L.



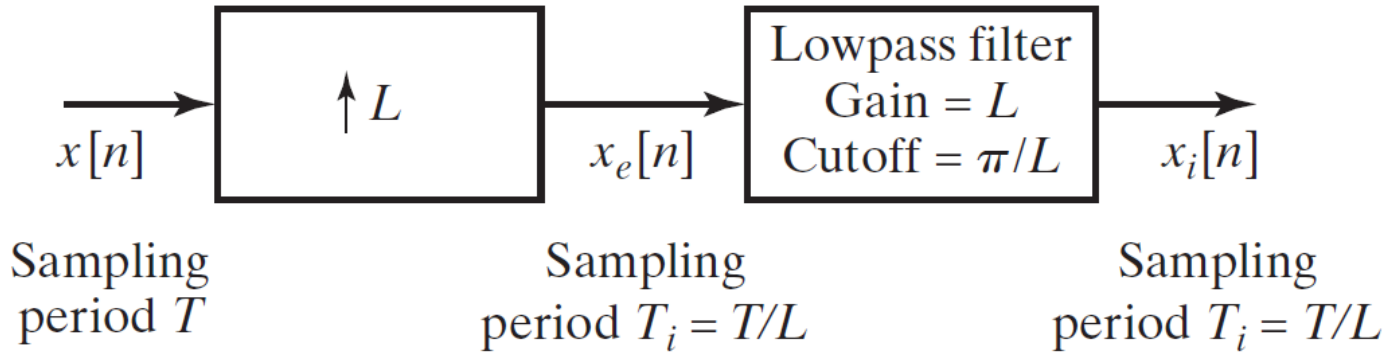
The FT of the continuous-time signal $x_c(t)$

The DTFT of the discrete-time signal $x[n]$ as a function of Ω. The signal is sampled with $f_s = {}^1/_T$. Note that the Nyquist frequency is exactly half the sample frequency.

The DTFT of the expanded signal $x_e[n]$ as a function of the compressed frequency $\omega = \Omega\, {}^T/_L$. For $L = 2$, the shape of the spectrum is not changed, but it now holds images…

To remove these unwanted images, $x_e[n]$ is low-pass filtered with $\omega_c = {}^\pi/_L$, i.e., multiply (c) and (d). Note the DC-gain equal to L.

After low-pass filtering, we obtain the sequence $x_i[n]$ which represents $X(e^{j\omega})$ but now with $T_i = {}^T/_L$.



The overall system, i.e., the cascaded expander and the LP-filter, is denoted an interpolator – and it performs interpolation on $L - 1$ samples located in between two consecutive samples from the original sequence.

O&S discuss several types of practical LP-filter which can be used in the interpolator.

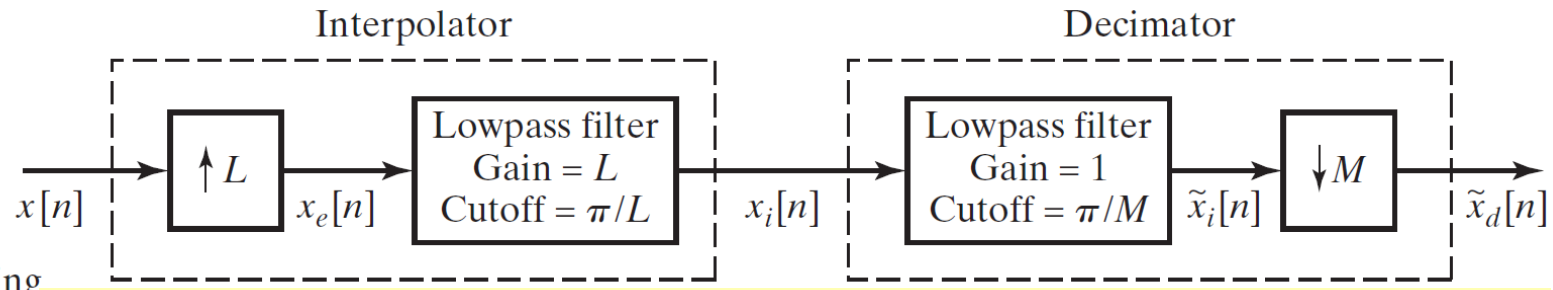
Since the practical LP-filter is not ideal (for obvious reasons), we should expect some ”coloring” of $x[n]$ when it is presented at the output as a sample-rate increased signal $x_i[n]$.

This is left for self-studying…

Sample-rate modification with a rational factor $^{L}/_{M}$

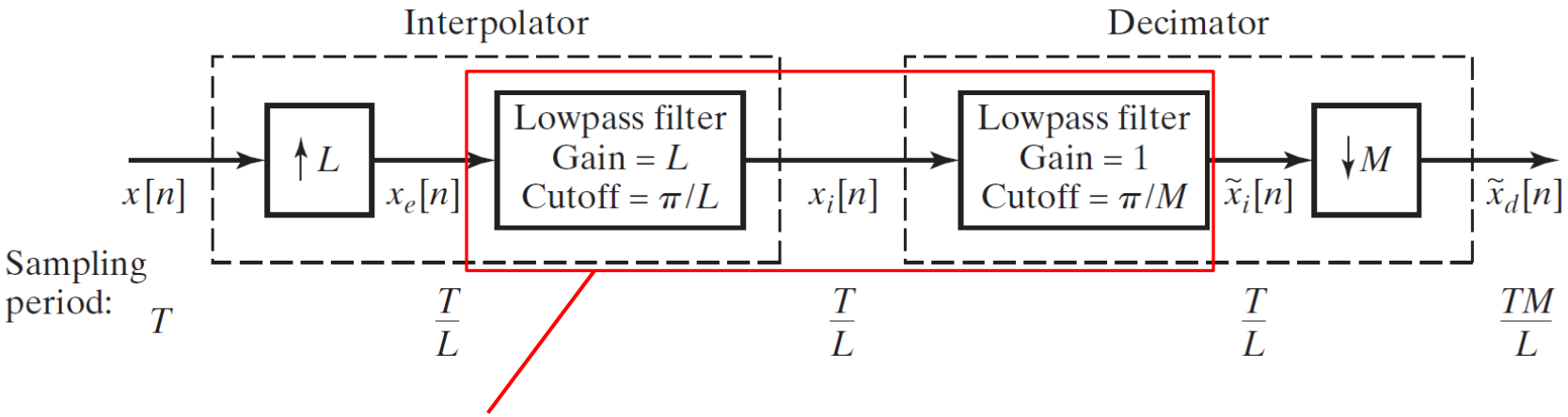
An immediate idea is to cascade up-sampling and down-sampling such that the resulting sample-frequency becomes a rational factor, $^{L}/_{M}$.

Note that the interpolation (L) is placed prior to the decimator (M)…!!



Sampling period:	T	$\frac{T}{L}$	$\frac{T}{L}$	$\frac{T}{L}$	$\frac{TM}{L}$
Sample frequency	f_s	Lf_s	Lf_s	Lf_s	$f_s' = \frac{L}{M} f_s$

If $L > M$ then $f_s' > 1$, and if $L < M$ then $f_s' < 1$.



An interesting observation is that in between the up-sampler and the down-sampler is a cascade of two LP-filters with different cut-off frequencies, π/L and π/M.

It means that the ”red box” represents one LP-filter with a cut-off frequency which is the numerically smallest one of the two, i.e.,

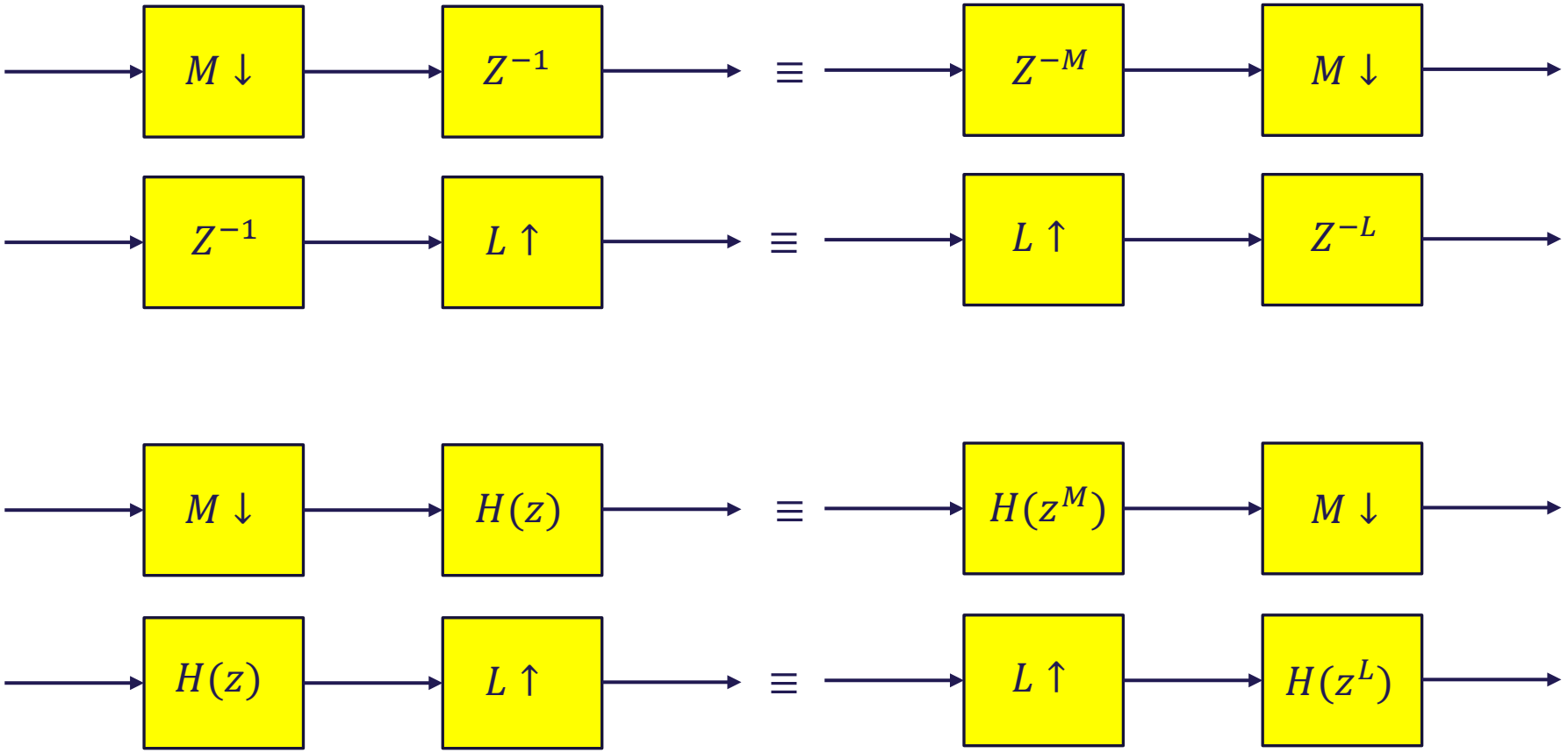
if $f_s' > 1$ then $f_c = \pi/L$, and similarly, if $f_s' < 1$ then $f_c = \pi/M$.

Note that both filters are operated at the same sample-frequency, L/T.

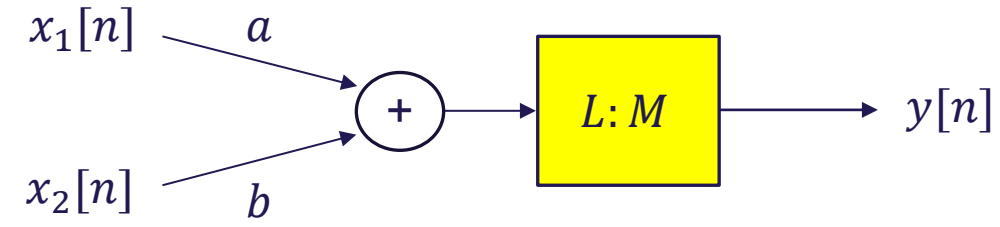
As a consequence, we could simply disgard the ”least strict” LP-filter – the other one has no (gain related) impact on the processing of $x_e[n]$ into $\tilde{x}_i[n]$.

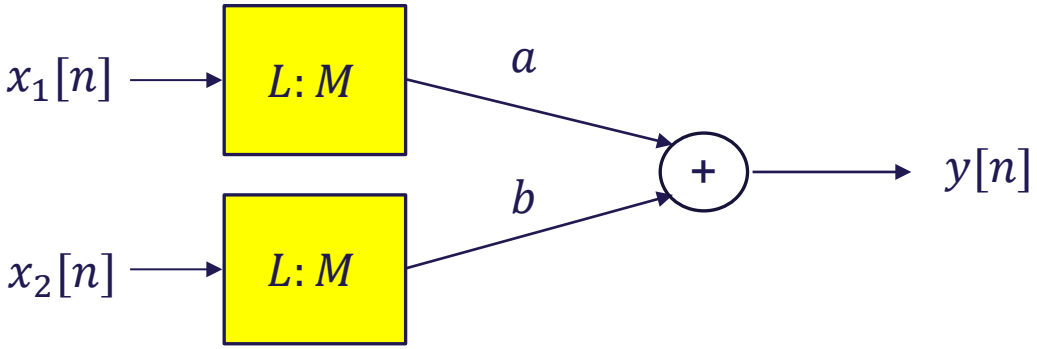
Before discussing the $^{L}/_{M}$ resampler any further, we need some helping tools which specify the relation between filtering and resampling operations – the Noble Identities

You may want to study the proofs yourself – here we will give only the results;



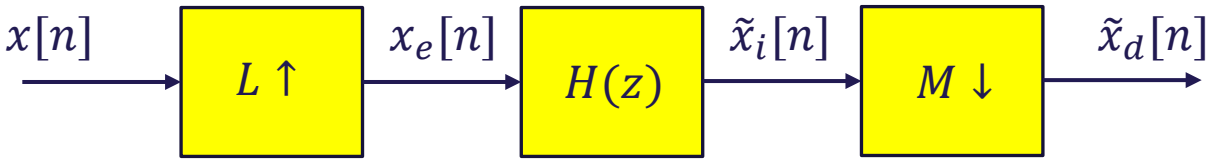
Furthermore, a resampler commutes with arithmetic operations



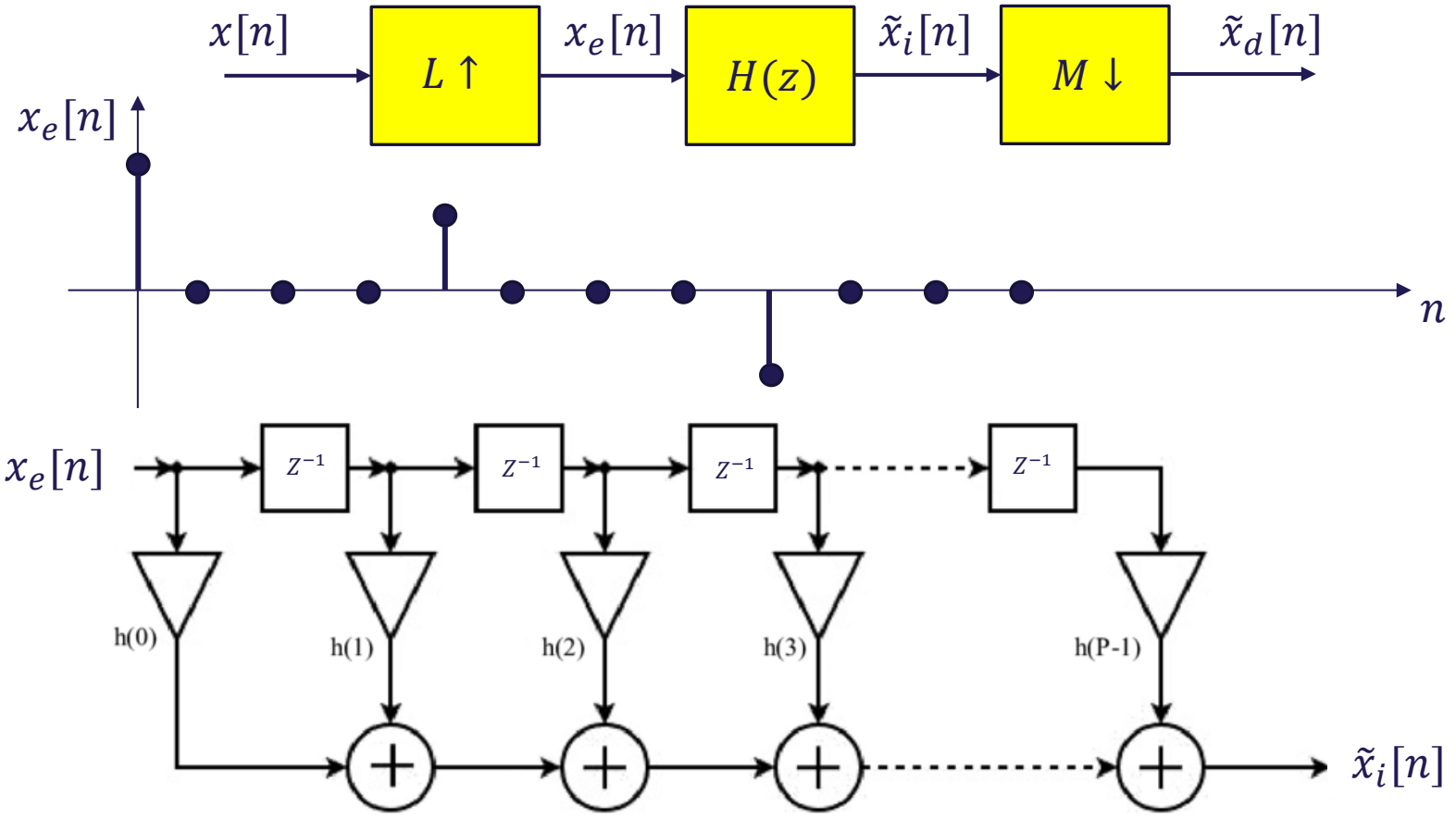


…and now back to the $^{L}/_{M}$ resampler

We argued that the overall system can be reduced to an up-sampler followed by a LP-filter, followed by a down-sampler.

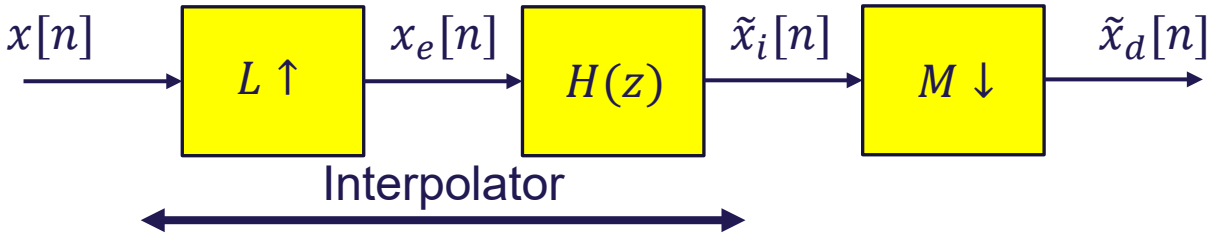


To illustrate the working of such a system, let's assume that $L = 4$ and that $H(z)$ is an FIR filter implemented as a direct form structure.

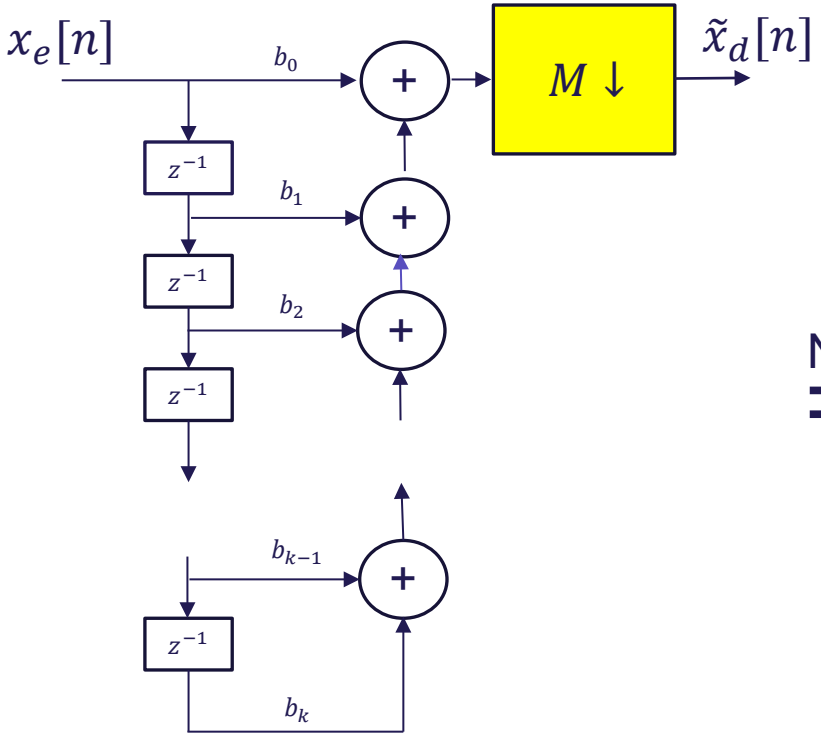


- For every sample fed input the filter, a total of $L-1$ multiplications equal zero…!!
- Only 1 of every M output samples from the filter is transferred to the output $\tilde{x}_d[n]$...!!
- The filter is being executed at the high sample frequency…!!

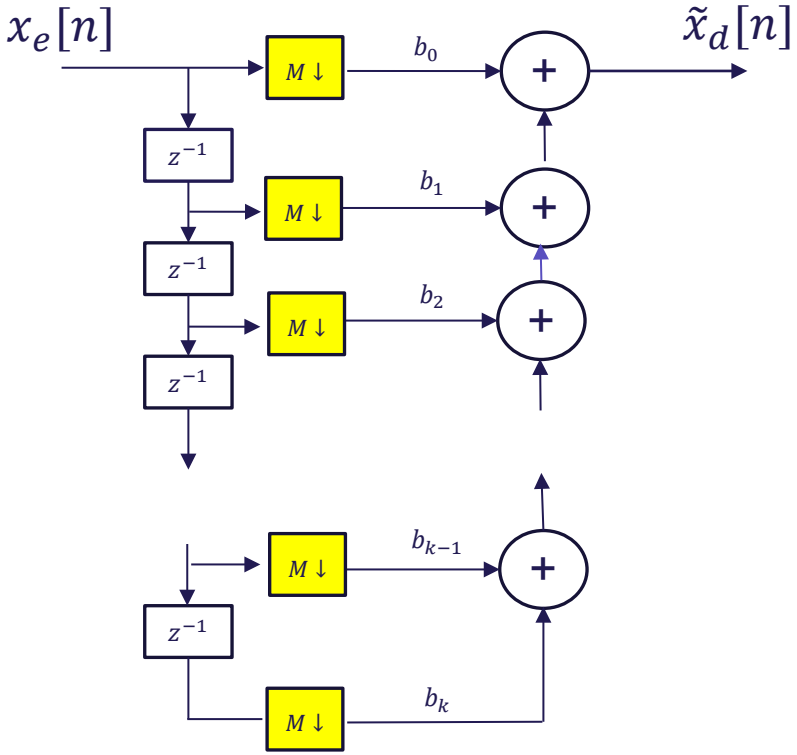
Something needs to be done to save som resources…



Let's look at the Decimator:



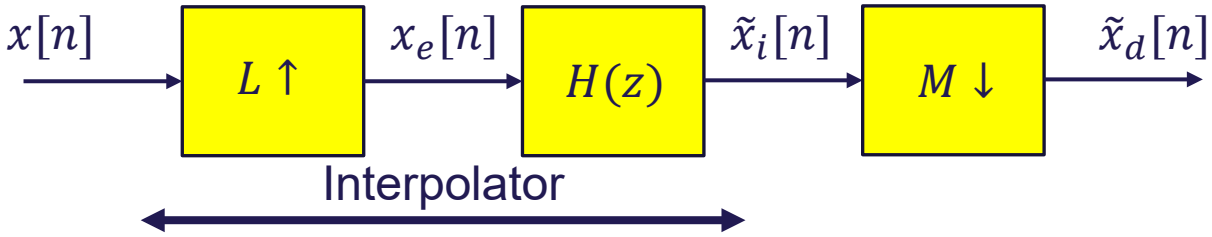
Noble $\Longrightarrow$



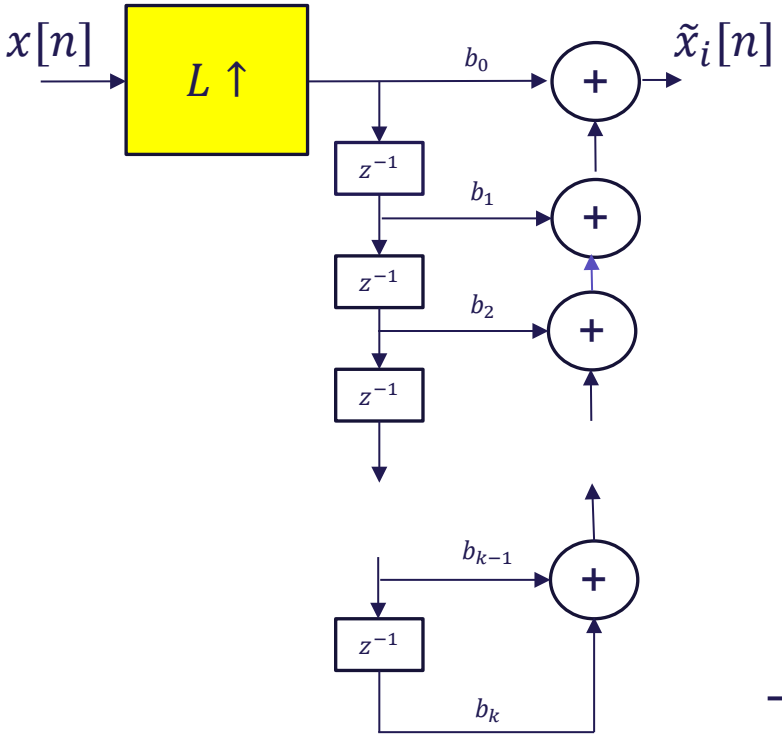
All arithmetic operations are executed at sample freq. Lf_s

Now, all arithmetic operations are executed at sample freq. $(Lf_s)/M$

Furthermore; for every M samples fed into the modified structure, only 1 output sample is computed…!!



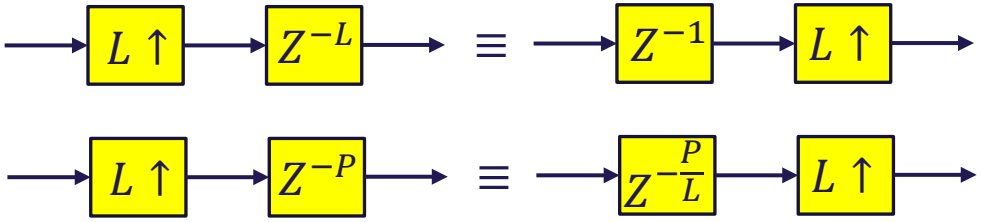
Let's look at the Interpolator:



All arithmetic operations are executed at sample frequency Lf_s. Basically, it means that after up-sampling, the complexity of the filter is increased with a factor of L.

Therefore, if we could only somehow get the up-sampler commuted onto "the other side" of the arithmetic operations, then YES…!

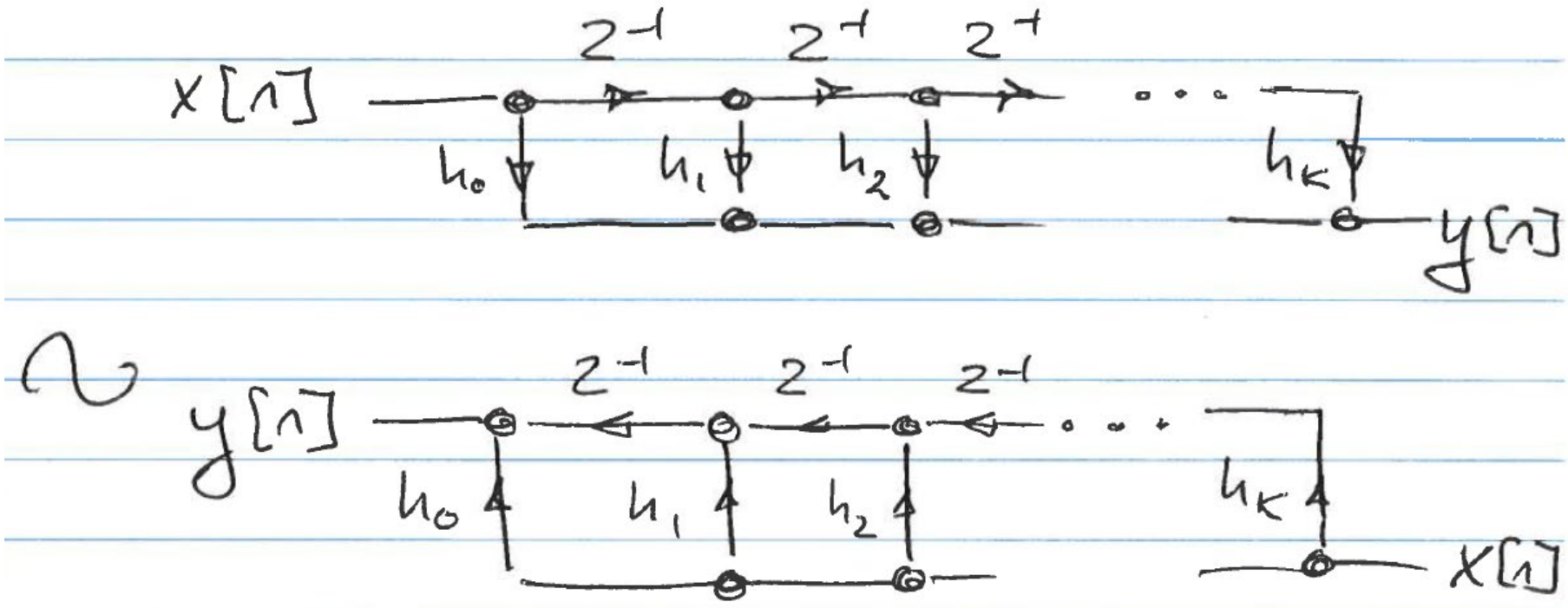
But this, unfortunately, is not easy…



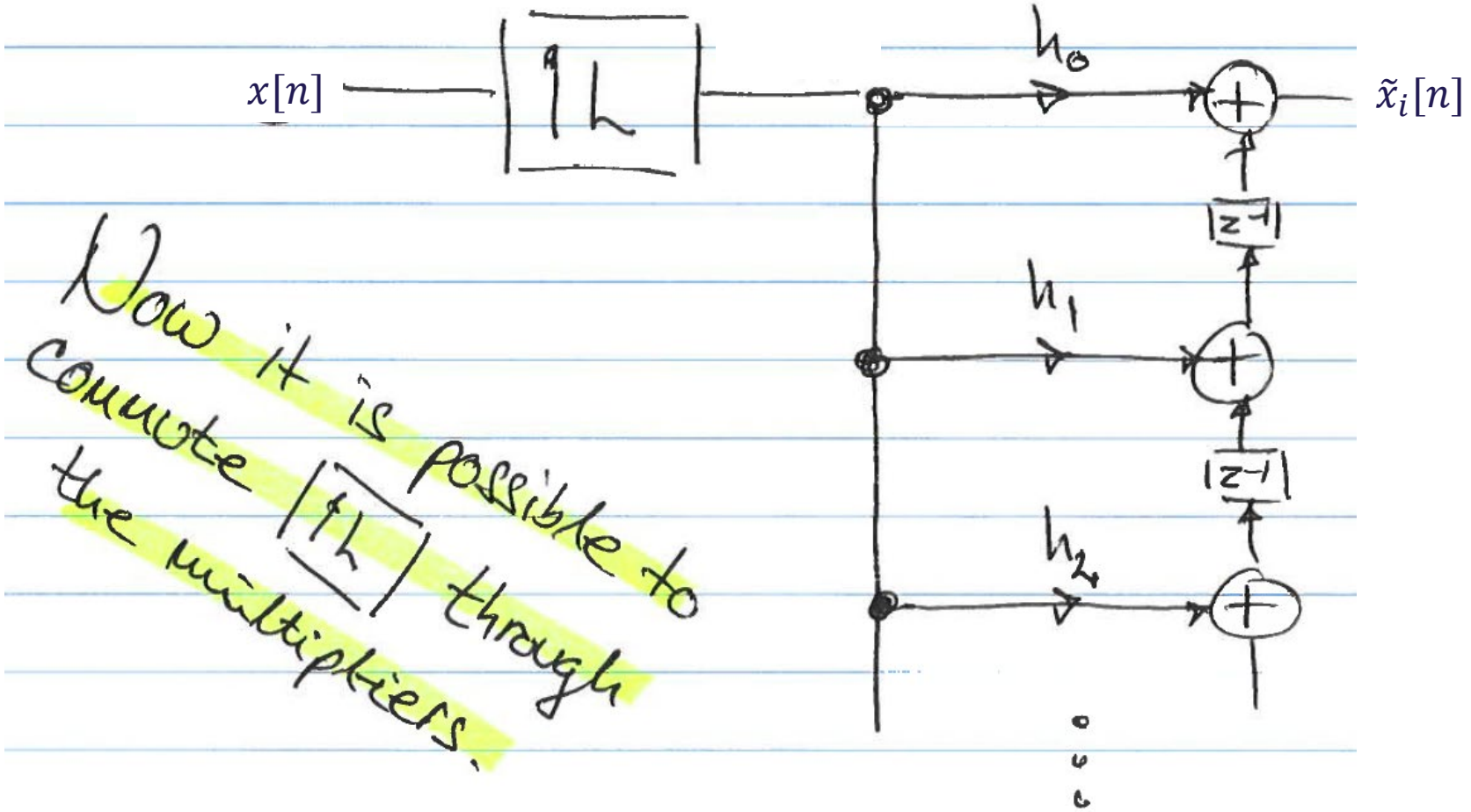
Fractional Delay Operators…!!

Convert the transversal FIR structure to its transposed form – same I/O relation

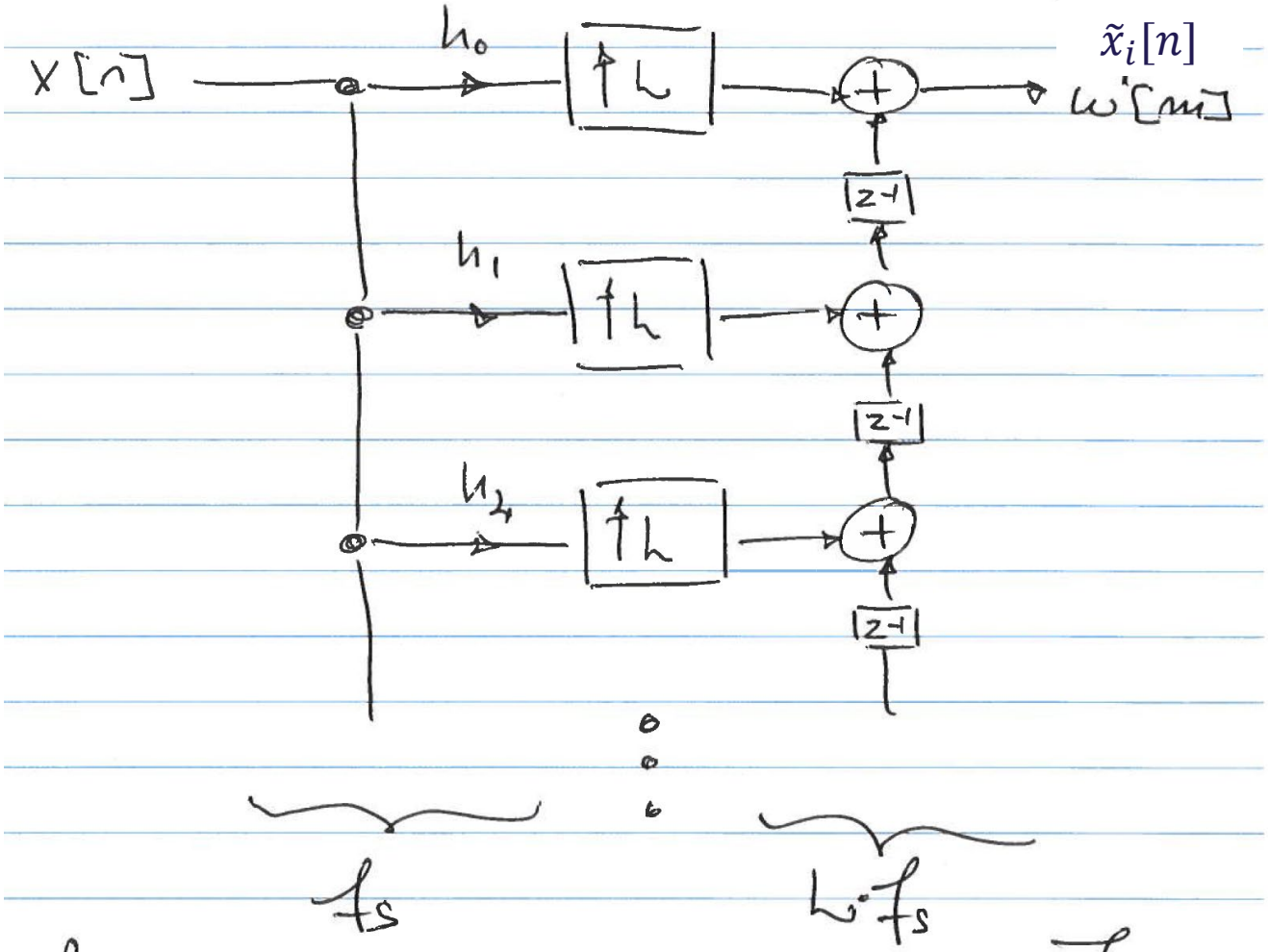
- Draw the structure as a Signal Flow Graph
- Exchange Input and Output
- Revert all signal directions



Now, apply the Noble identities…

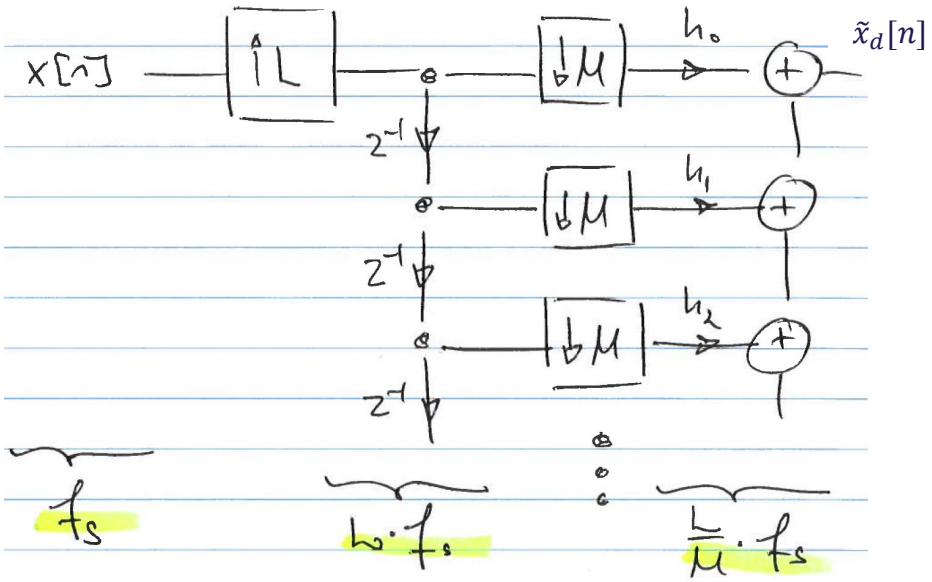


Optimized interpolator with transposed FIR structure

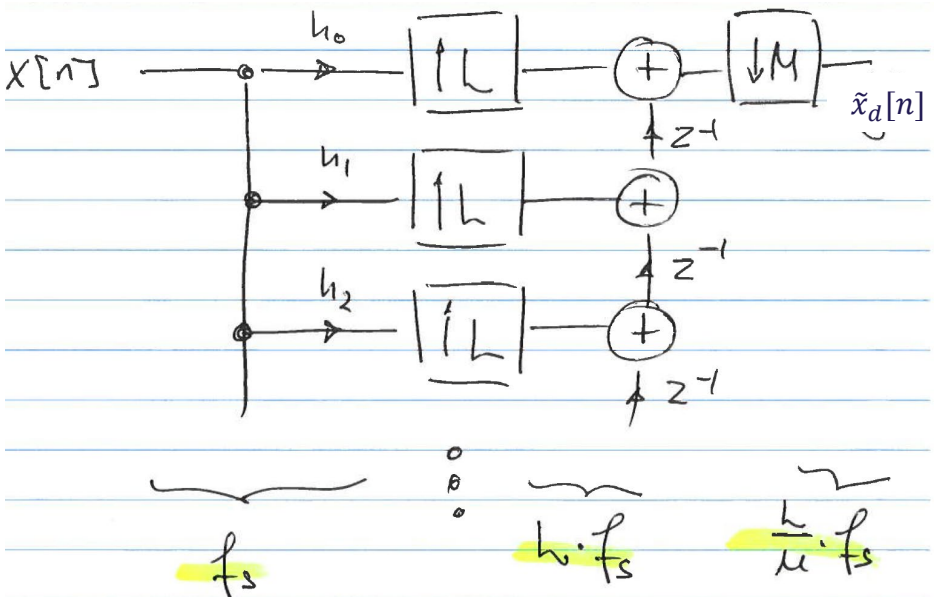


So, all multiplications are now performed at the low sample frequency.

In conclusion we have two possibilities for implementing a ”low computational complexity” fractional resampler



…with optimized decimator



…with optimized interpolator

Choosing the one or the other depends on the actual values of M and L.

A few words on Polyphase Filter – just a teaser…

Given a transfer function $H(z) = \sum_{n=-\infty}^{\infty} h[n] z^{-n}$ -- divide it into even and odd terms

$$H(z) = \{ \cdots h[-4] z^{4} + h[-2] z^{2} + h[0] + h[2] z^{-2} + \cdots \}$$

$$+ \{ \cdots h[-3] z^{3} + h[-1] z + h[1] z^{-1} + h[3] z^{-3} + \cdots \}$$

The "odd term" is now written as

$$z^{-1} \{ \cdots h[-3] z^{4} + h[-1] z^{2} + h[1] + h[3] z^{-2} + \cdots \}$$

Introducing the terms

$$E_0(z) = \sum_{n=-\infty}^{\infty} h[2n] \cdot z^{-n} \qquad \text{(even terms)}$$

$$E_1(z) = \sum_{n=-\infty}^{\infty} h[2n+1] \cdot z^{-n} \qquad \text{(odd terms)}$$

we can now write

$$\boxed{H(z) = E_0(z^2) + z^{-1} E_1(z^2)}$$

In any practical situation, we will consider $h[n]$ causal and finite…

$$H(z) = E_0(z^2) + z^{-1} E_1(z^2)$$

$E_0(z)$ and $E_1(z)$ are the poly-phase components of $H(z)$

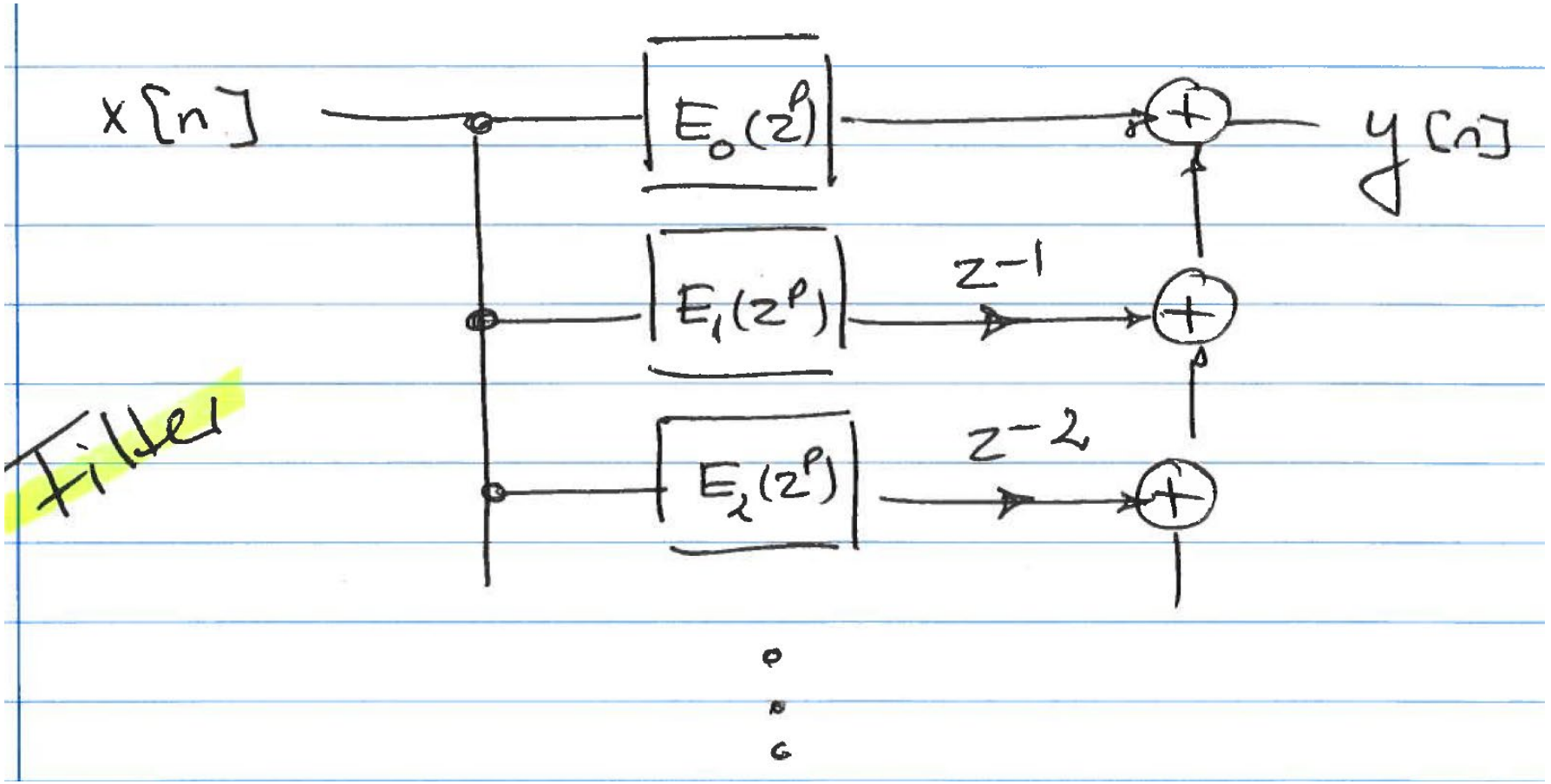
In general, $H(z)$ can be decomposed into the P-fold poly-phase decomposition

$$H(z) = \sum_{k=0}^{P-1} z^{-k} E_k(z^P)$$

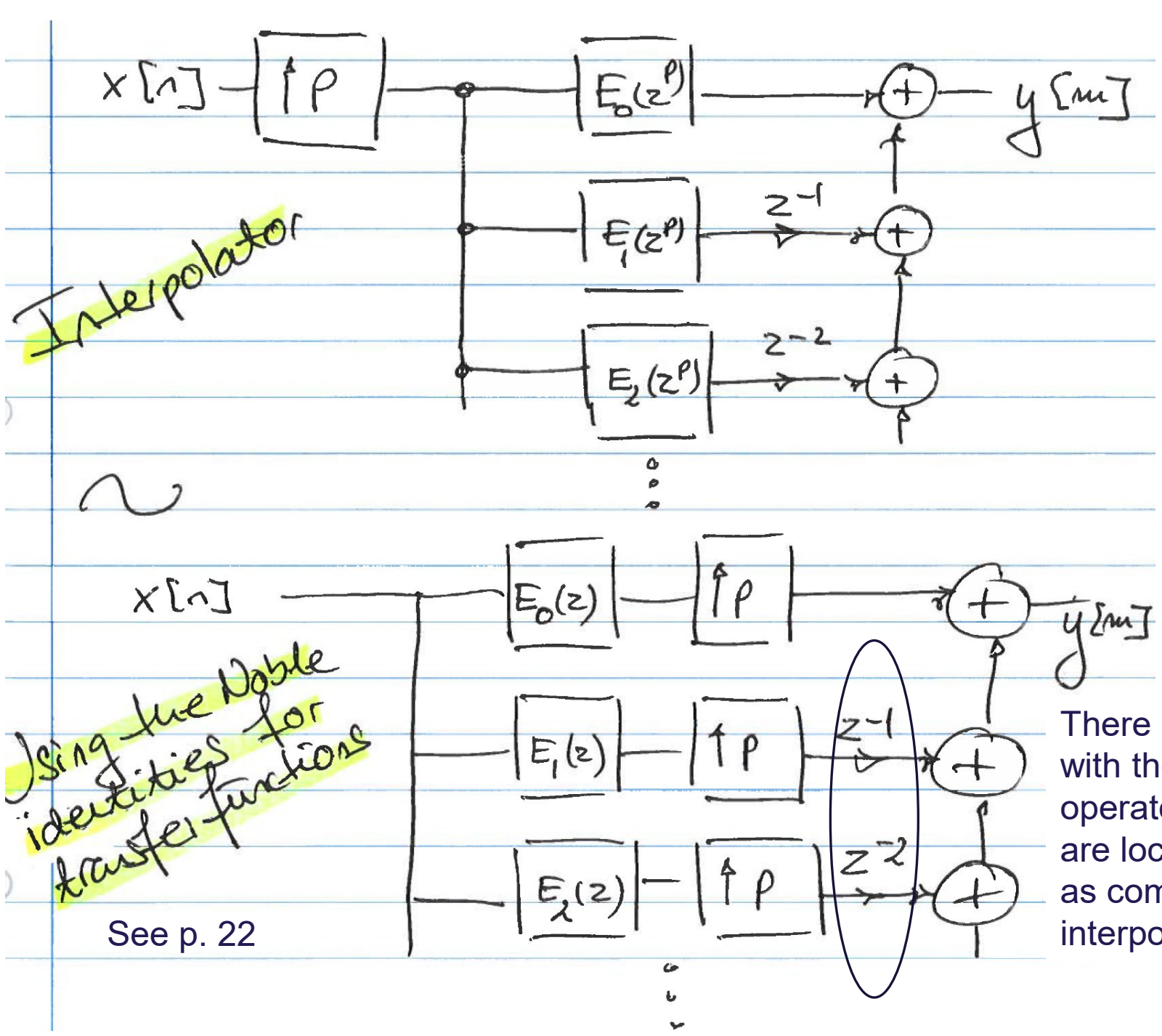
$$\Downarrow$$

$$H(z) = E_0(z^P) + z^{-1} E_1(z^P) + z^{-2} E_2(z^P) + \cdots$$

Using P-fold Poly-phase decomposition, a filter can be re-structured as;



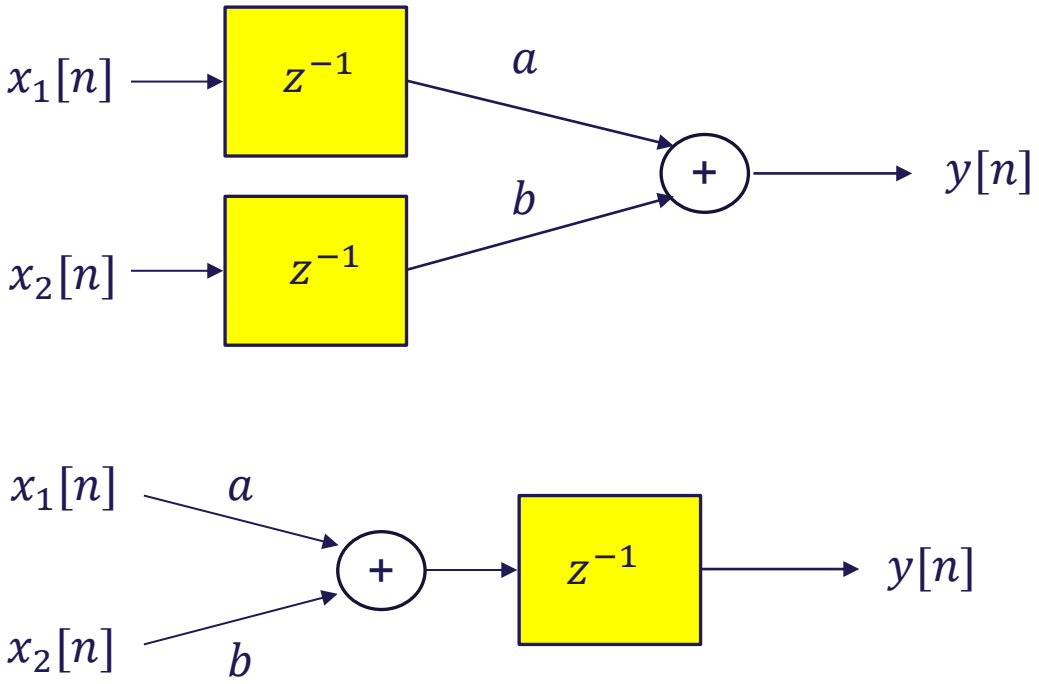
The idea now is to up-sample the input signal $x[n]$…



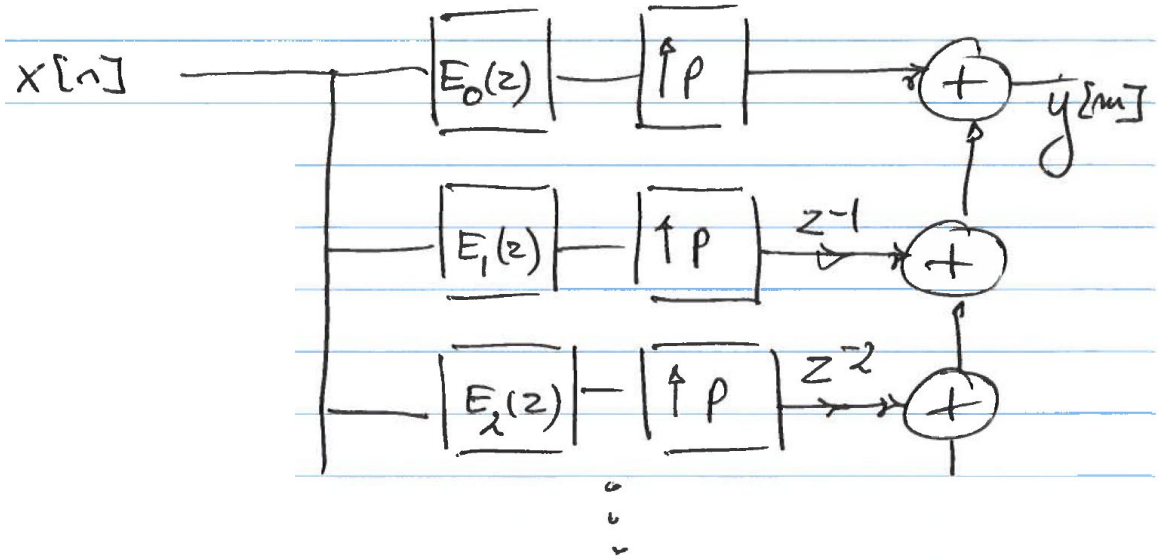
See p. 22

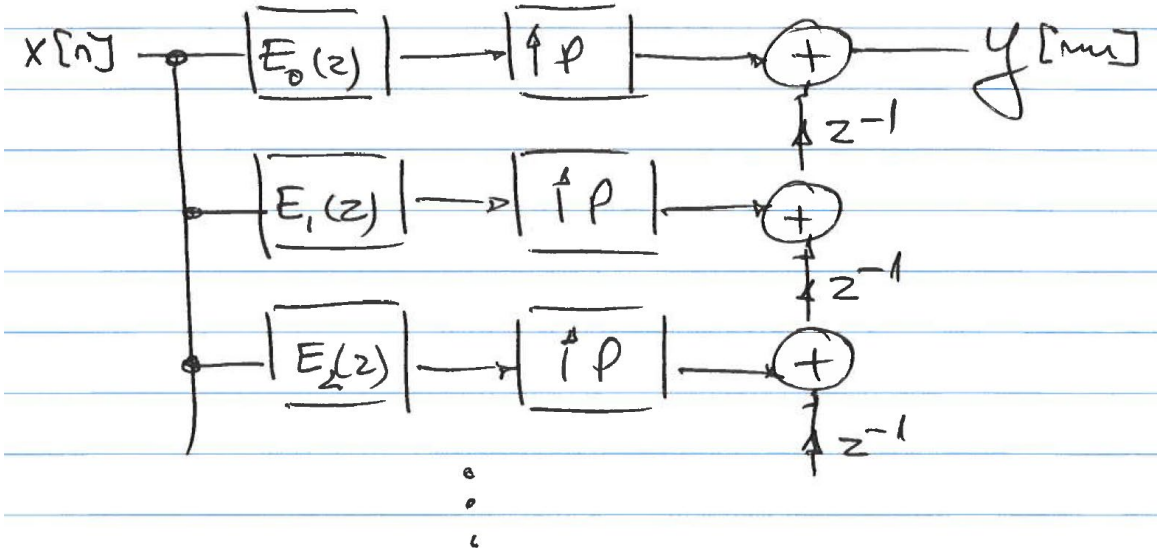
There is an issue with these delay operators – they are located different as compared to the interpolator on p. 31

Introducing the ”rules of re-timing”

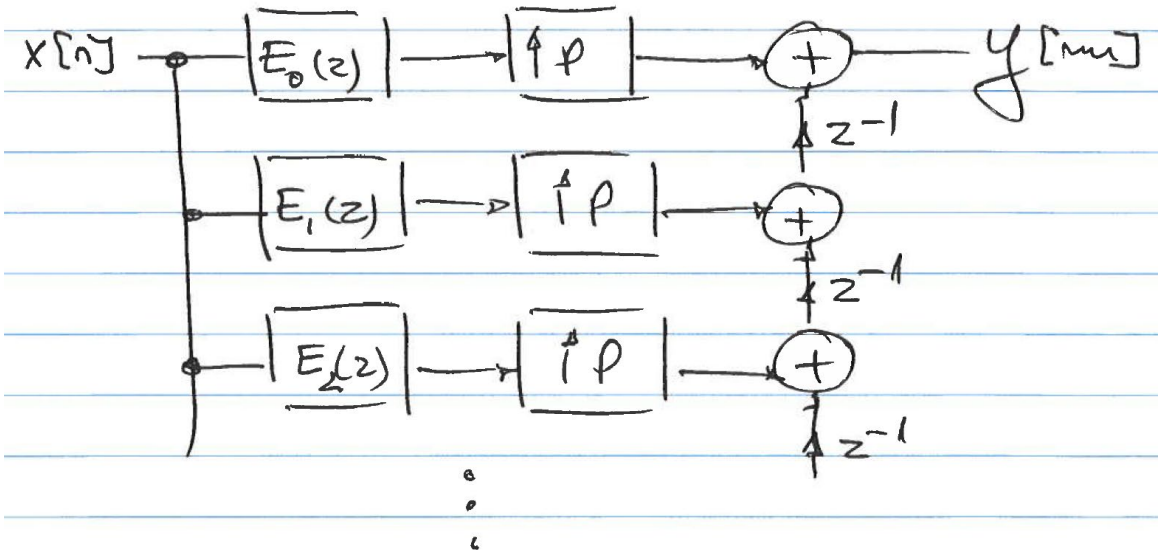


The interesting thing here is the ”addition” which can be considered as a ”collection node”. The timing on input and output should match..!





This structure is known as the Poly-phase structure for the $1:P$ interpolator



An interesting observation about this filter is that each poly-phase branch contributes samples to the output $y[m]$ for different sample times…

Therefore, instead of implementing the delay-line on the output, a much more efficient method select the right samples is the ”commutator model”.

The Poly-phase $1:P$ interpolator implemented with the commutator.