Lecture

8

ECE216B

Digital Filters

Prof. Dejan Marković

ee216b@gmail.com

Agenda

Example: radio systems [1]

- Band-select filters
- Adaptive equalization
- Decimation/interpolation

Filter types:

- Direct
- Recursive
- Multi-rate

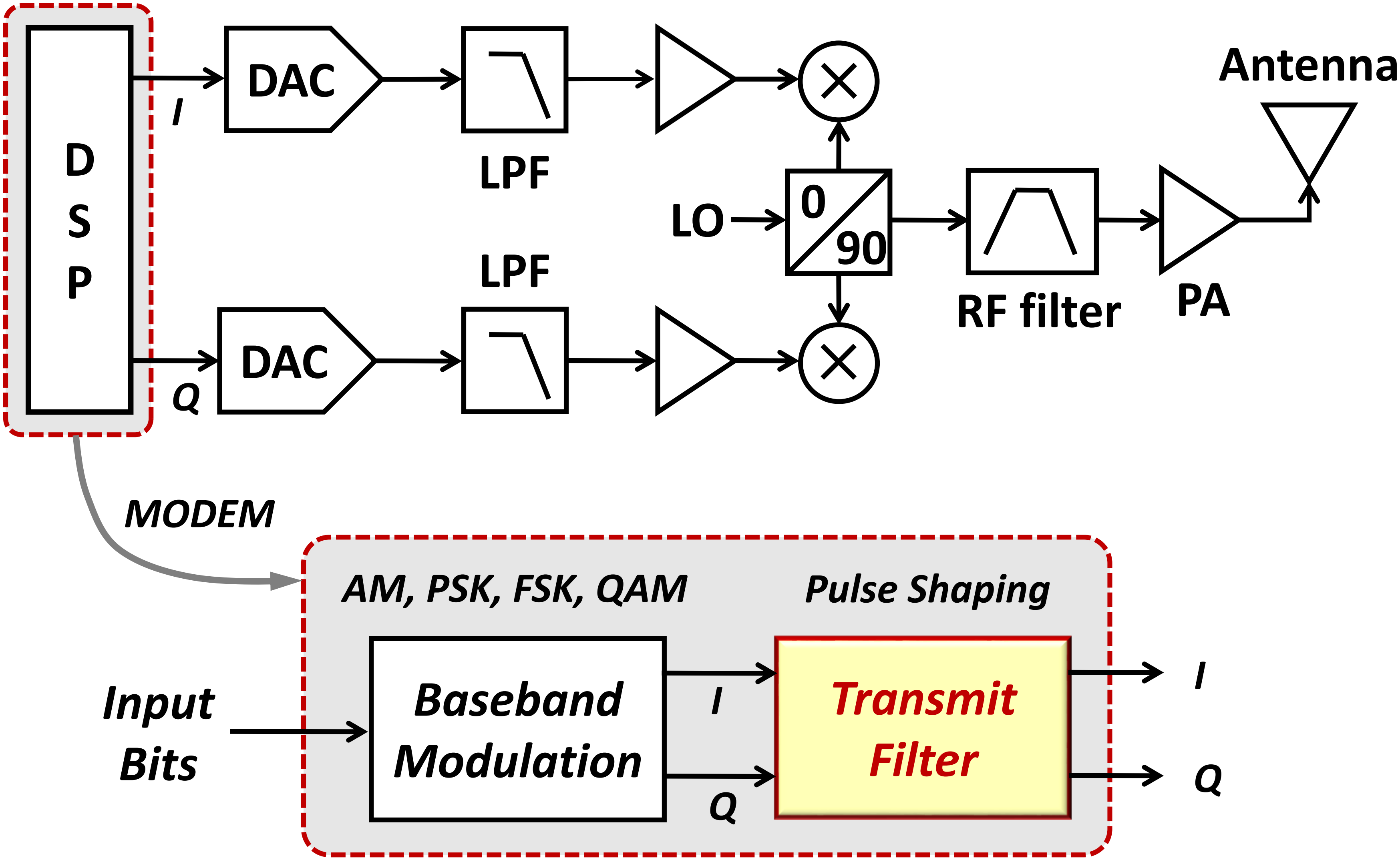
Implementation

- Conventional
- Distributed arithmetic

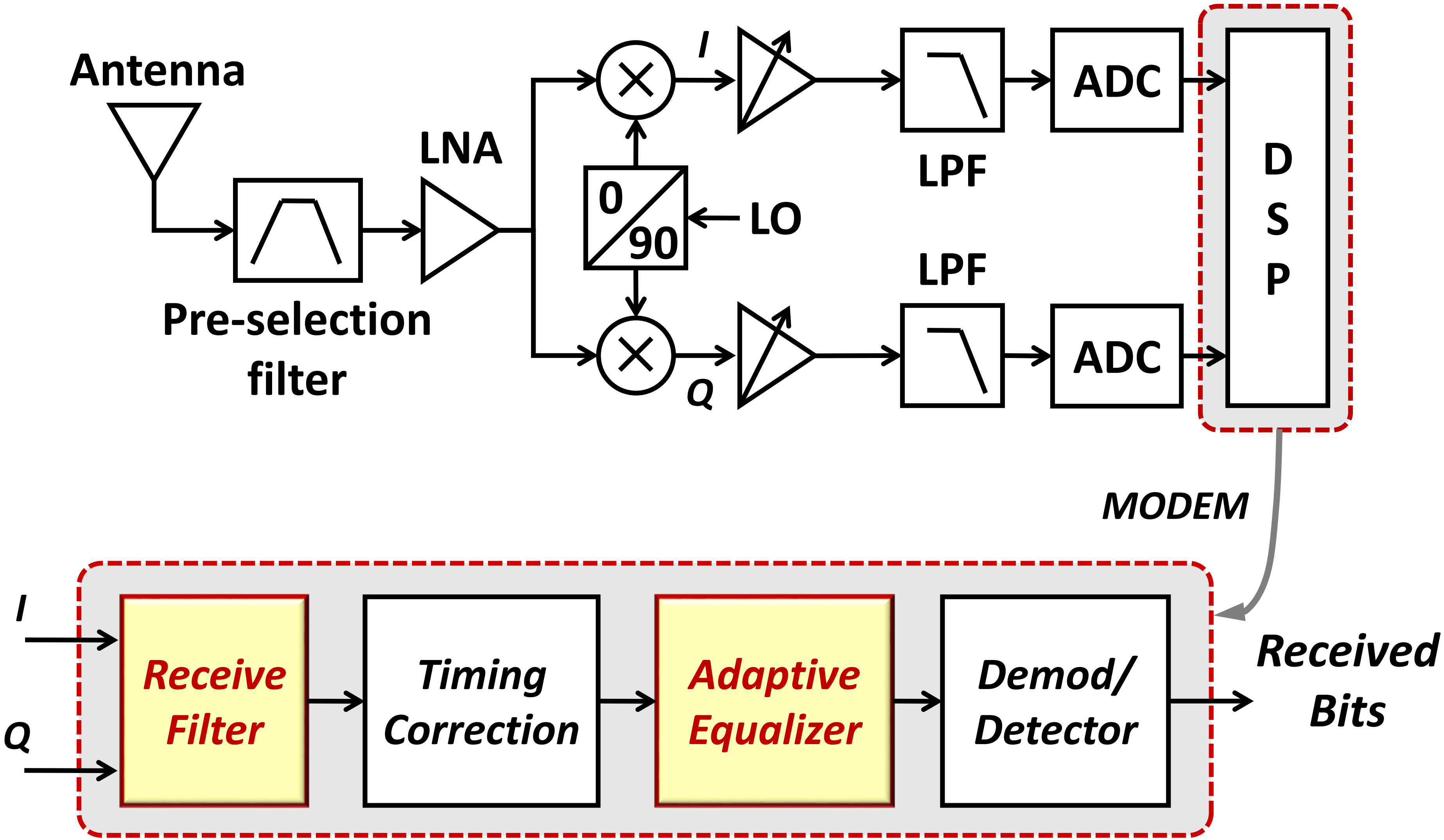
[1] J. Proakis, Digital Communications, (3rd Ed), McGraw Hill, 2000.
[2] A.V. Oppenheim and R.W. Schafer, Discrete Time Signal Processing, (3rd Ed), Prentice Hall, 2009.
[3] J.G. Proakis and D.K. Manolakis, Digital Signal Processing, (4th Ed), Prentice Hall, 2006.
[4] K.K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, Wiley, 1999.

Filter Design for Digital Radios

Radio Transmitter



Radio Receiver



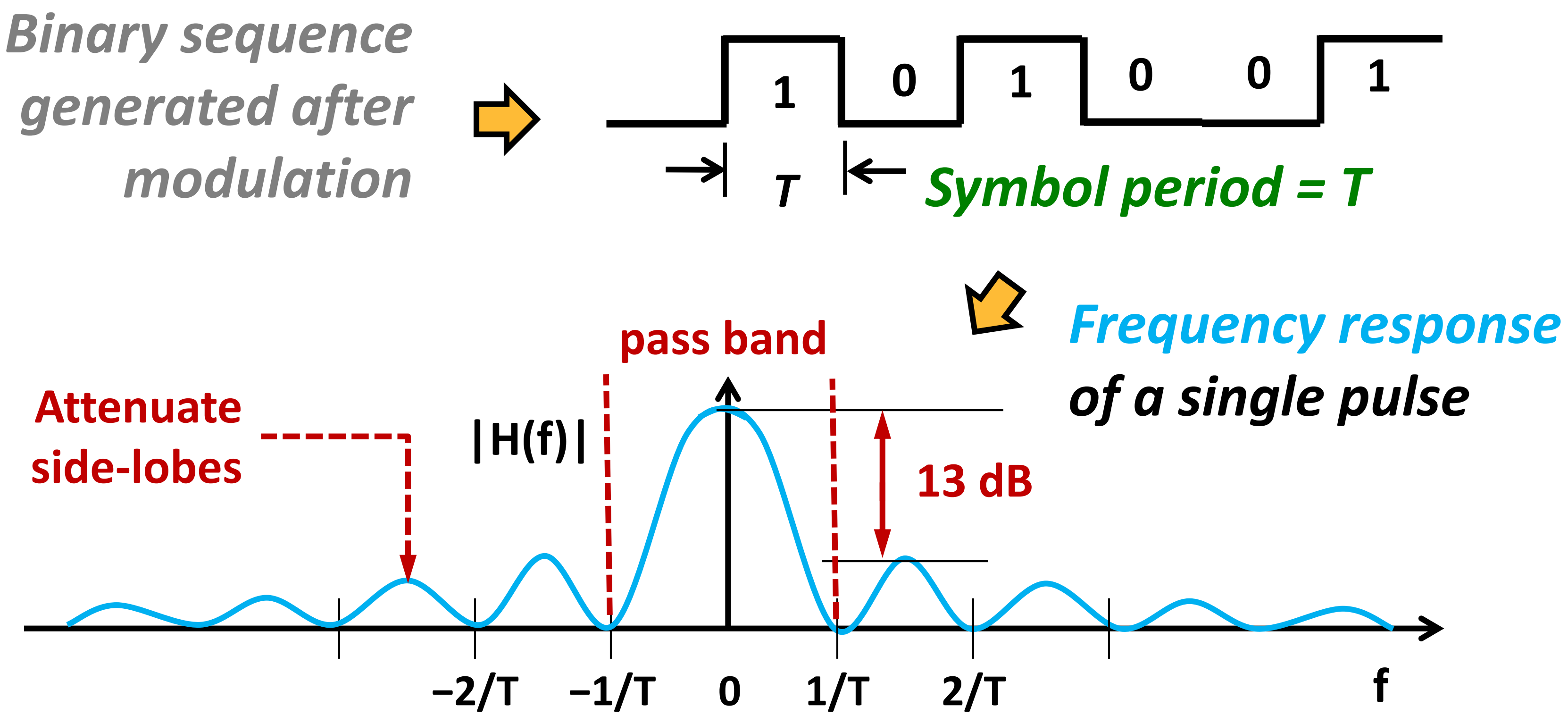
Design Procedure

Assume: Modulation, T_s (symbol period), bandwidth

- **Algorithm design**
  - **Transmit / receive filters**
  - Modulator
  - Demodulator
  - Detector
  - Timing correction
  - **Adaptive equalizer**
- **Implementation architecture**
- **Wordlength optimization**
- **Hardware mapping**

Signal Bandwidth Limitation

- **Modulation generates pulse train of zeros and ones**
  - Its frequency response is not band-limited
  - Filter before Tx to restrict bandwidth of symbols

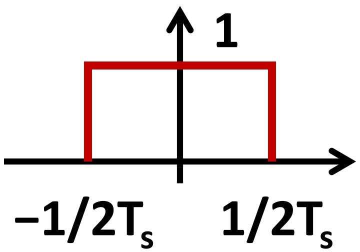


Ideal Filter

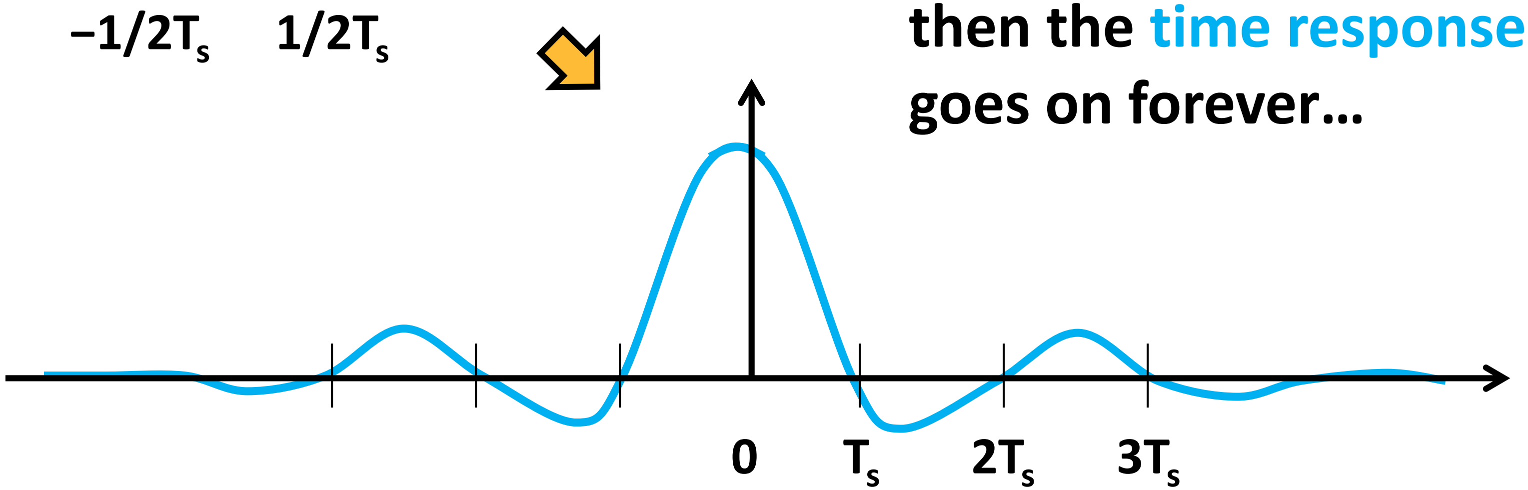
Sample rate = f_s

Baseband BW = $f_s/2$ (pass band BW = f_s)

- **If we band-limit to the minimum possible amount $1/2T_s$,**

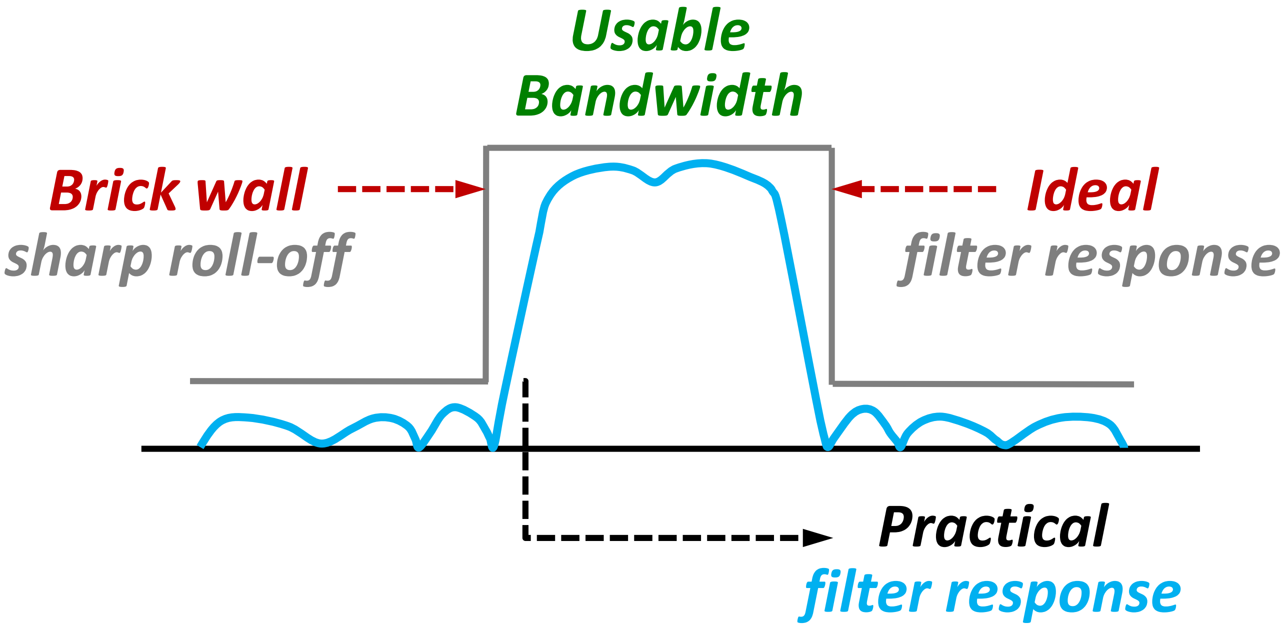


then the time response goes on forever…



Practical Transmit / Receive Filters

- **Tx:** restrict the Tx signal BW to a specified value
- **Rx:** extract the signal from a specified BW of interest

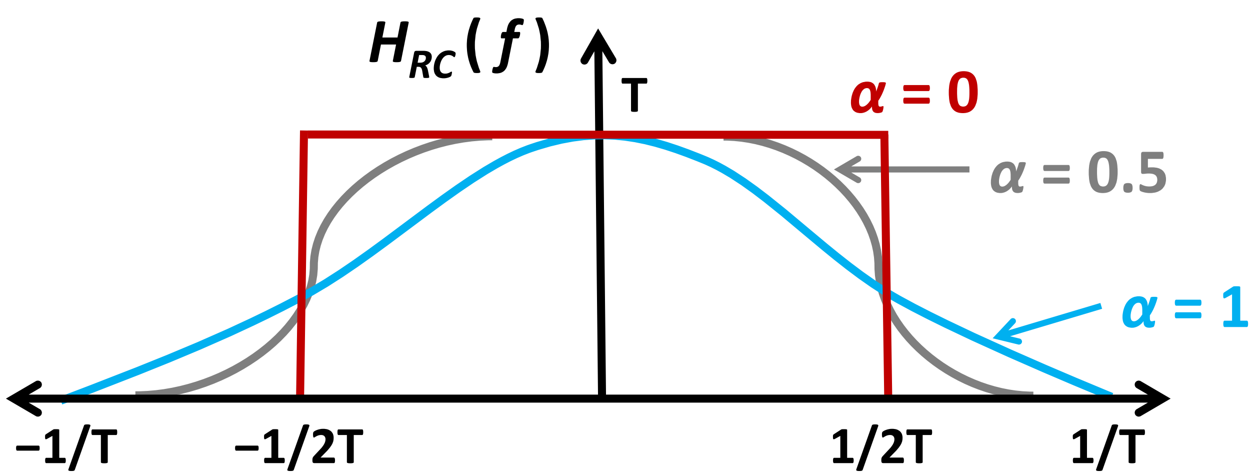


Ideal filter** has **infinitely long impulse response.

Raised-cosine filter** is a **practical realization.

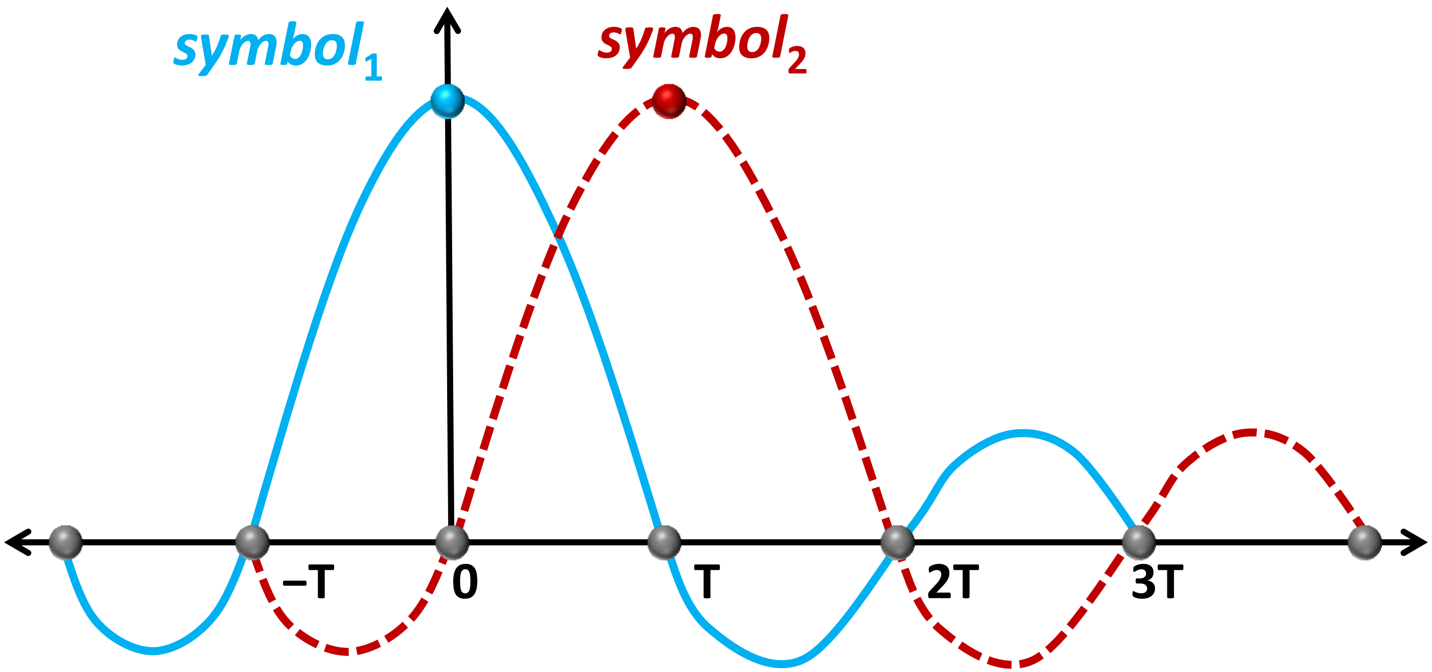
Raised-Cosine Filter: Frequency Response

$$H_{RC}(f) = \begin{cases} 1 \Leftrightarrow |f| \le \dfrac{1-\alpha}{2T_s} \\ \dfrac{T_s}{2}\left[1 + \cos\dfrac{\pi T_s}{\alpha}\left(|f| - \dfrac{1-\alpha}{2T_s}\right)\right] \Leftrightarrow \dfrac{1-\alpha}{2T_s} < |f_s| < \dfrac{1+\alpha}{2T_s} \\ 0 \Leftrightarrow |f| > \dfrac{1+\alpha}{2T_s} \end{cases}$$



Raised-Cosine Filter: Time Response

- **Time-domain pulse shaping with raised-cosine filters**
- **No contribution of adjacent symbols at the $k{\cdot}T_s$ instants**
  - No inter-symbol interference with this pulse shaping



Square-Root Raised-Cosine Filter

Split the filter between Tx & Rx

$$H_{RC}(f) = H_{Tx}(f) \cdot H_{Rx}(f)$$

$$H_{Tx}(f) = H_{Rx}(f) = \sqrt{H_{RC}(f)}$$

- **Choose sample rate for the filter:**
  - f_{sample} equal to D/A frequency on the Tx side and A/D frequency on the Rx side
  - If $f_{D/A} = f_{A/D} = 4 \cdot (1/T_{symbol})$,

$$\Rightarrow \quad h_{Tx}(n) = \sum \sqrt{H_{RC}(4m / NT_s)} \cdot e^{j(2\pi mn)/N}$$

for $-(N-1)/2 \leq n \leq (N-1)/2$

***Impulse response is finite:* implement as FIR filter**

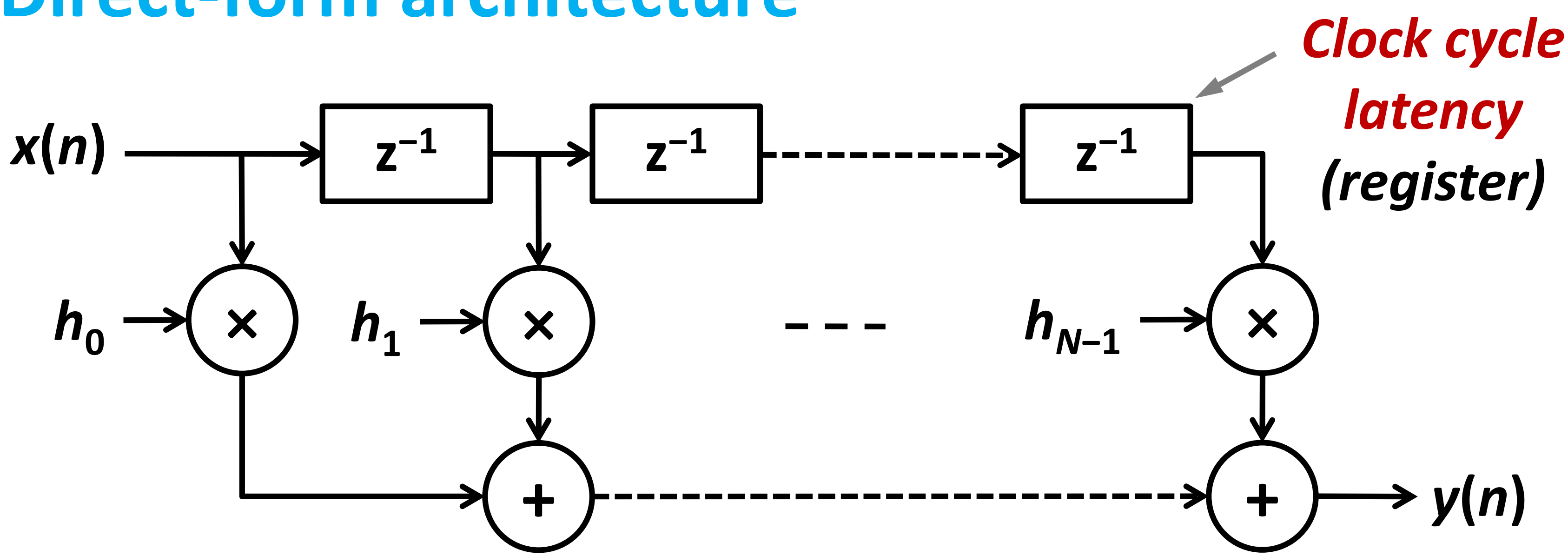
Direct (FIR) Filters

Implementing the Filter

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) + \ldots + h_{N-1} x(n-N+1)$$

- **An *N*-tap filter is a series of *multiply* and *add* operations**

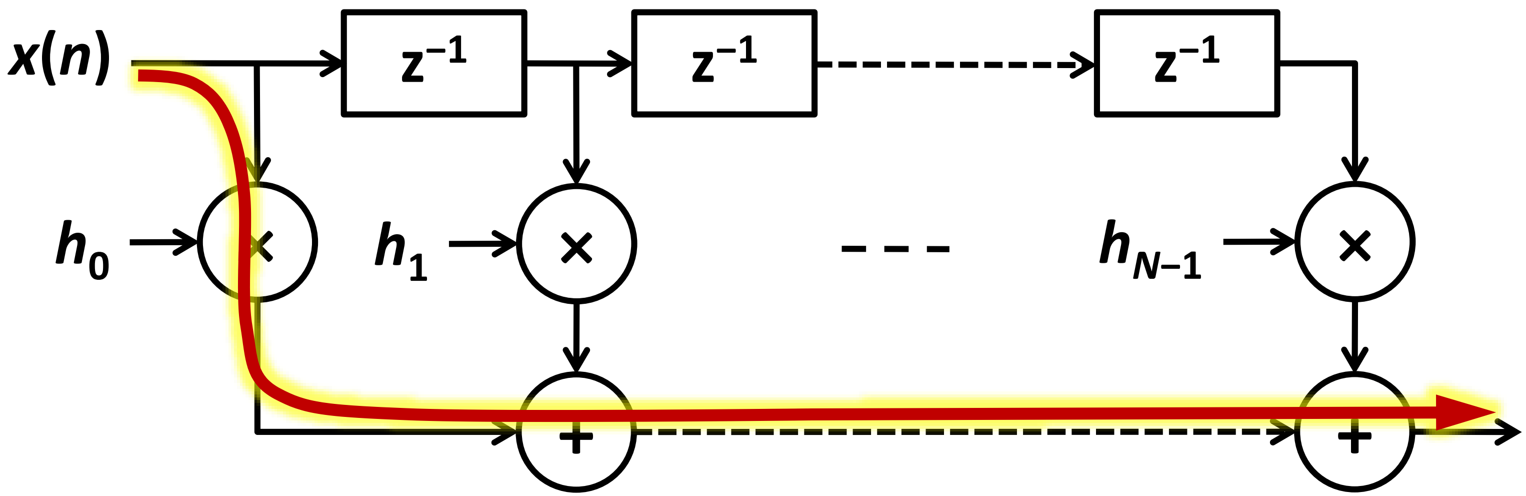
Direct-form architecture



Direct-Form FIR Filter: Critical Path

Critical path = t_{mult} + (N–1) · t_{add}

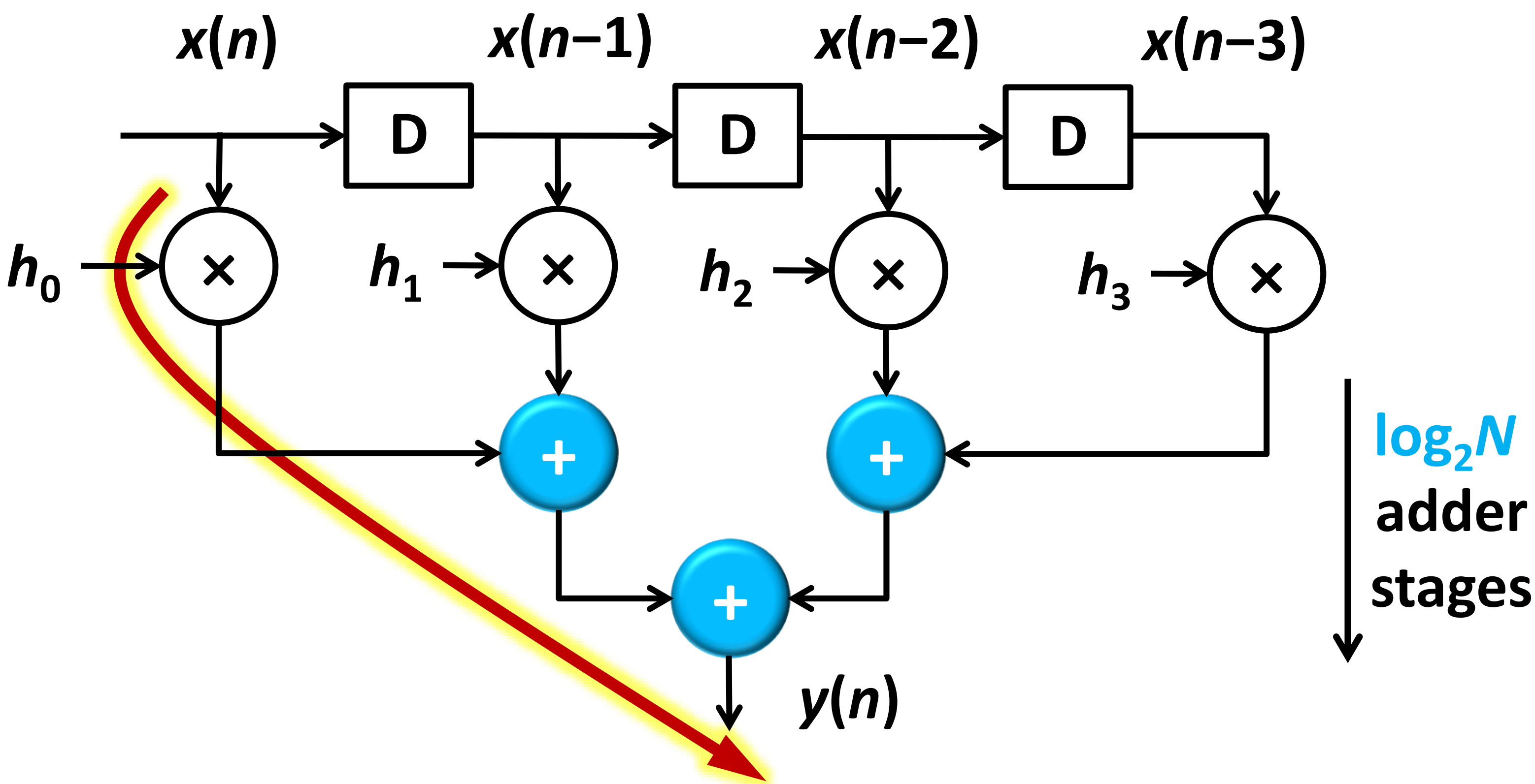
- **Critical-path delay proportional to filter order**
  - Suitable for small number of taps



Multi-Operand Addition

Critical path = $t_{mult} + \log_2(N) \cdot t_{add}$

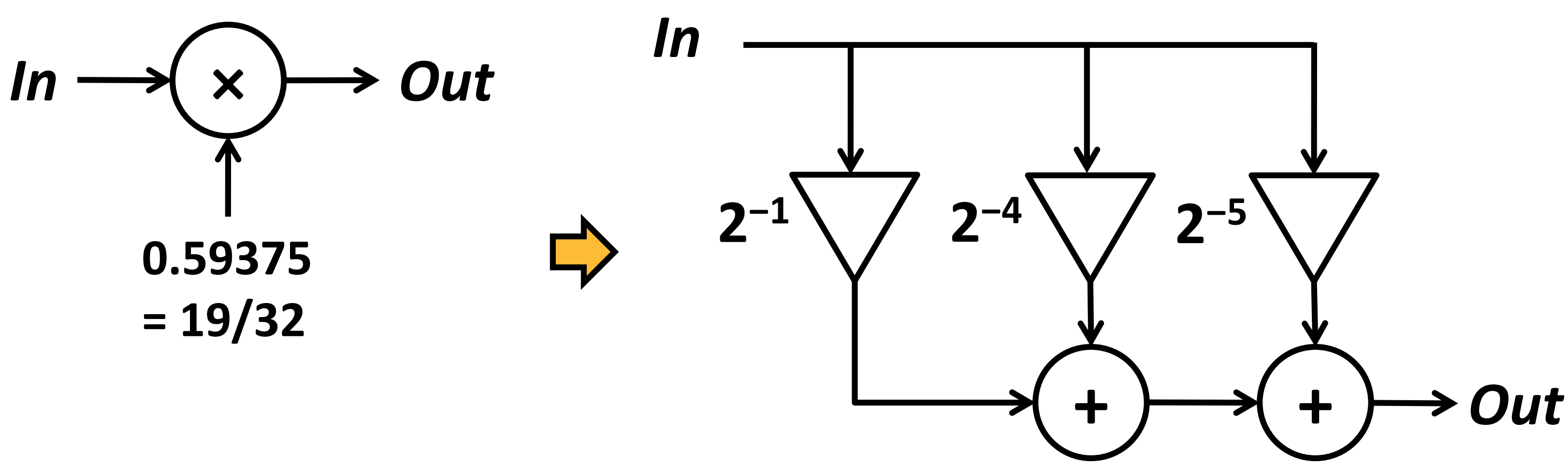
instead of $N-1$



Use an adder tree instead of a chain

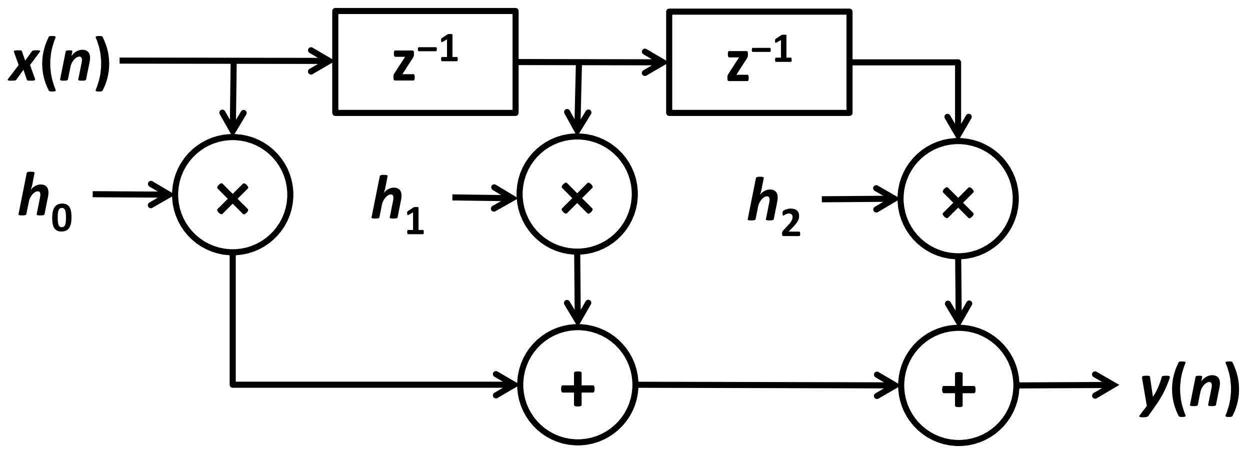
Multiplier-less FIR Filter Implementation

- **Low-complexity power-of-two based multiplications**
  - Obtained for free by simply shifting data buses
  - Round off multiplier coefficients to nearest power of 2
  - Small performance degradation in most cases

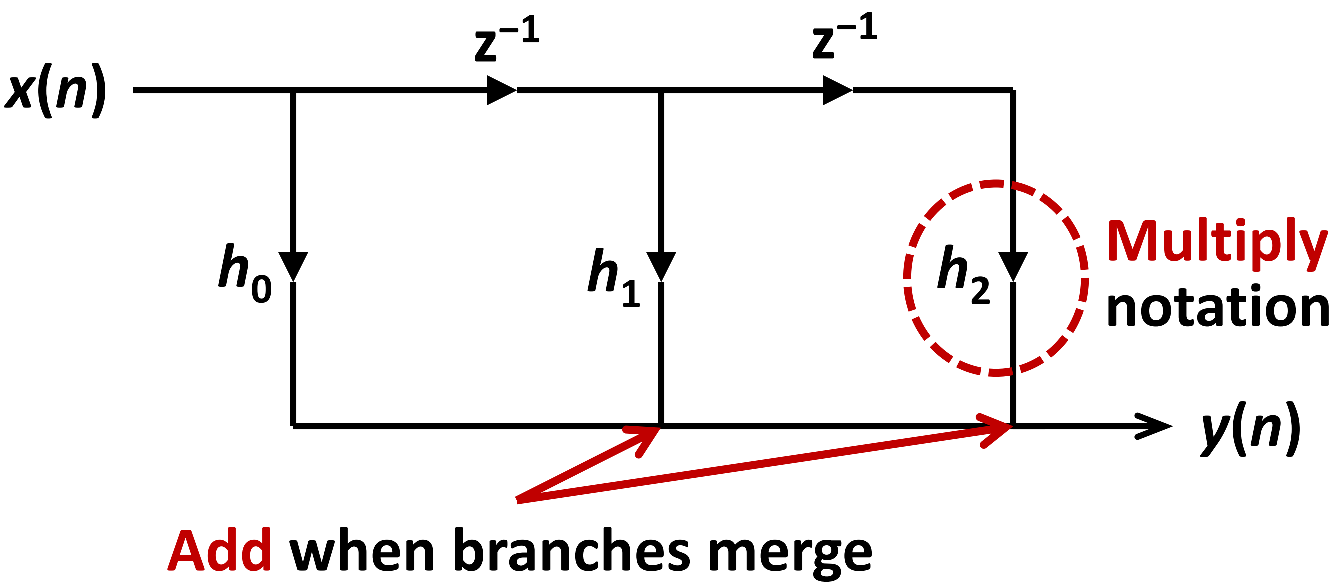


H. Samueli, "An Improved Search Algorithm for the Design of Multiplierless FIR Filters with Powers-of-Two Coefficients," *IEEE Trans. Circuits and Systems,* vol. 36 , no. 7, pp. 1044-1047, July 1989.

FIR Filter: Simplified Notation



A more abstract and efficient notation



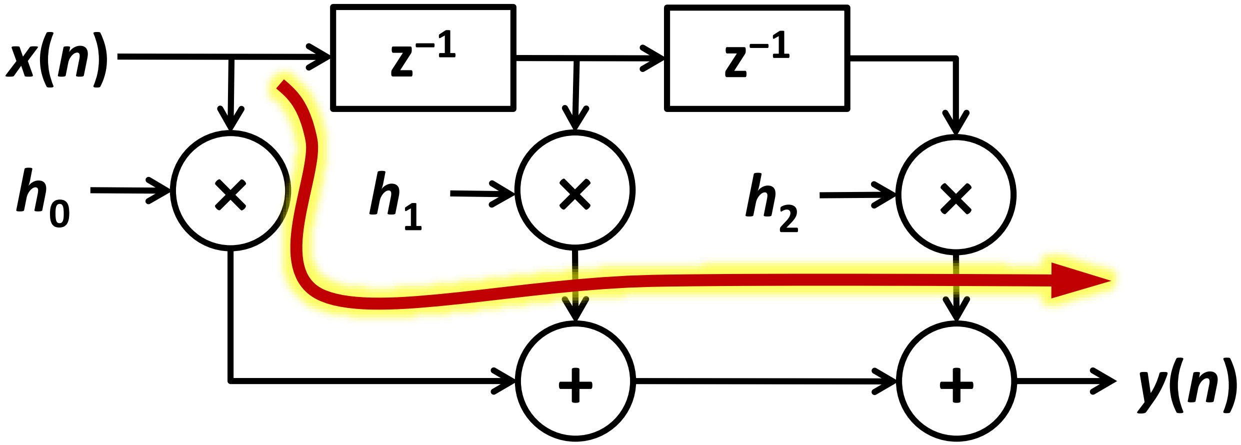
Pipelining

- **Direct-form architecture is throughput-limited**
  - Pipelining can be used to increase throughput
- **Pipelining: adding the same number of delay elements in each forward cut-set**
  - Cut-set: set of edges in a graph that if removed, graph becomes disjoint
  - Forward cut-set: all edges in the cut-set are in the same direction
- **Increases latency**
- **Register overhead (power, area)**

K.K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, Wiley, 1999.

Example 9.1: Pipelining FIR Filter

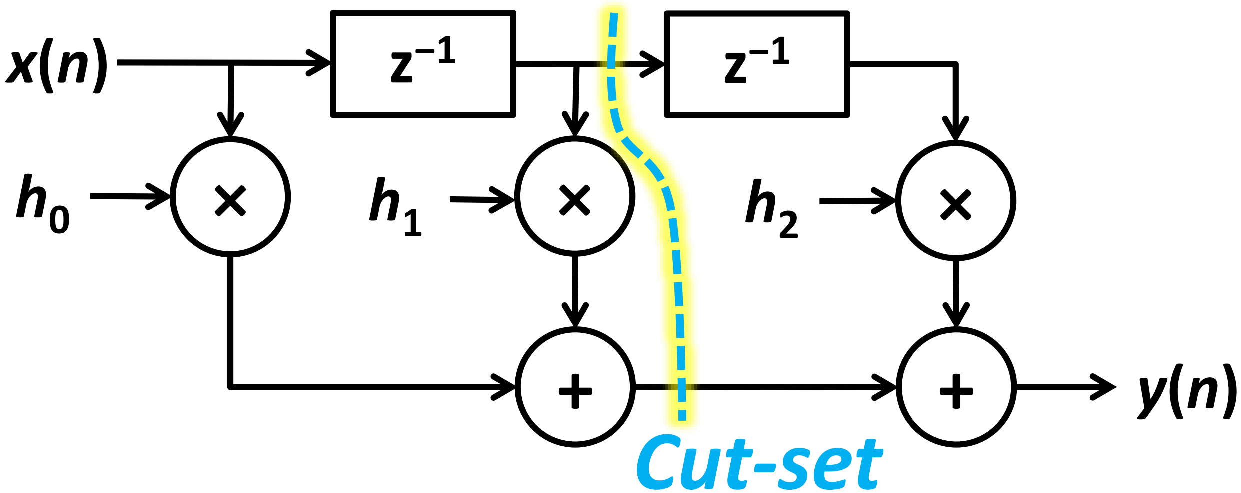
- **Calculate initial critical path**



$$\textit{Critical path} = t_{mult} + 2 \cdot t_{add}$$

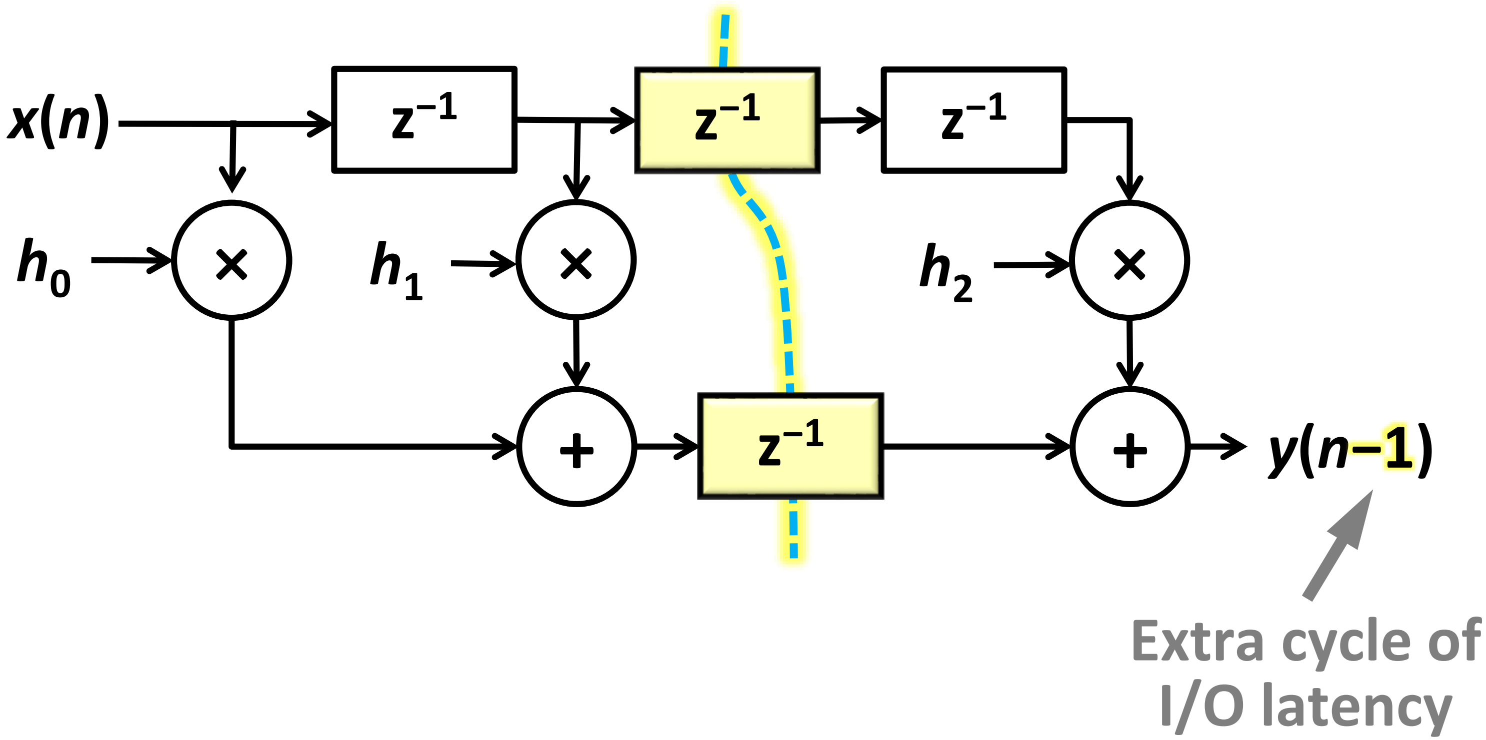
Example 9.1: Pipelining FIR Filter

- **Find forward cut-set**



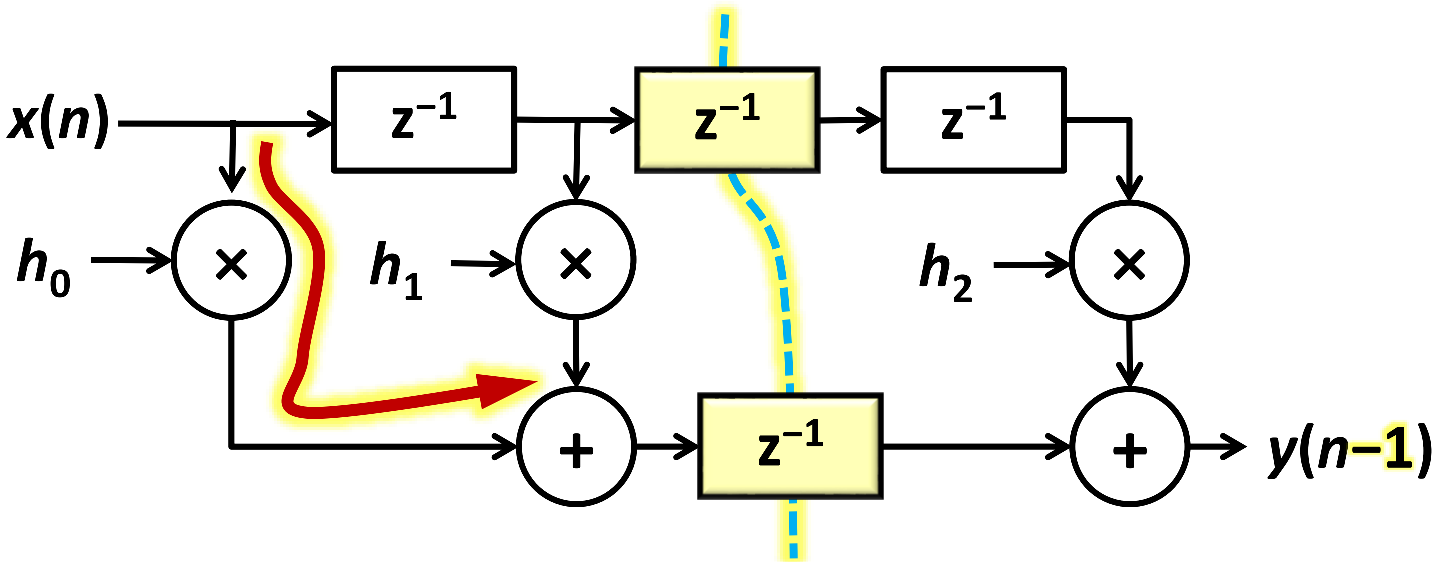
Example 9.1: Pipelining FIR Filter

- **Insert pipeline registers**
  - I/O latency increases



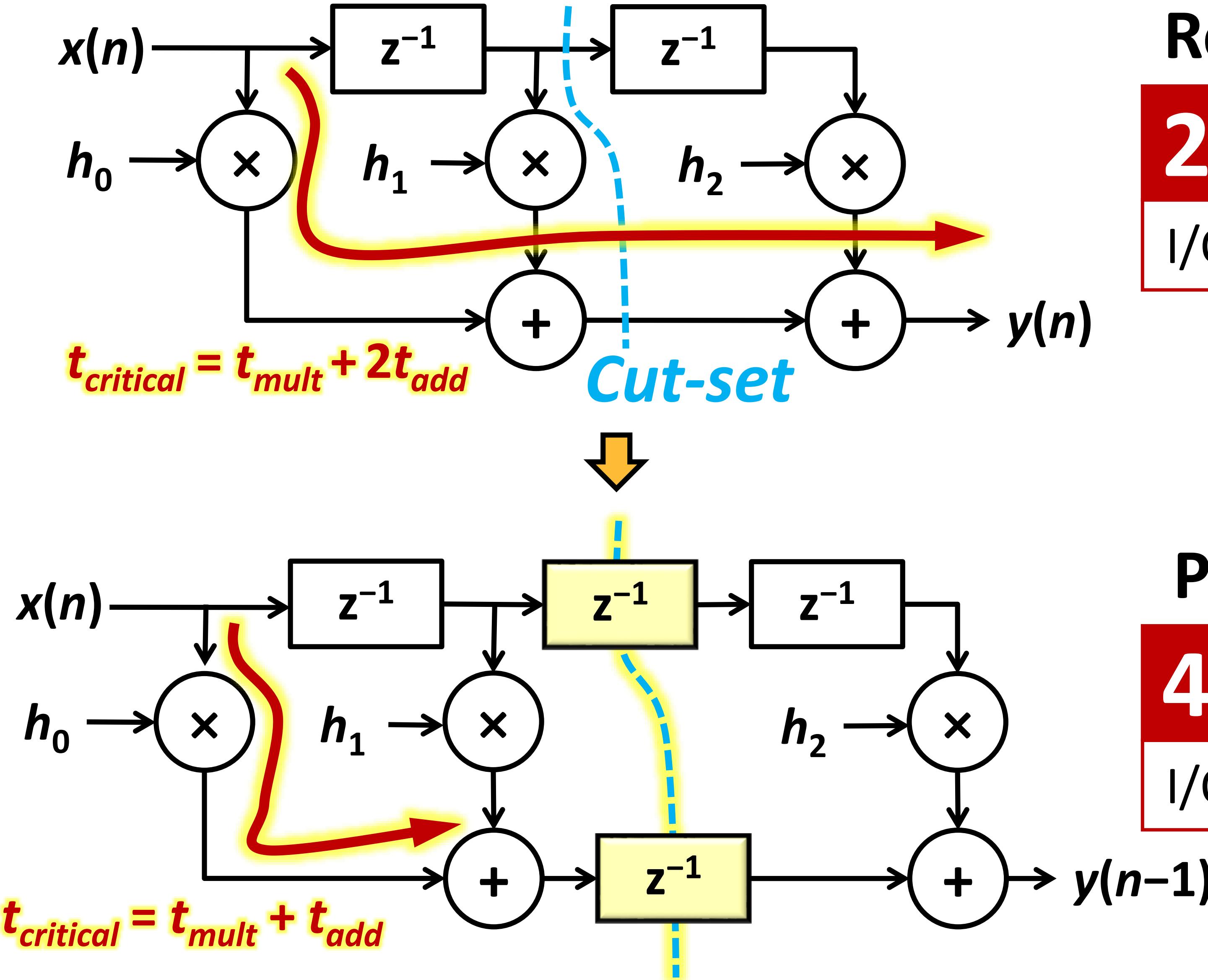
Example 9.1: Pipelining FIR Filter

- **Calculate new critical path**



Critical path = $t_{mult} + t_{add}$

Example 9.1: Pipelining FIR Filter



Reference

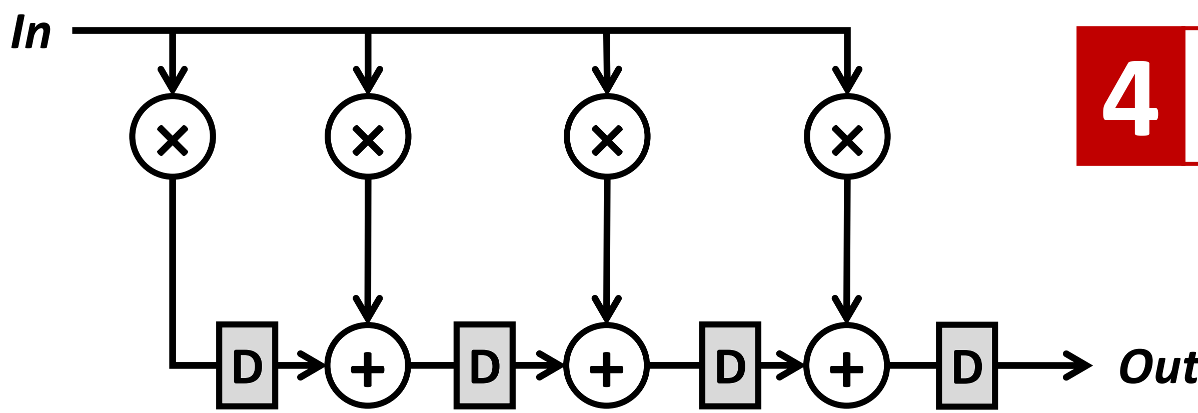
2	registers
I/O latency: **0**	

Pipelined

4	registers
I/O latency: **1**	

High-Level Retiming

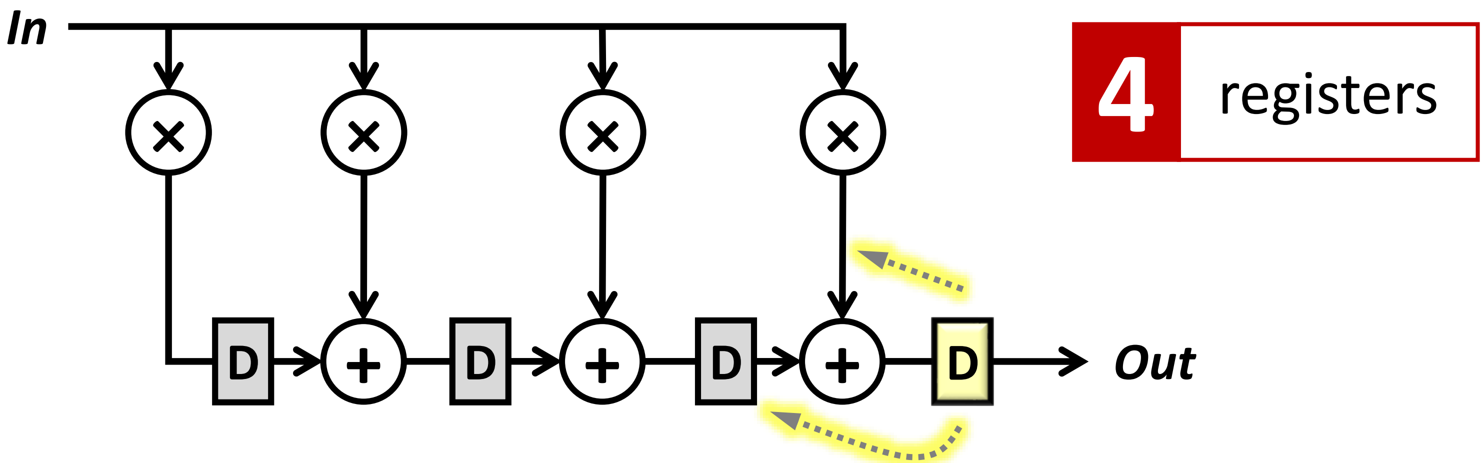
- **Optimal placement of existing data-path registers**
  - Register movement does not alter functionality
- **Objective:** balance $t_{critical}$ for maximum throughput



4 registers

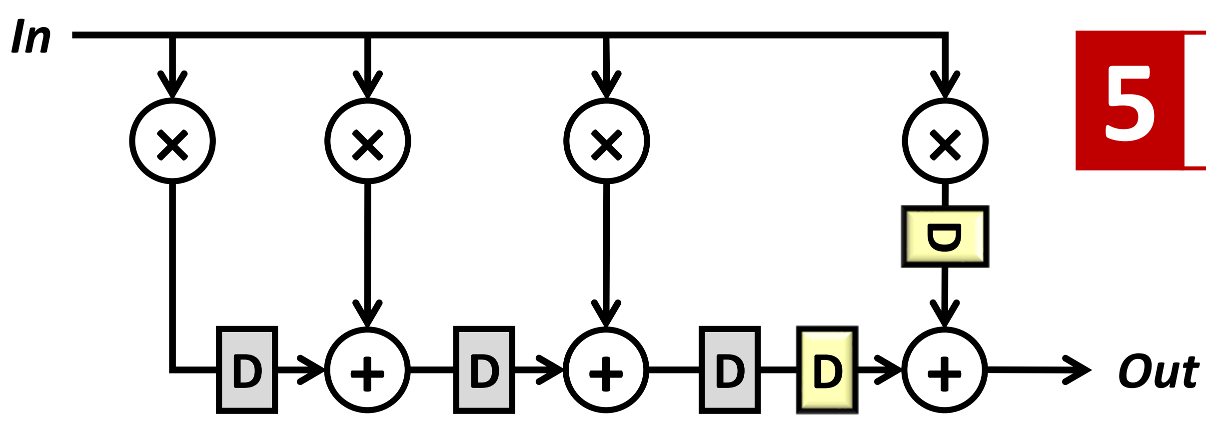
Example 9.2: Retiming of FIR Filter

- **Step 1:** move output register inside



Example 9.2: Retiming of FIR Filter

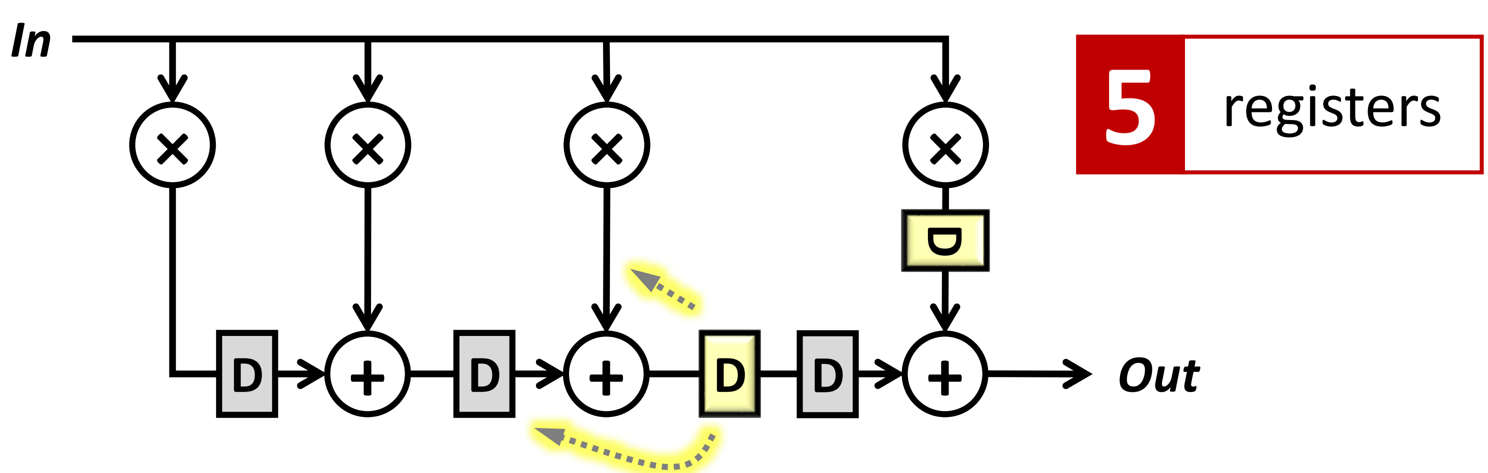
- **Output register splits into two input registers after the first step**
  - Total number of registers increases



5 registers

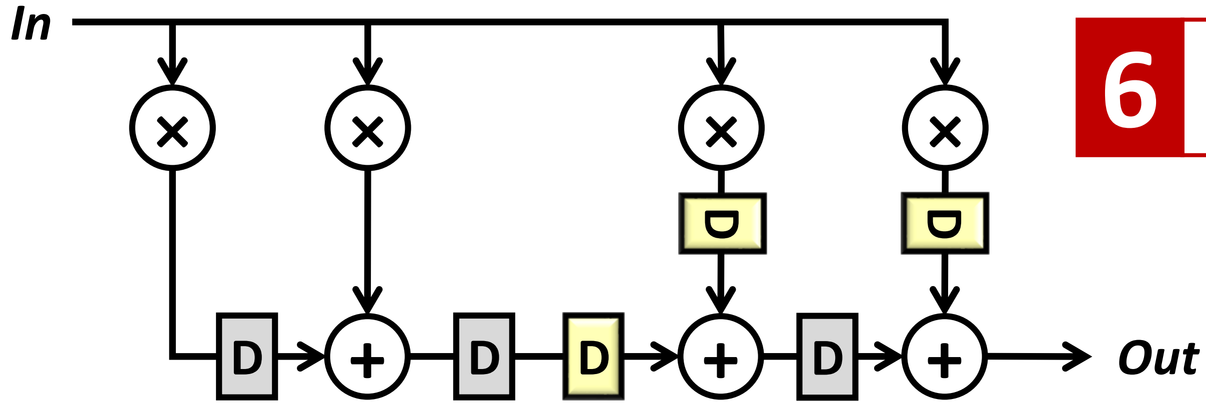
Example 9.2: Retiming of FIR Filter

- **Step 2:** move internal register further inside



Example 9.2: Retiming of FIR Filter

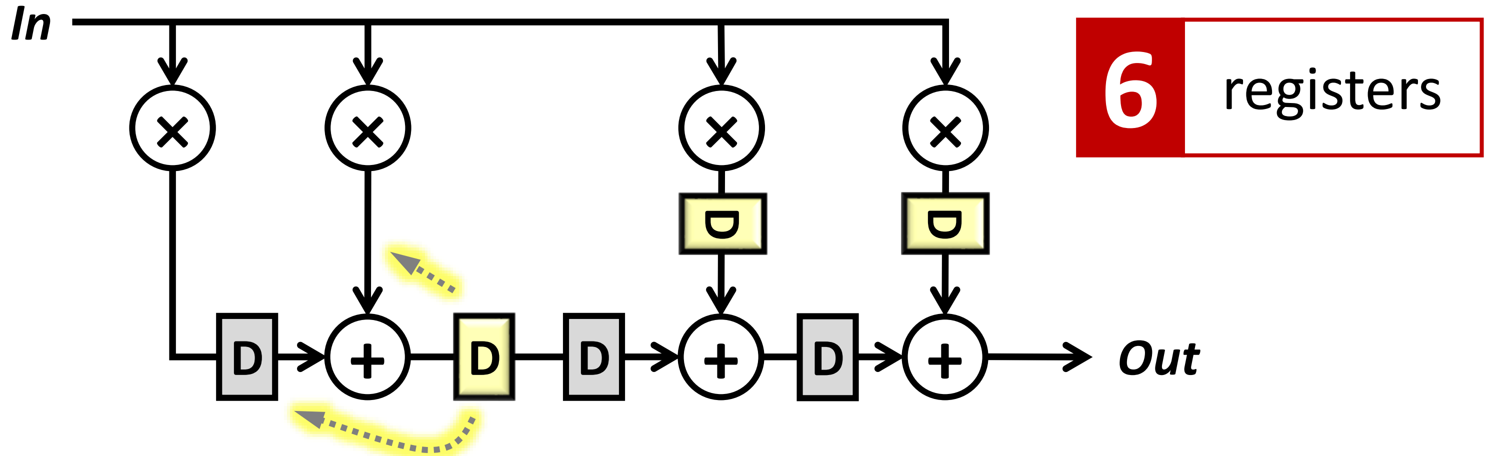
- **Internal register splits into two input registers after the second step**
  - Total number of registers increases



6 registers

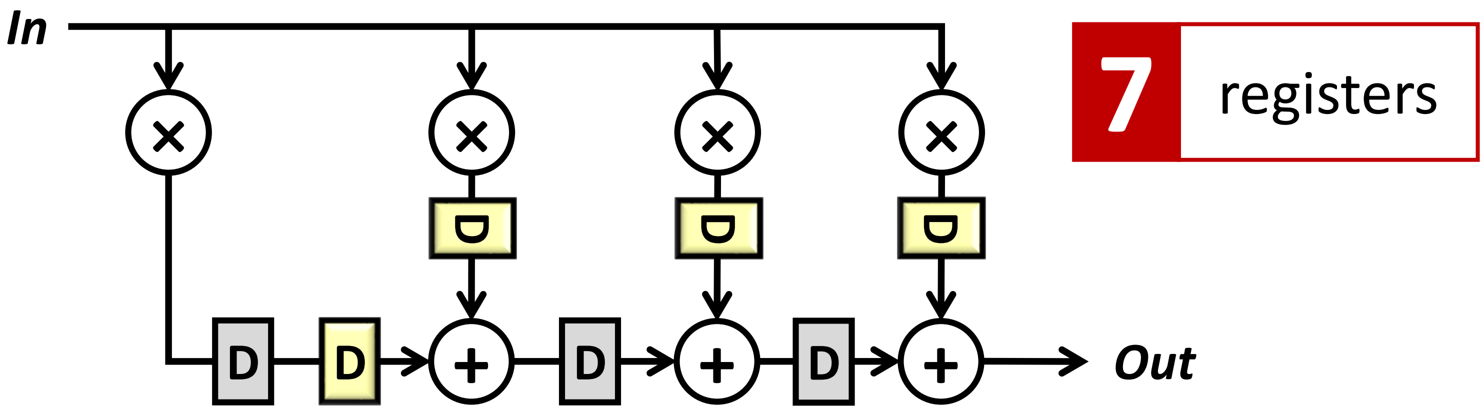
Example 9.2: Retiming of FIR Filter

- **Step 2+:** continue moving internal registers until you reach input



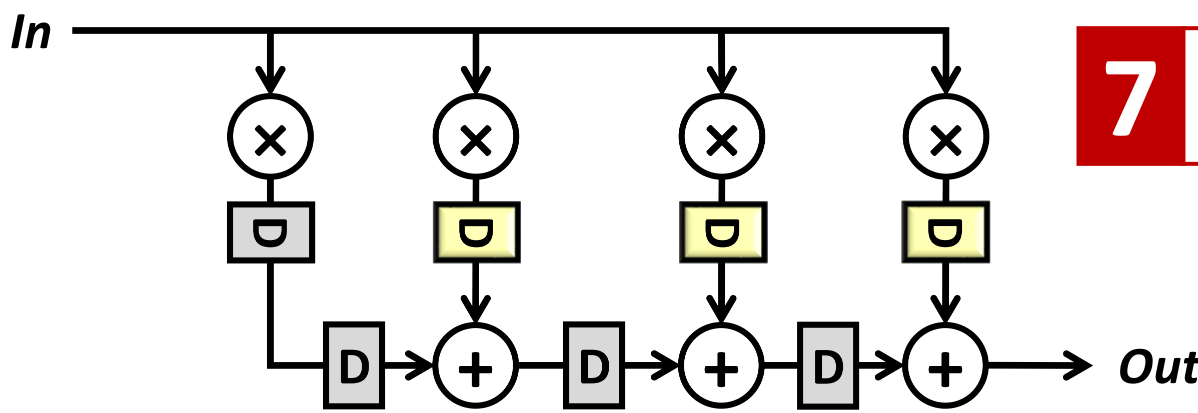
Example 9.2: Retiming of FIR Filter

- **Each out/in split added an extra register**
  - 4 filter stages: 3 additional registers



Example 9.2: Retiming of FIR Filter

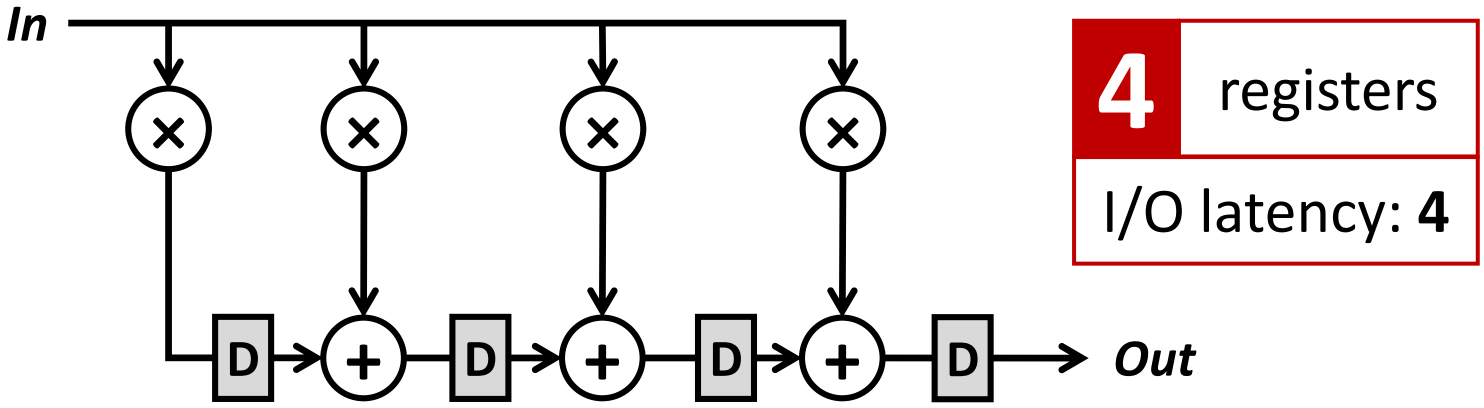
- **Each out/in split added an extra register**
  - 4 filter stages: 3 additional registers

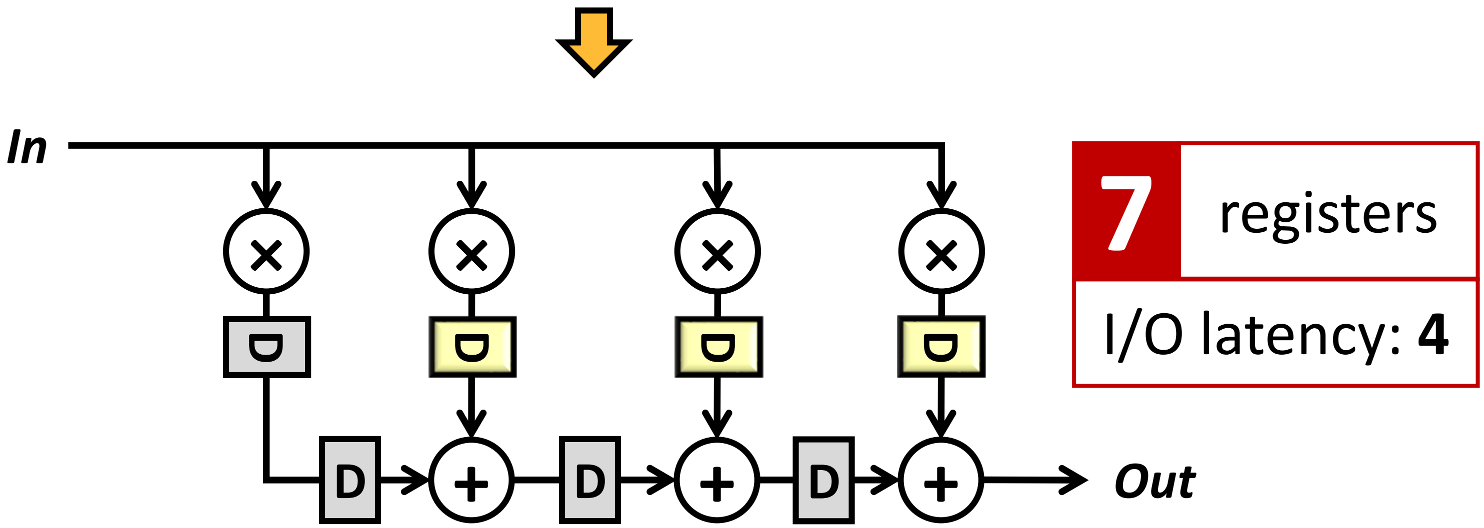


7 registers

High-Level Retiming

Number of registers increased, I/O latency unchanged



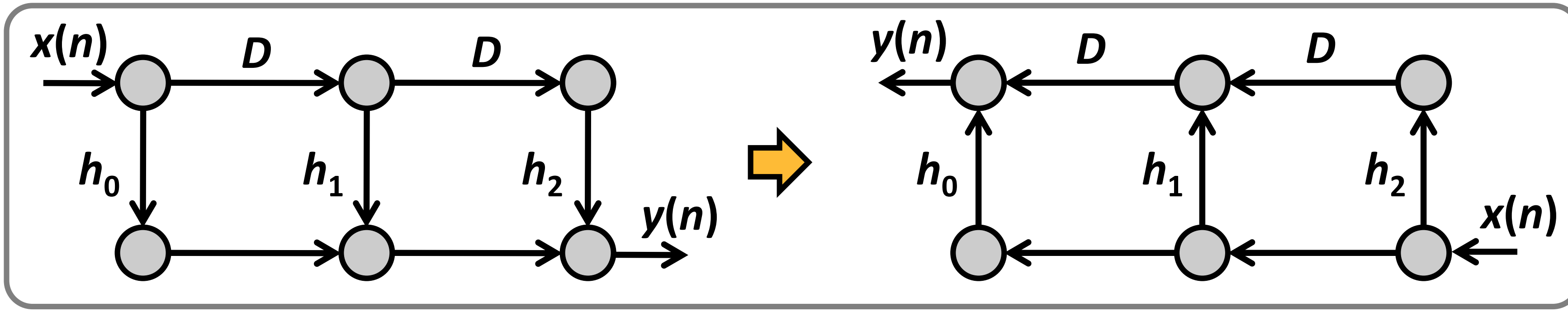


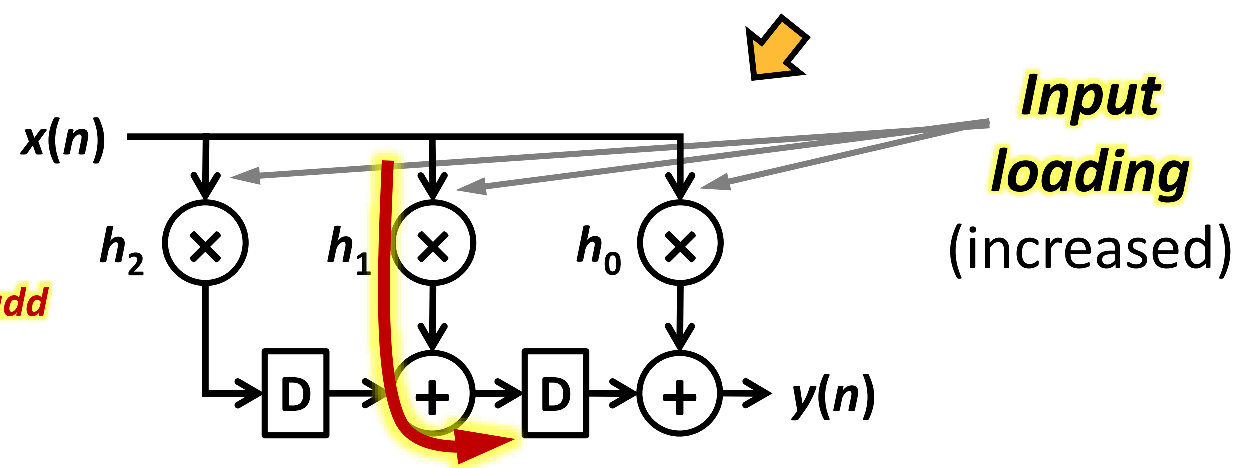
Pipelining vs. Retiming

- **Pipelining**
  - Inserts registers into cut-sets
  - I/O latency increases
- **Retiming**
  - Moves existing registers
  - I/O latency unchanged

Transposing FIR

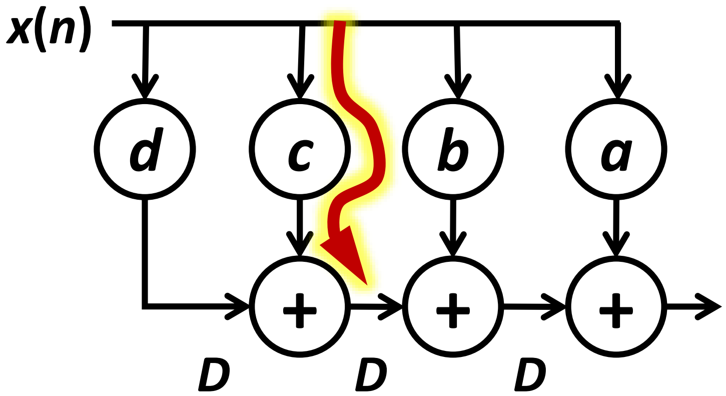
- Reverse the direction of edges in a signal-flow graph
- Interchange the input and output ports
- Functionality unchanged





$t_{critical} = t_{mult} + t_{add}$
(decreased)

Transposed + Parallel FIR

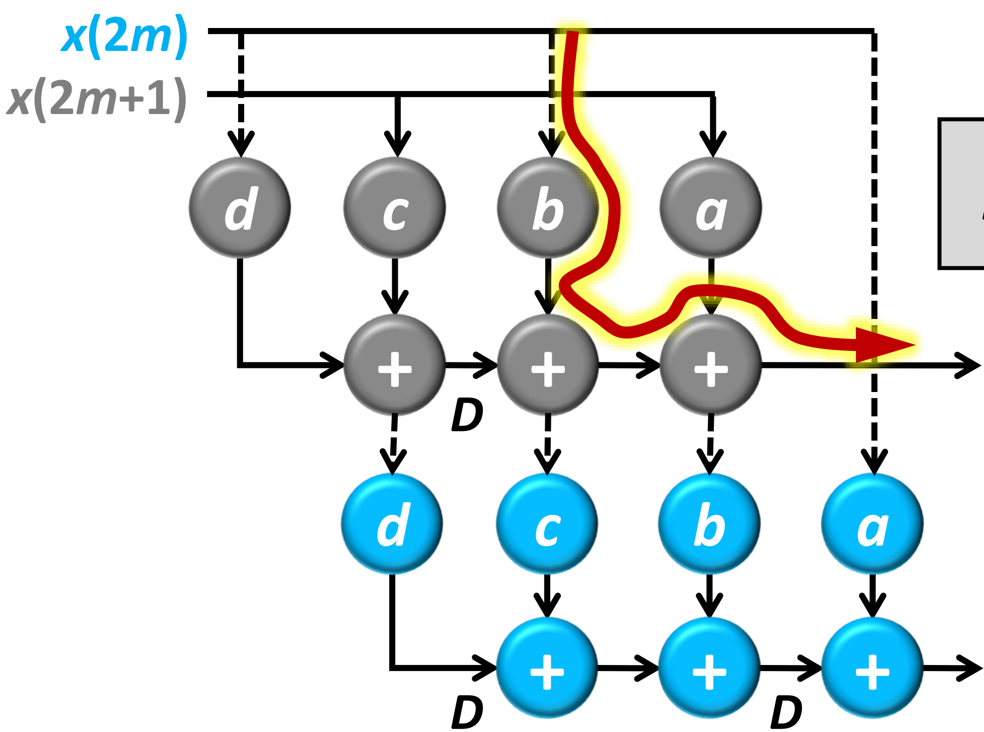


Critical path = $t_{add} + t_{mult}$

$y(n) = a \cdot x(n) + b \cdot x(n-1) + c \cdot x(n-2) + d \cdot x(n-3)$

Critical path = $2t_{add} + t_{mult}$

Per iteration: $t_{add} + t_{mult}$ / 2



$y(2m+1) = a \cdot x(2m+1) + b \cdot x(2m)$
$+ c \cdot x(2m-1) + d \cdot x(2m-2)$

$y(2m) = a \cdot x(2m) + b \cdot x(2m-1)$
$+ c \cdot x(2m-2) + d \cdot x(2m-3)$

FIR Summary

Main features:

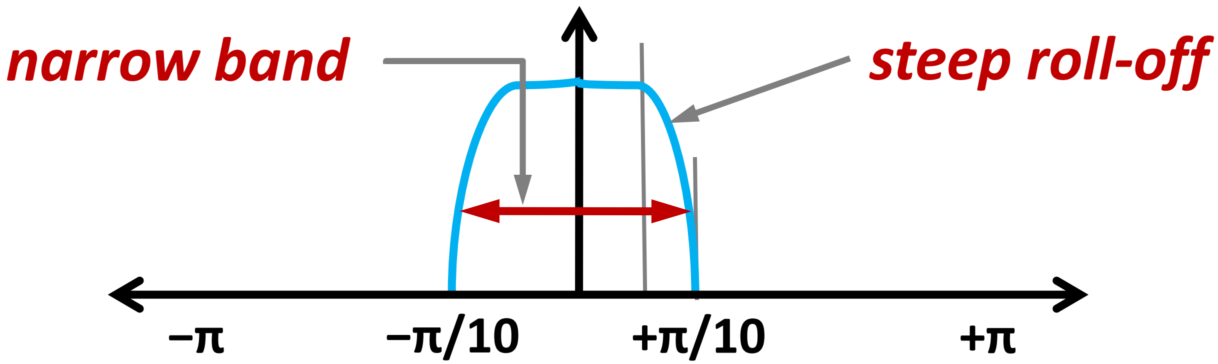
- Easy to design, always stable
- Feed-forward: can be pipelined, parallelized
- Linear phase response

Realize narrow band, steep roll-off?

- FIR filters require large number of taps
- Area and power cost can make FIR unsuitable

Recursive (IIR) Filters

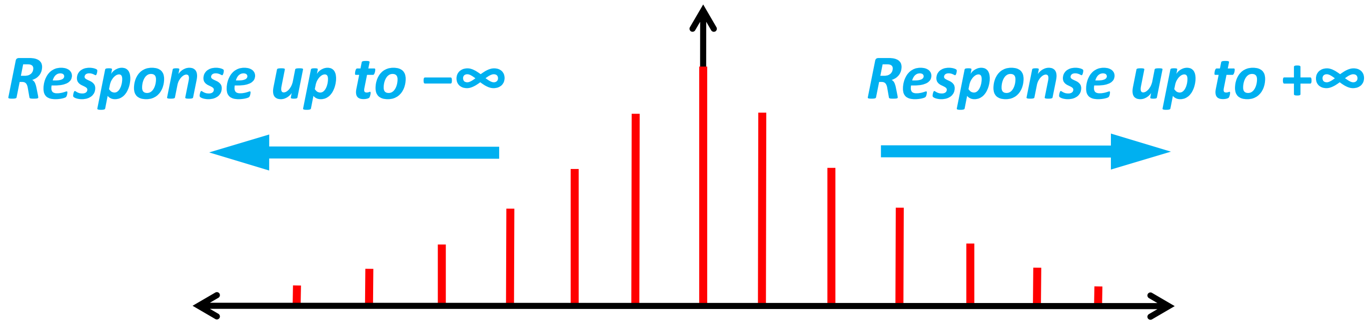
IIR Filters for Narrow Band, Steep Roll-Off



Infinite impulse response (IIR) filters are more suited to achieve such a frequency response with low area, power

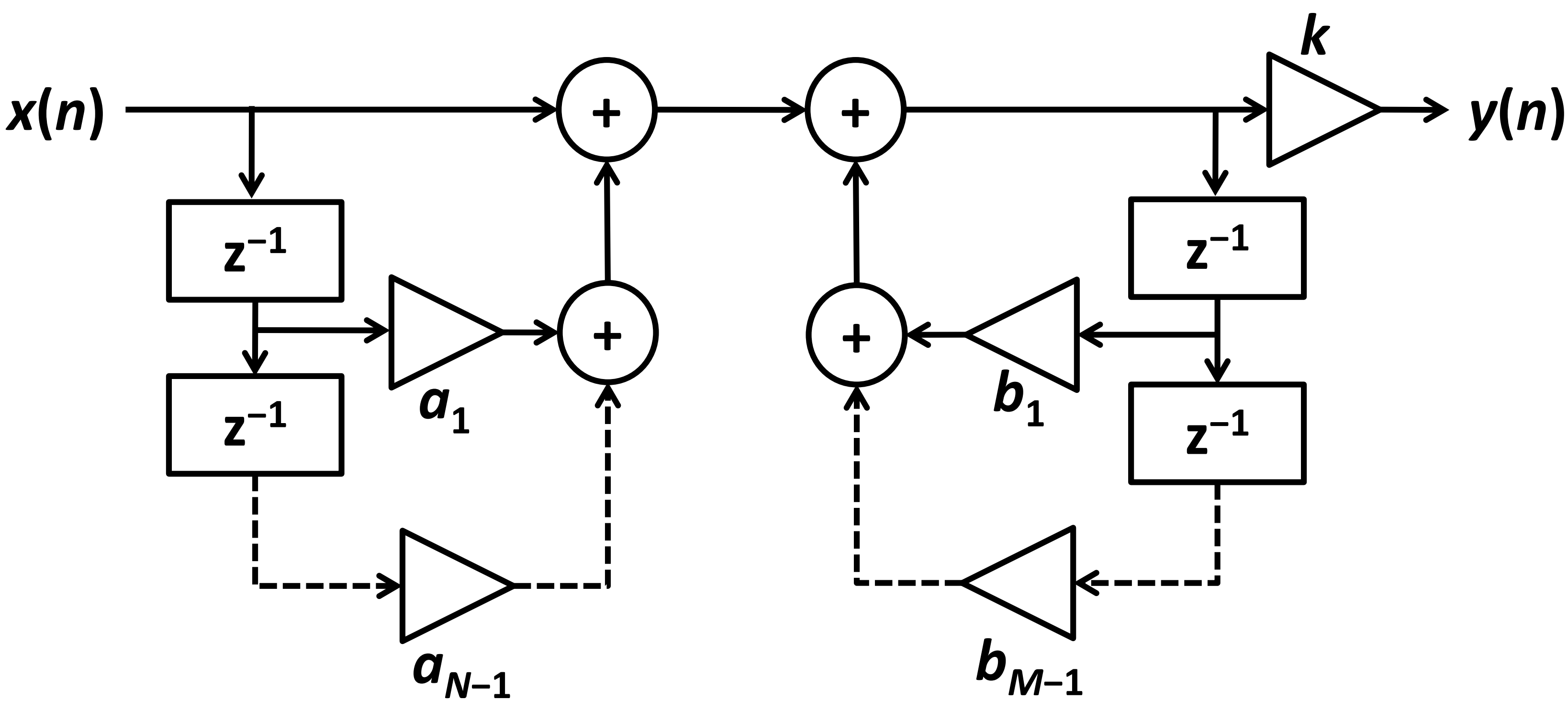
Generalized IIR Transfer Function

$$y(n) = \underbrace{\sum_{m=1}^{N} b_m y(n-m)}_{\textit{Feedback}} + \underbrace{\sum_{p=0}^{N} a_p x(n-p)}_{\textit{Feed-forward}}$$



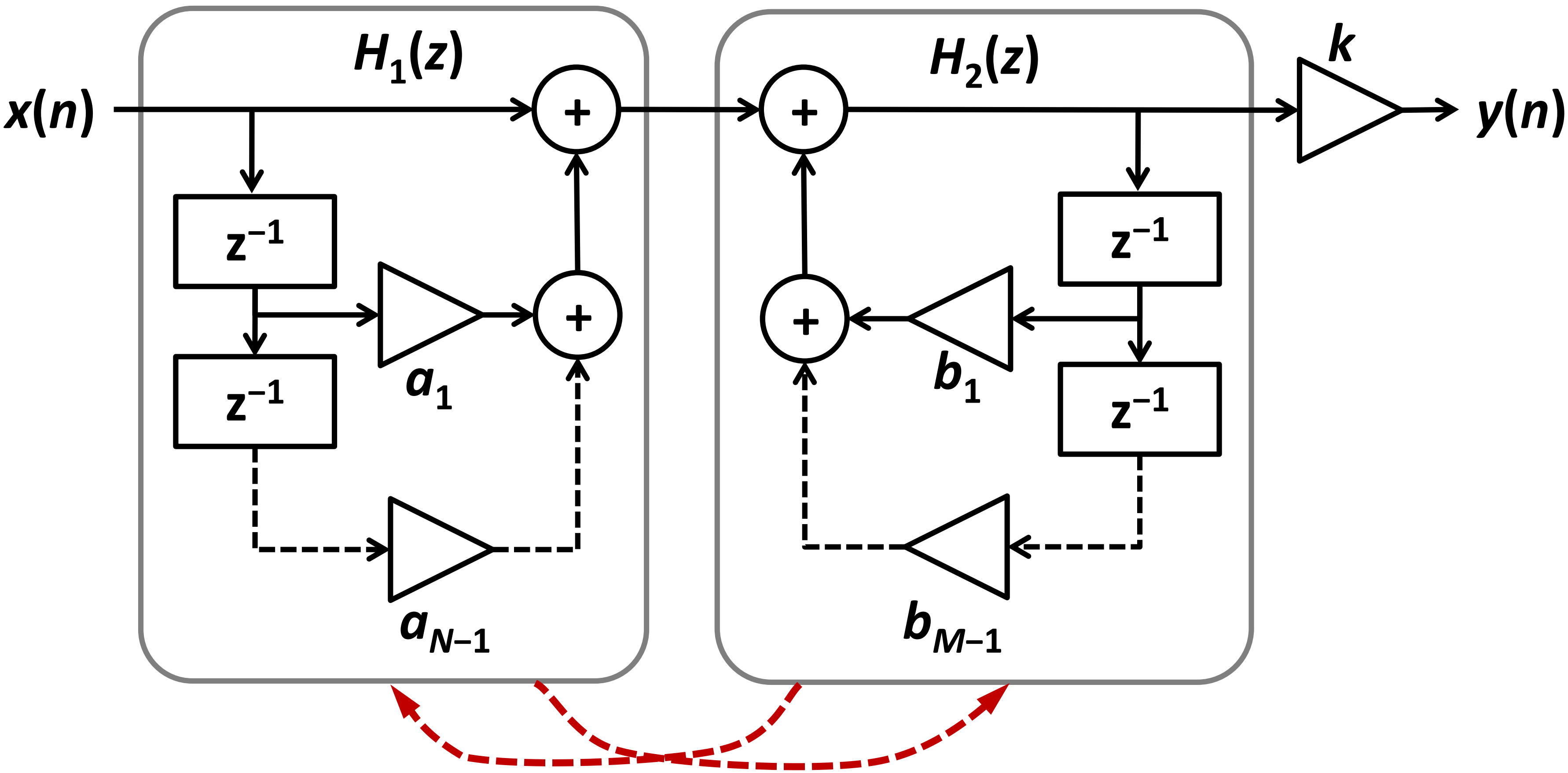
Direct-Form IIR Architecture

$$H(z) = k\frac{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{N-1} z^{-(N-1)}}{1 - b_1 z^{-1} - b_2 z^{-2} - \ldots - b_{M-1} z^{-(M-1)}}$$



IIR Architecture Optimization

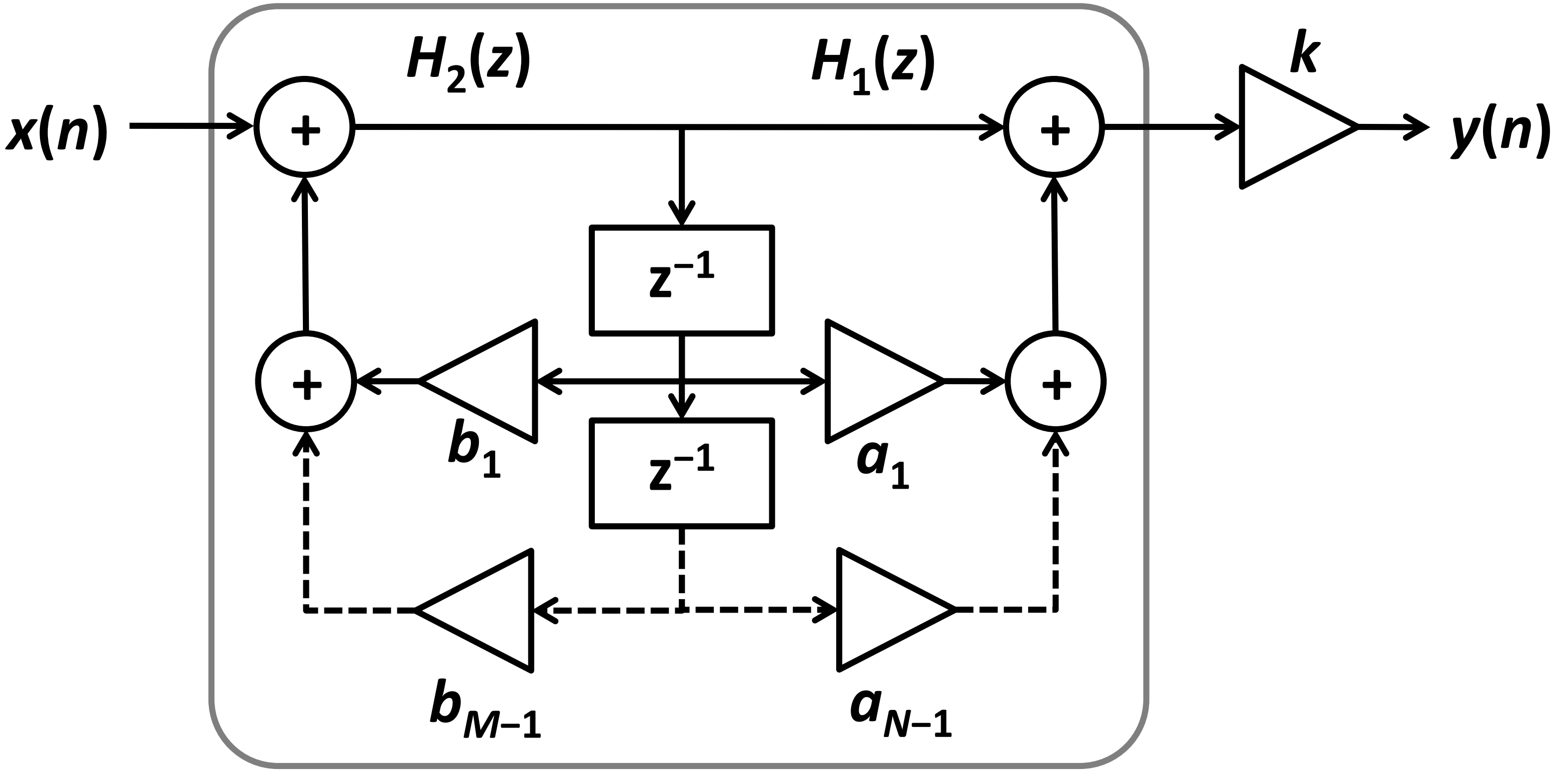
Functionality unchanged if H_1, H_2 are swapped



Swap the order of execution

IIR Architecture Optimized

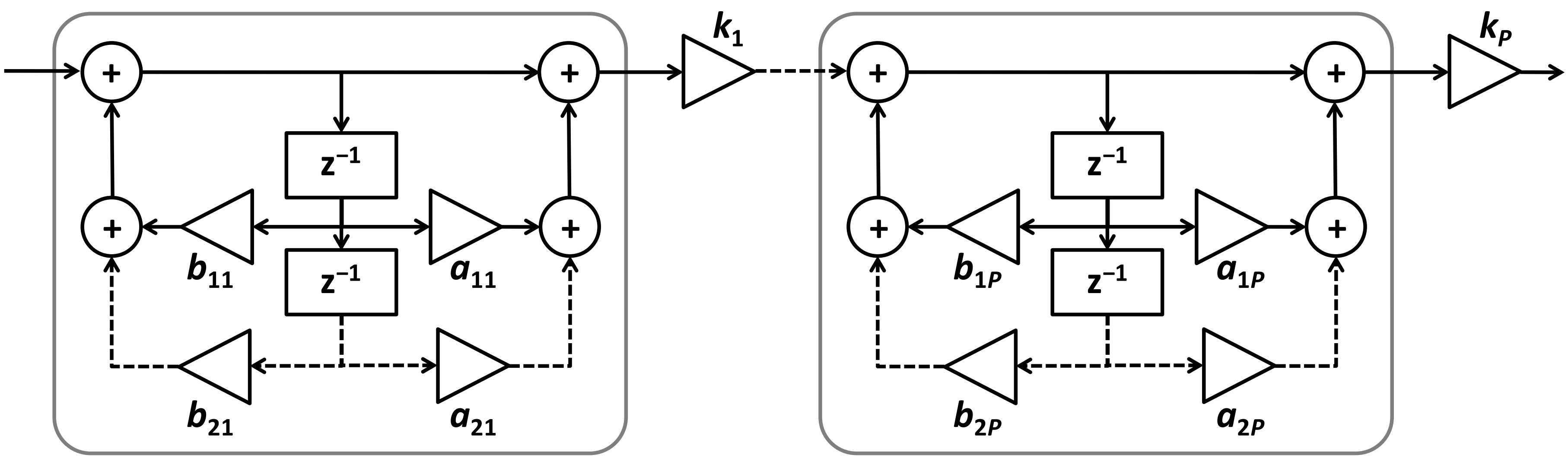
- **H_1, H_2 can share the central register bank**
  - If $M \neq N$, the central bank will have max(M,N) registers



Cascaded IIR

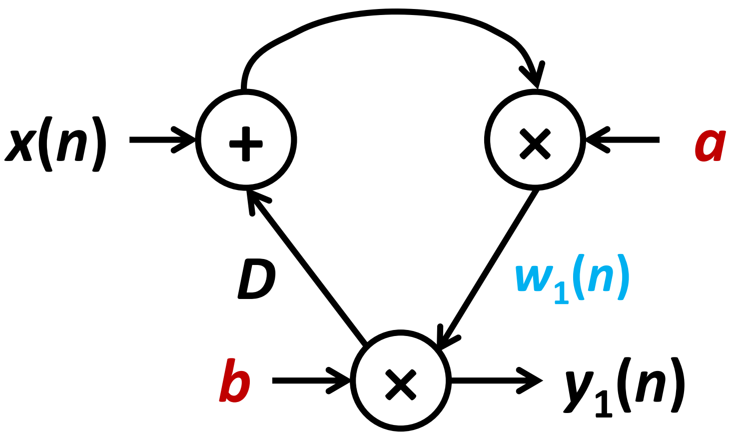
$$H(z) = k_1 \frac{1 + a_{11} z^{-1} + a_{21} z^{-2}}{1 - b_{11} z^{-1} - b_{21} z^{-2}} \ldots k_p \frac{1 + a_{1p} z^{-1} + a_{2p} z^{-2}}{1 - b_{1p} z^{-1} - b_{2p} z^{-2}}$$

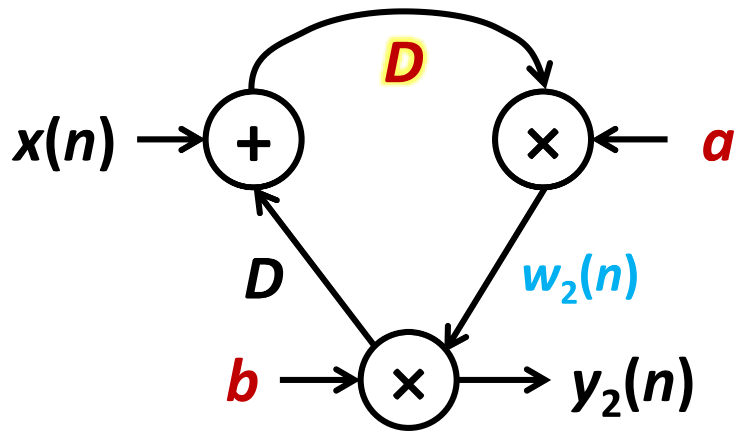
- **Implement IIR as cascade of 2nd order sections**
  - Shorter wordlengths in the feedback loop adders
  - More area and power efficient architecture



Recursive-Loop Bottlenecks

- **Pipelining loops not possible**
  - # registers in feedback loops must remain fixed





$$w_1(n) = a\cdot(y_1(n-1) + x(n))$$
$$y_1(n) = b\cdot a\cdot y_1(n-1) + b\cdot a\cdot x(n)$$

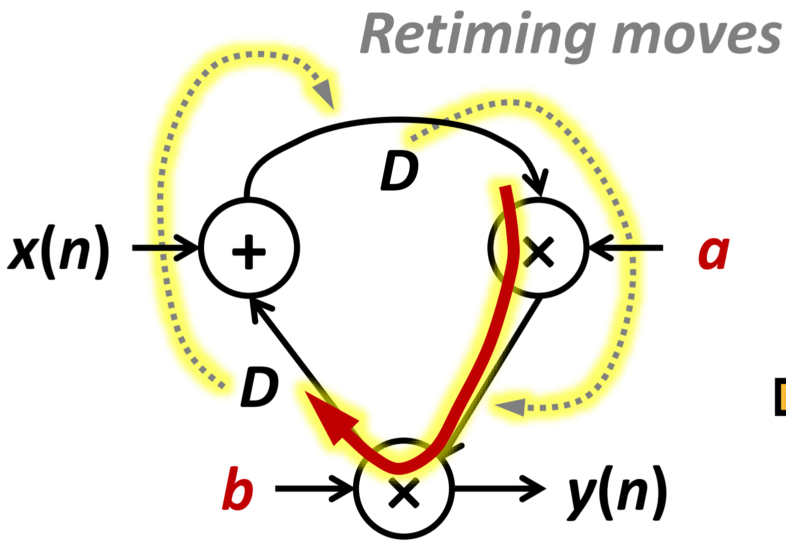
$$w(n) = a\cdot(y_2(n-2) + x(n-1))$$
$$y_2(n) = b\cdot a\cdot y_2(n-2) + b\cdot a\cdot x(n-1)$$

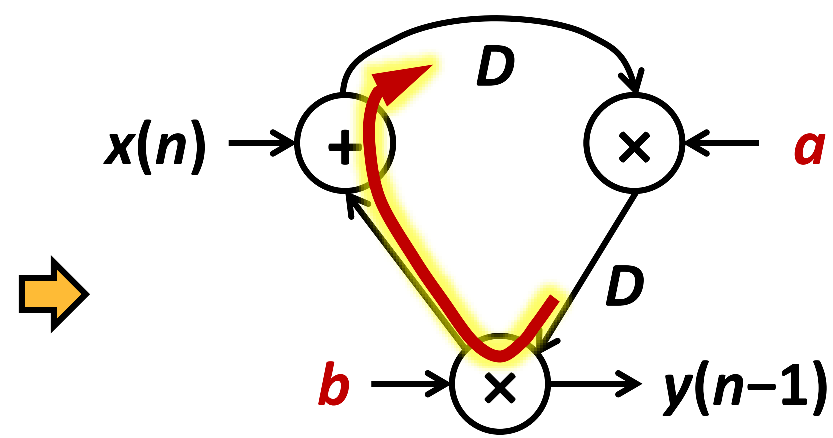
$$\boxed{y_1(n) \neq y_2(n)}$$

Changing the # delays in a loop alters functionality

High-Level Retiming of an IIR Filter

- IIR throughput is limited by the retiming in the feedback sections
- Optimal placement of registers in the loops leads to max speed



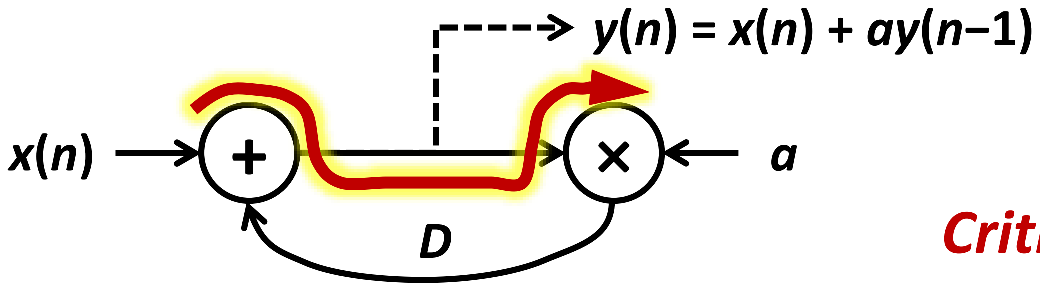


$t_{critical} = 2t_{mult}$

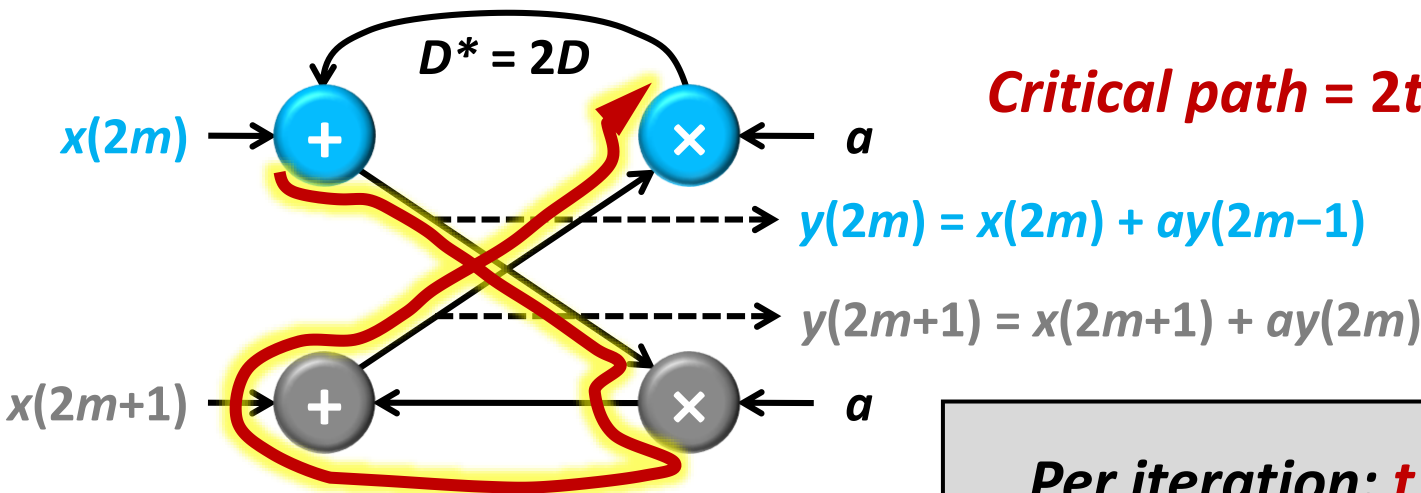
$t_{critical} = t_{add} + t_{mult}$

Unfolding: Constant Throughput

- Maximum throughput limited by iteration bound (IB)
- Unfolding does not help if IB is already achieved



Critical path = $t_{add} + t_{mult}$



Critical path = $2t_{add} + 2t_{mult}$

Per iteration: $t_{add} + t_{mult}$

IIR Summary

- **Pros**
  - Suitable when filter response has sharp roll-off, has narrow-band, or large attenuation in the stop-band
  - More area- and power-efficient compared to FIR
- **Cons**
  - Difficult to ensure filter stability
  - Sensitive to finite-precision arithmetic effects (limit cycles)
  - Does not have linear phase response unlike FIR filters
  - All-pass filters required if linear phase response desired
  - Difficult to increase throughput
    - Pipelining not possible
    - Retiming and parallelism has limited benefits

Multi-Rate Filters

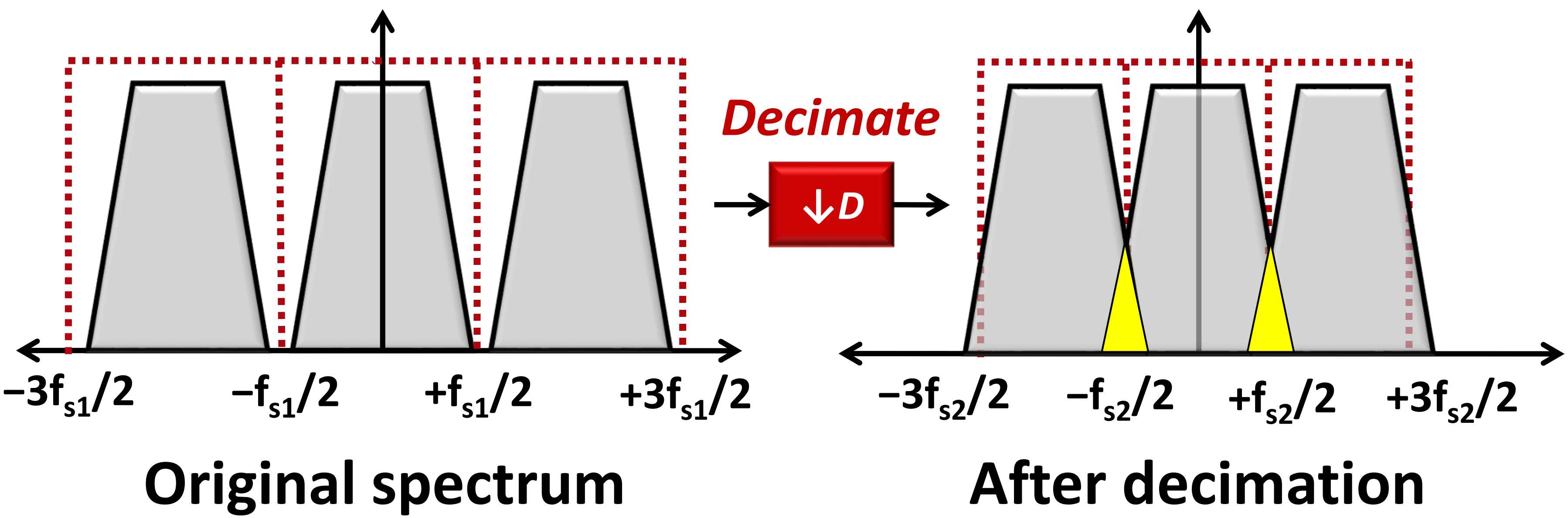
Multi-Rate Filters

- **Data transfer between blocks at different sample rate**
  - **Decimation:** higher rate f_{s1} to lower rate f_{s2}
  - **Interpolation:** lower rate f_{s2} to higher rate f_{s1}

- **For integer f_{s1}/f_{s2}**
  - Drop samples when decimating
    - Leads to aliasing in the original spectrum
  - Stuff zeros when interpolating
    - Leads to images at multiples of f_{s2}

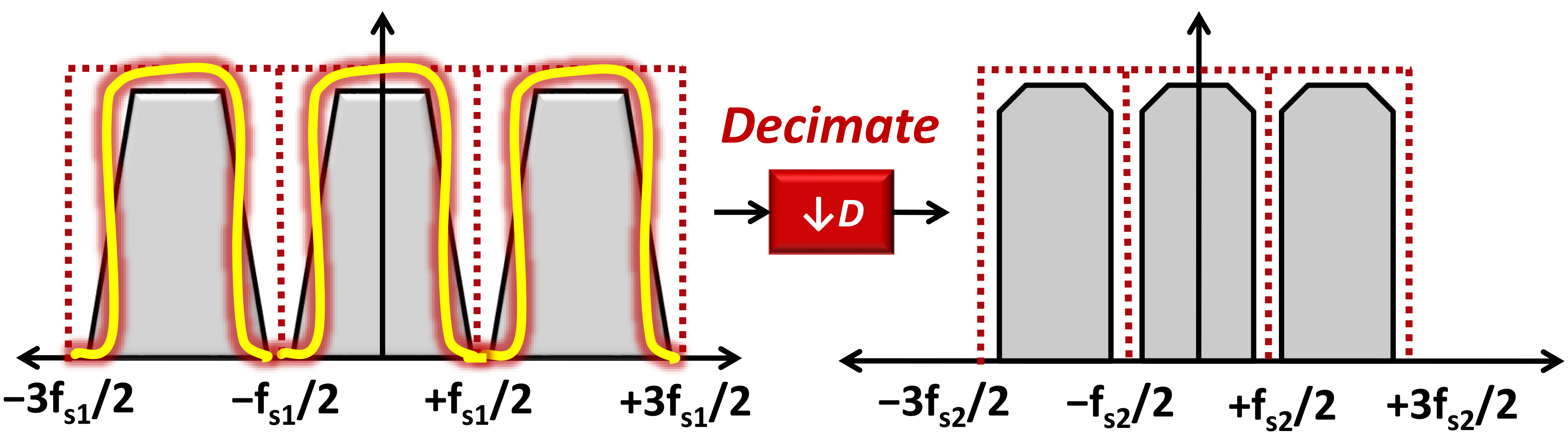
Decimation

- **Frequency-domain representation**
  - Spectrum replicated at intervals of f_{s1} originally
  - Spectrum replicated at intervals of f_{s2} after decimation
  - Aliasing of spectrum lying beyond B/W f_{s2} in original spectrum
- **Decimation filters to remove content beyond f_{s2}**



Decimation Filters

- **Low-pass filters used before decimation**
  - Usually FIR realization (IIR if linear phase is not necessary)
  - Cascade integrated comb filter for hardware efficient realization
    - Much cheaper to implement than FIR or IIR realizations
    - Less attenuation, useful in heavily over-sampled systems

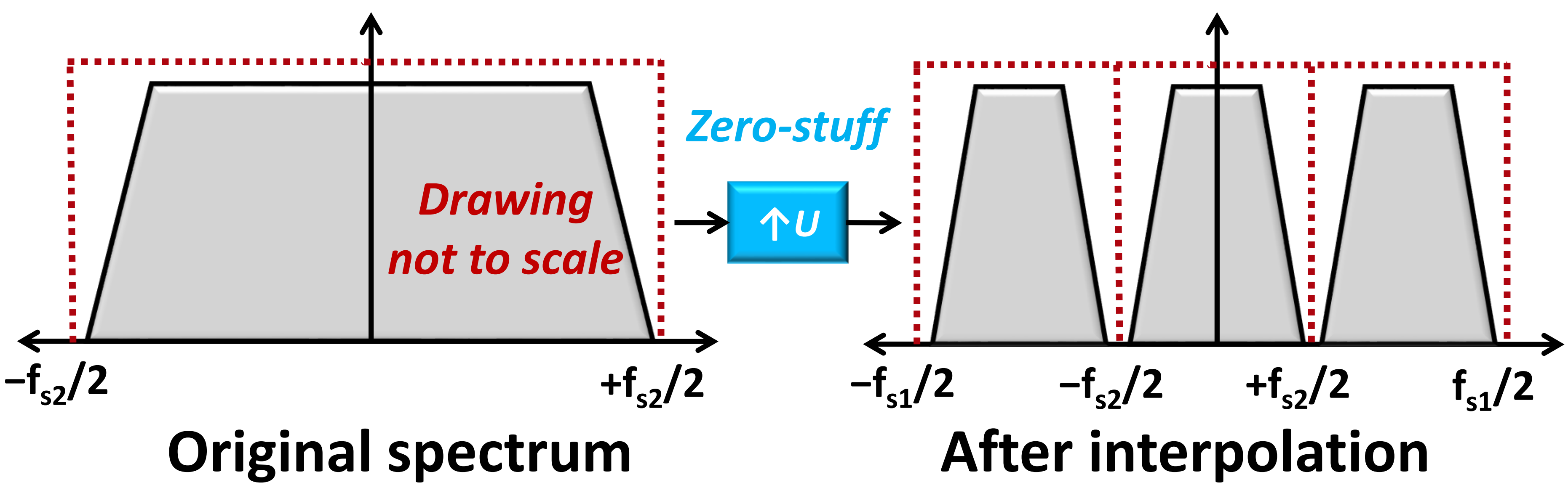


Original spectrum

After decimation

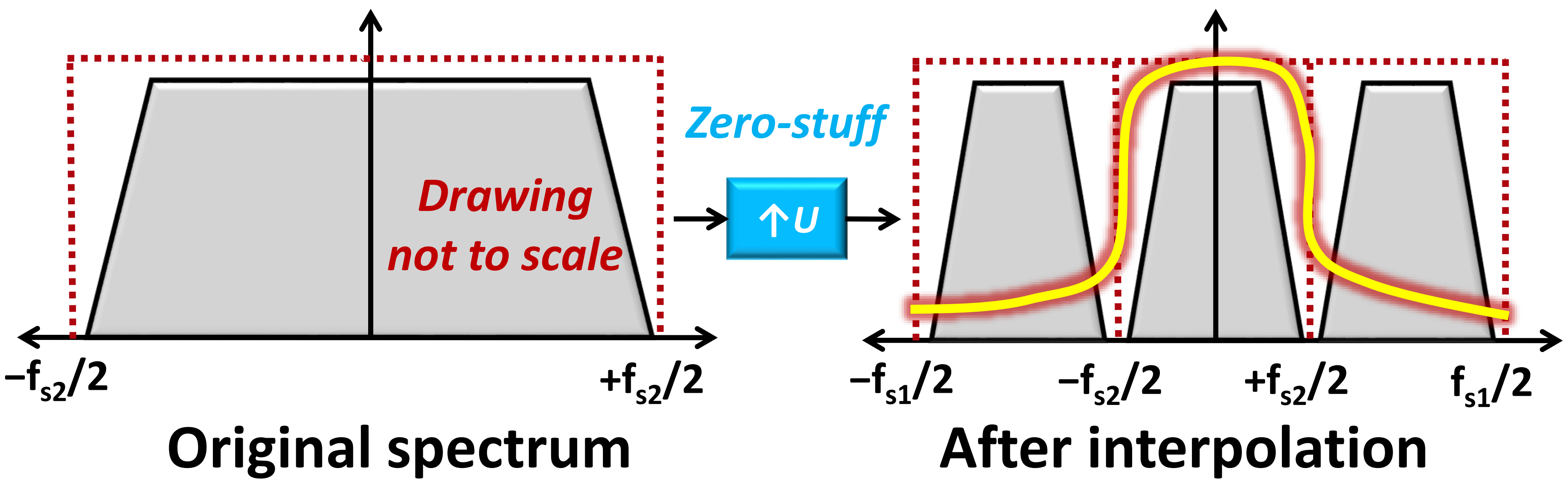
Interpolation

- **Frequency domain representation**
  - Spectrum spans bandwidth of f_{s2} originally
  - Spectrum spans bandwidth of f_{s1} after interpolation
  - Images of spectrum at intervals of f_{s2} after interpolation
- **Interpolation filters to remove images at multiples of f_{s2}**



Interpolation Filters

- **Low-pass filter used after interpolation**
  - Suppress spectrum images at multiples of f_{s2}
  - Usually FIR realizations (IIR if linear phase not necessary)
  - Cascade integrated comb filter for hardware efficient realization
    - Much cheaper to implement than FIR or IIR realizations
    - Less attenuation, useful for large interpolation factors (U)

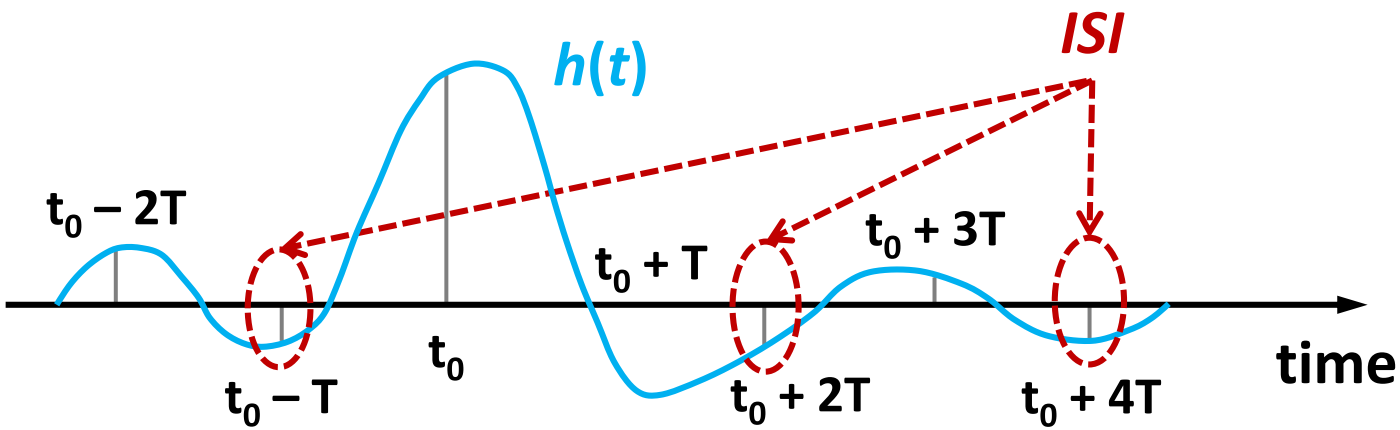


Adaptive Filters:

Equalization

Inter-Symbol Interference (ISI)

- **Channel response causes delay spread in Tx symbols**
- **Adjacent symbols contribute at sampling instants**

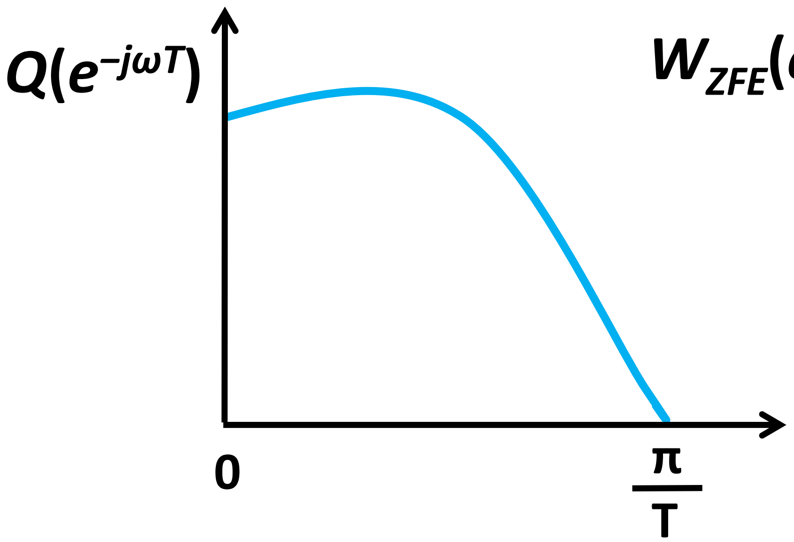


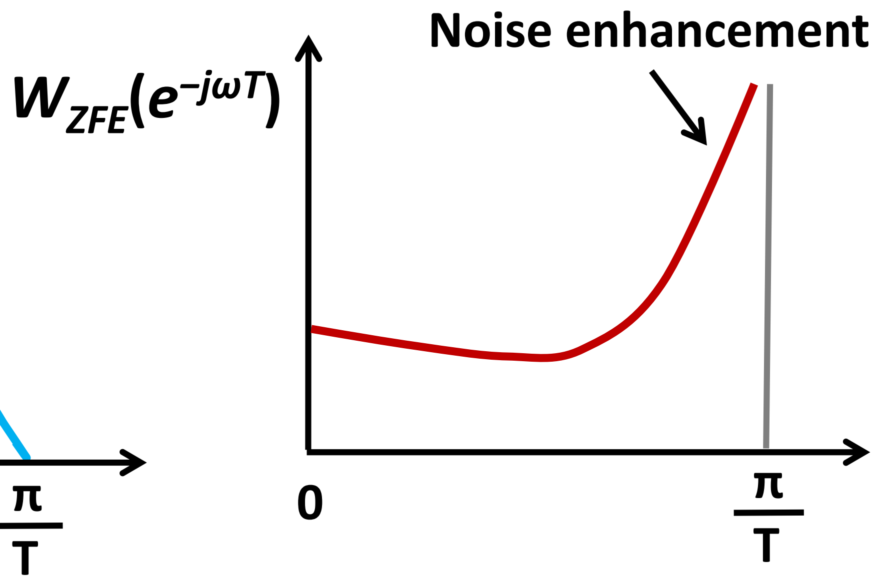
- **ISI and adaptive noise modeling**

$$r(t_0 + kT) = x_k h(t_0) + \underbrace{\sum_{j \neq k} x_j h(t_0 + kT - jT)}_{ISI} + \underbrace{n(t_0 + kT)}_{noise}$$

Zero-Forcing Equalizer

- **Basic equalization techniques**
  - Zero-forcing (ZF)
    - Causes noise enhancement





- **Adaptive equalization**
  - Least-mean squares (LMS) algorithm
    - More robust

Adaptive Equalization

- **Achieve minimum mean-square error (MMSE)**
  - Equalized signal: z_k, transmitted signal: x_k
  - Error: $e_k = z_k - x_k$
  - **Objective:** Minimize expected error, $\min E[e_k^2]$
  - Equalizer coefficients at time instant k: $c_n(k)$, $n \in \{0,1,\ldots,N\}$
  - For real optimality: set

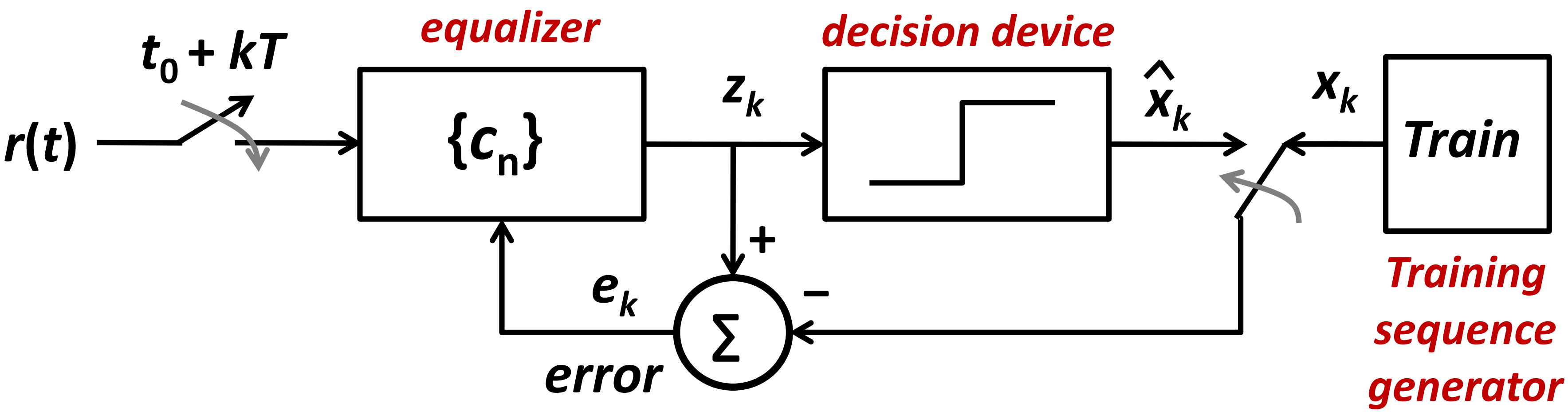
$$\frac{\partial E[e_k^2]}{\partial c_n(k)} = 0$$

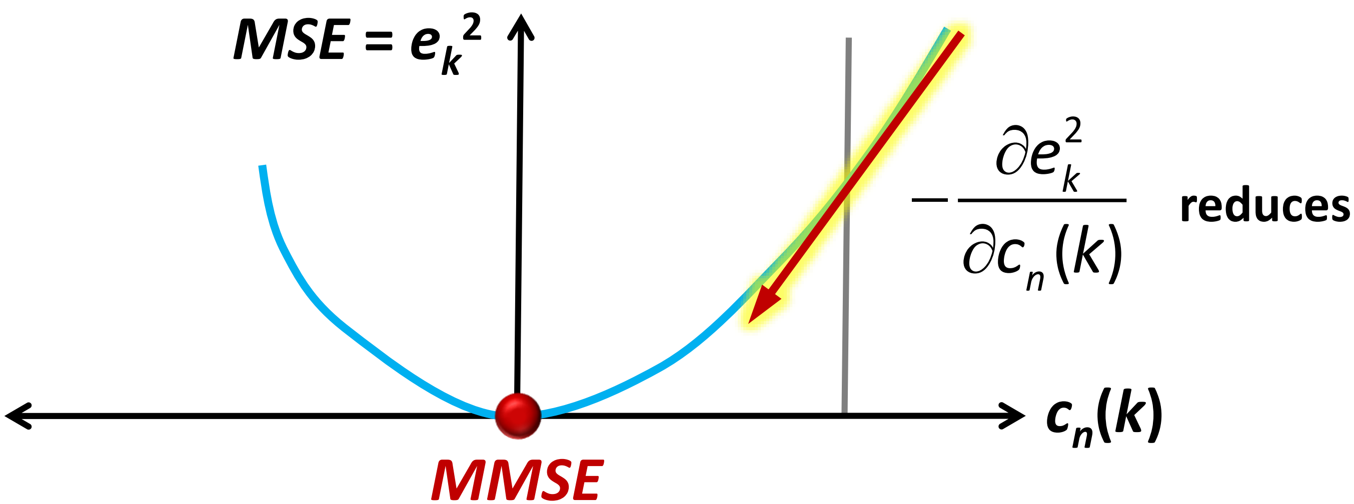
- **Equalized signal z_k given by:**

$$z_k = c_0 r_k + c_1 r_{k-1} + c_2 r_{k-2} + \ldots + c_{n-1} r_{k-n+1}$$

- **z_k is convolution of received signal r_k w/ tap coefficients**

LMS Adaptive Equalization





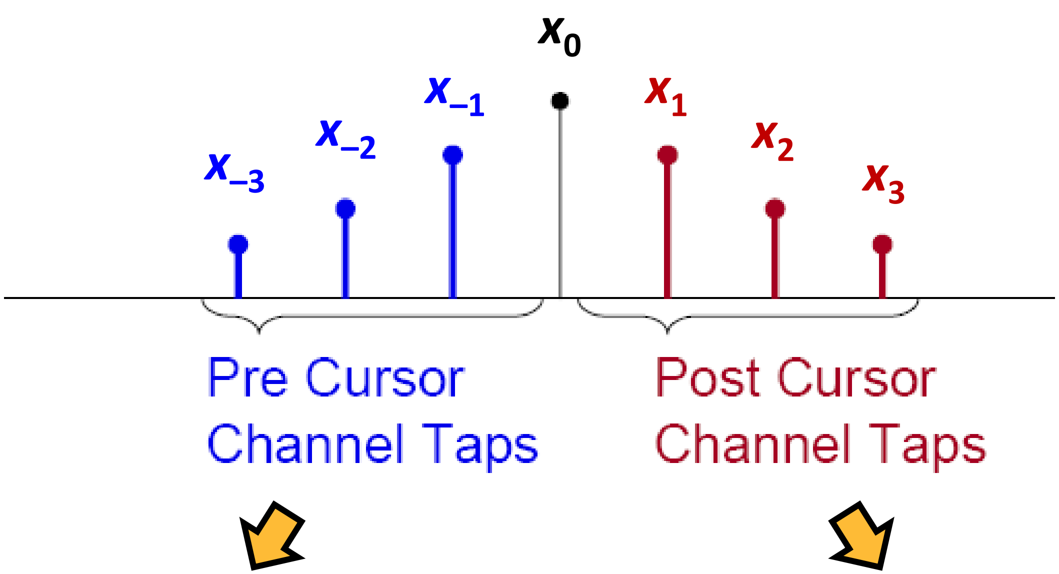
Approximate Calculation of MMSE

- **For computational optimality**
  - Set: $\boldsymbol{\partial E[e_k^2]/\partial c_n(k) = 2e_k r_{k-n} = 0}$
- **Tap update equation: $\boldsymbol{e_k = z_k - x_k}$**
- **Step size: Δ**
- **Good approximation if using:**

  $\boldsymbol{c_n(k+1) = c_n(k) - \Delta e_k r(k-n)}$, $n = 0, 1, \ldots, N-1$
  - Small step size
  - Large number of iterations
- **Error-signal power comparison: $\boldsymbol{\sigma_{LMS}^2 \le \sigma_{ZF}^2}$**

Decision-Feedback Equalizers

Cancel the interference from previously detected symbols



Feed-forward equalizer

- Remove the pre-crs ISI
- FIR (linear)

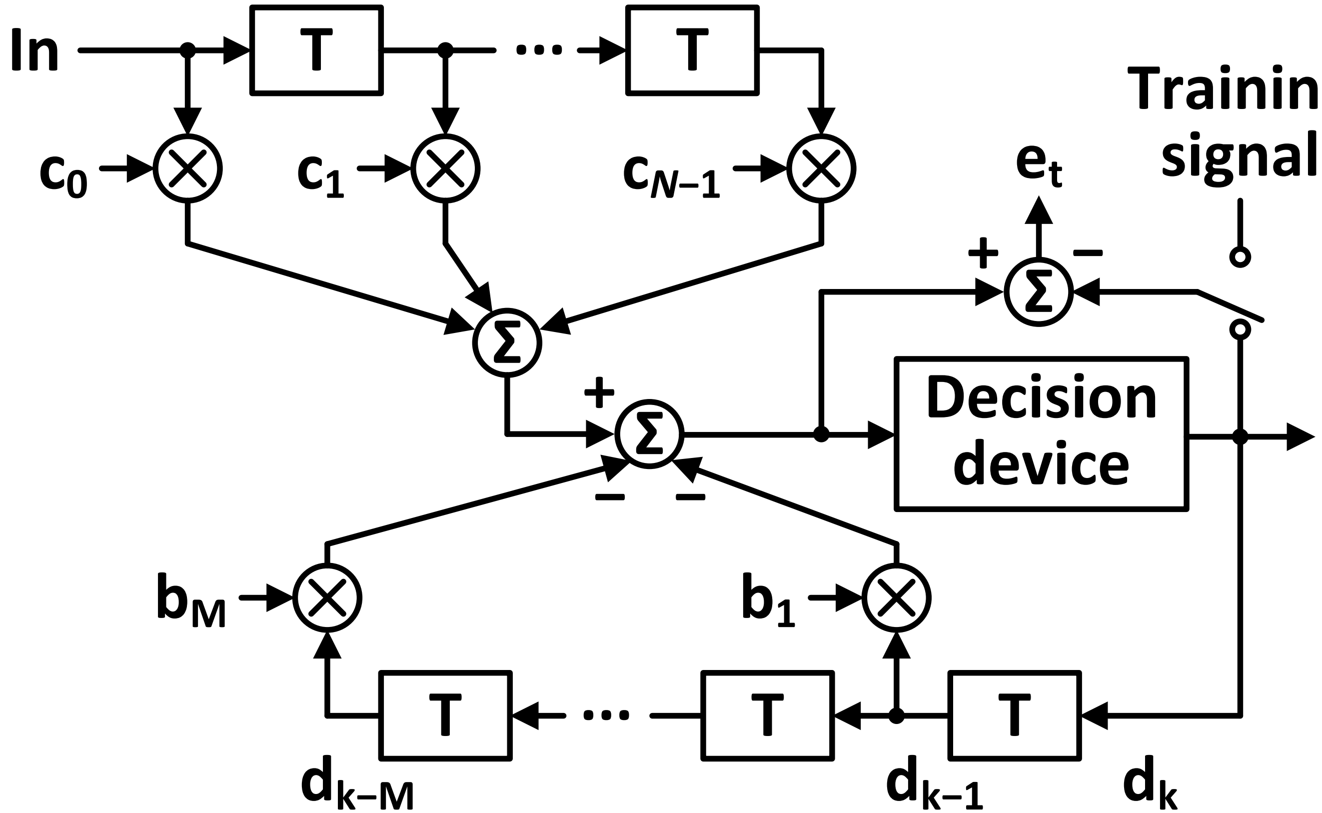
Feedback equalizer

- Remove the post-crs ISI
- Like a ZF equalizer

$$b_{m+1}(k+1) = b_m(k) - \Delta e_k d_{k-m}$$

DFE Architecture

Feed-forward equalizer: $c_0, c_1, \ldots, c_{N-1}$



Feedback equalizer: $b_1, b_2, \ldots, b_M$

DFE Properties

- **Less noise enhancement compared with ZF or LMS**
- **More freedom in selecting feed-forward coefficients**
  - Feed-forward equalizer need not fully invert channel response
- **Symbol decision may be incorrect**
  - Error propagation (slight)

Fractionally-Spaced Equalizers

- **Sampling at symbol period**
  - Equalizes the aliased response
  - Sensitive to sampling phase
- **Can over-sample (e.g. 2x higher rate)**
  - Avoid spectral aliasing at the equalizer input
  - Sample the Rx input at higher rate (e.g. 2x faster)
  - Produce equalizer output signal at symbol rate
  - Can update coefficients at symbol rate
  - Less sensitive to sampling phase

$$c_n(k+1) = c_n(k) - \Delta e_k \cdot r(t_0 + kT - NT/2)$$

Equalizers Summary

- **Use equalizers to reduce ISI and achieve high data rate**
- **Use adaptive equalizers to track time-varying channel**
- **LMS-based equalization prevails in MODEM design**
- **DFE uses previous decisions to estimate current symbol**
- **Fractionally spaced equalizers resilient to sampling phase variation**
- **Properly select step size for convergence**

S. Qureshi, "Adaptive Equalization," *Proceedings of the IEEE,* vol. 73, no. 9, pp. 1349-1387, Sep. 1985.

Digital Filters Summary

- **Digital filters are key building elements in DSP systems**
- **FIR filters can be realized in direct or transposed form**
  - Direct form has long critical-path delay
  - Transposed form has large input loading
  - Multiplications can be simplified by using coefficients that can be derived as sum of power-of-two numbers
- **Performance of IIR filters is limited by the longest loop delay (iteration bound)**
  - IIR filters are suitable for sharp roll-off characteristics
  - More power and area efficient than FIR
- **Multi-rate filters for decimation and interpolation**