DSP Design – Lecture 1

Introduction and DSP Basics

Steffen Malkowsky, PhD

steffen.malkowsky@eit.lth.se

Lecturers

Fredrik Edman
Mail: fredrik.edman@eit.lth.se
Room E:2538

Steffen Malkowsky
Mail: steffen.malkowsky@eit.lth.se
Room E:2334

Sidra Muneer (exercises & labs)
Mail: sidra.muneer@eit.lth.se
Room E:2339

…and several invited speakers!

Course Administrator

Erik Göthe
anne.andersson@eit.lth.se
Room E:3152b (3rd floor in the north-west part of the building)

Course information

- www.eit.lth.se/course/etin45

- Lectures
  - Please see detailed schedule

    https://cloud.timeedit.net/lu/web/lth1/ri1X50gQ0560YfQQ25Z5974Y0Zy7007315Y61Q569.html
- Seminars
  - Please the detailed scheduled
  - No seminar 1st week
- Labs

- Lab 1 Wednesday 5th Feb between 8 - 12
- Lab 2 Wednesday 12th Feb between 8 - 12
- Lab 3 Wednesday 19th Feb between 8 - 12
- Lab 4 Wednesday 4th March between 8 - 12

Course information

- Register for course
  - Done by signing up for labs
  - Do this before Friday, 24th Jan. 12.00
  - Registration is needed for
    - Getting access to labs
    - Getting access to software used in lab
    - If you miss this deadline, you may not be able to participate in first lab!

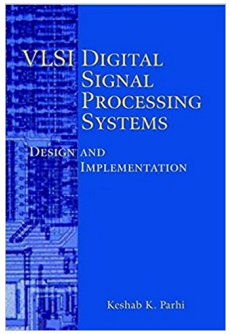
Compulsary Parts

- Pass 4 Laborations (MATLAB & Hardware design in CatapultC)
- Pass Homework exercises & Homework seminar

  ⇨ results in grade 3

- Written exam for grade 4 & 5

Litterature



- **Course Litterature**
  - **Keshab K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation***

- **Extended Reading**
  - **Alan V. Oppenheim, Ronald W. Schafer with John R. Buck, Discrete-Time Signal Processing, Prentice Hall, 1999, ISBN 0-13-754920-2.**
  - **John G. Proakis and Dimitris Manolakis, Digital Signal Processing: Principles, Algorithms and Applications, Prentice Hall, 1995, ISBN 0133737624.**
  - **Sanjit K. Mitra, Digital Signal Processing. A Computer Based Approach, McGRAW-HILL, 2001 ISBN: 0-07-118175-X**
  - **Lars Wanhammar, DSP Integrated Circuits, Academic Press, 1999, ISBN 0-12-734530-2**
  - **etc.**

Scope of the Course

How to get from a signal processing algorithm to an EFFICIENT implementation using a number of ”tools” such as;

- **Different numbering systems**
- **Pipelining**
- **Parallelism**
- **Unfolding/Folding**
- **Strength reduction, i.e. complexity of operations.**
- **etc, etc,...**

in a structured way according to the specification!

Goals

Aims: Knowledge

After completing the course the student should:

- have gained an understanding for the relationship between parameters such as calculation capacity, power consumption and silicon area
- be familiar with transformations that help the designer to develop different solutions for a given signal processing algorithm.
- understand how different number representations affect the solution.

Aims: Skills

After completing the course the student should:

- be able to suggest an architecture from a given set of criteria.
- be able to analyze an architecture and suggest alternative solutions.

Aims: Attitude

After completing the course the student should:

- have gained an overview of the field of implementation aspects of signal processing algorithms.
- feel well equipped to design an application specific processor given a specification using the methodologies covered in the course.

Introduction to DSP

Definition DSP (Digital Signal Processing)

- ***Digital signal processing*** *(**DSP**) is the mathematical manipulation of an information signal to modify or improve it in some way. It is characterized by the representation of discrete time, discrete frequency, or other discrete domain signals by a sequence of numbers or symbols and the processing of these signals.*

  Wikipedia

- ***Digital signal processing (DSP)*** *is the process of analyzing and modifying a signal to optimize or improve its efficiency or performance. It involves applying various mathematical and computational algorithms to analog and digital signals to produce a signal that's of higher quality than the original signal.*

  Technopedia

Be aware that sometimes DSP = Digital Signal Processor!

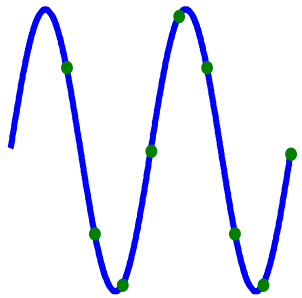
Example of DSP Applications

- Speech & Audio
  - coding, MP3
  - recognition
  - echo cancellation
- Image
  - coding, MPEG4
  - Filtering
- Wireless Communication
  - channel coding/decoding
  - equalization
  - channel estimation
  - smart antennas
    - beam forming
    - MIMO, Multiple Input Multiple Output
- Seismology
  - classification
  - recognition
- Radar and sonar
  - classification
  - detection
- Financial signal processing
  - filtering
  - classification
  - Calculations
  - Bvlock chain technology
- Biomedicin
  - smart sensors
  - telemedicin
  - pacemakers
  - Image processing
- ESS, MAX IV, etc.

Definition DSP (Digital Signal Processor)

- ***Digital signal processor*** *(**DSP**) A digital signal processor (DSP) is a specialized microprocessor, with its architecture optimized for the operational needs of digital signal processing.*

The goal of DSPs is usually to measure, filter and/or compress continuous real-world analog signals. Most general-purpose microprocessors can also execute digital signal processing algorithms successfully, but dedicated DSPs usually have better power efficiency thus they are more suitable in portable devices such as mobile phones because of power consumption constraints.



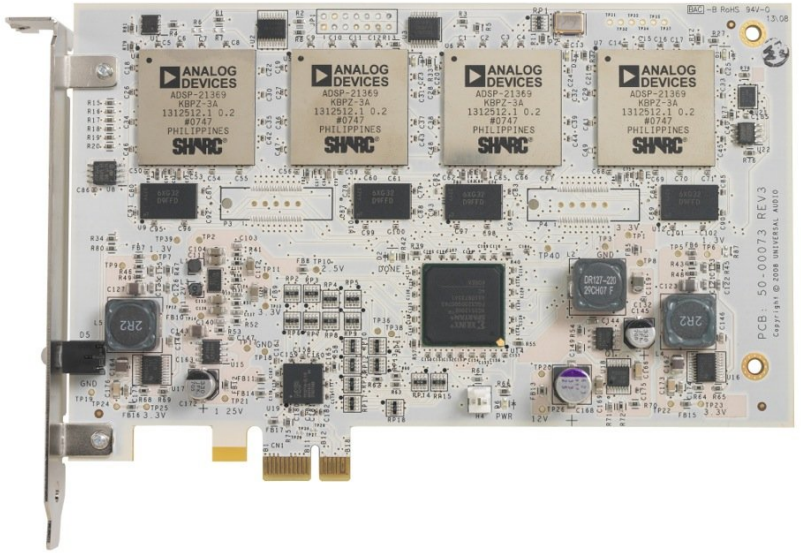
Wikipedia

Different types of Digital Signal Processors

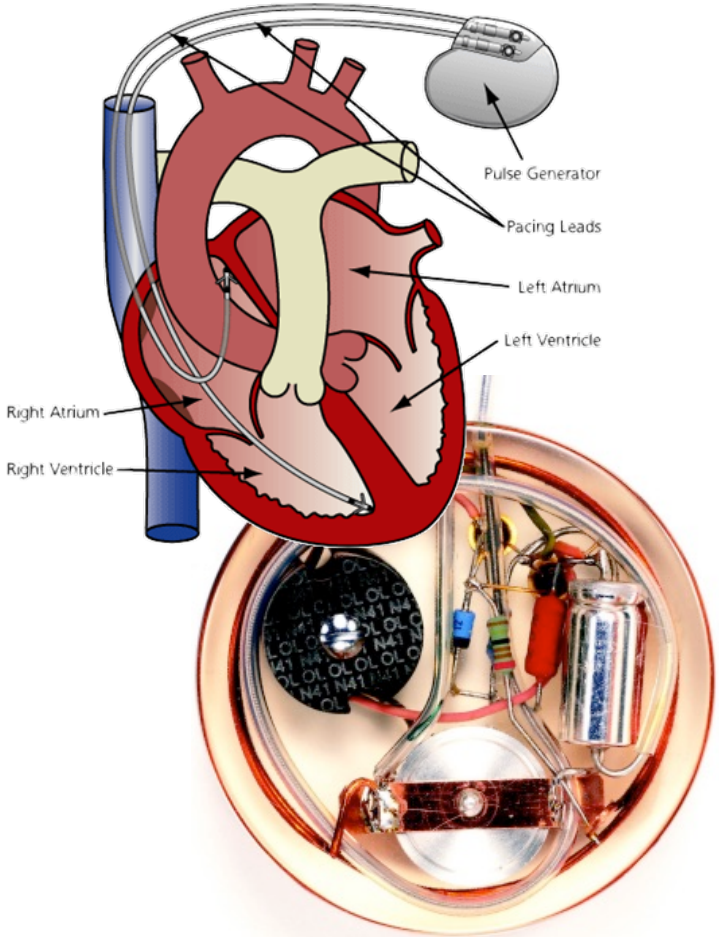
Programmable or Custom DSPs

What to use depends on requirements

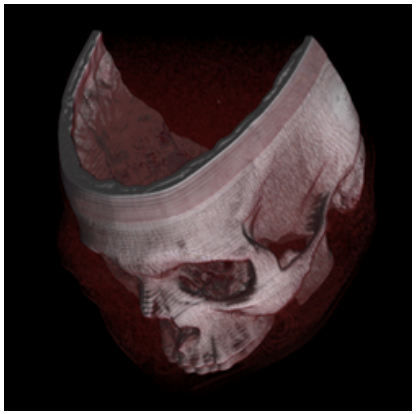
- Sample rate
- Throughput
- Energy consumption
- Area
- Wordlength – precision
- Flexibility
- Time to market
- Volume/size



Where do we find them?



Extremely Low Power



Large volume of data

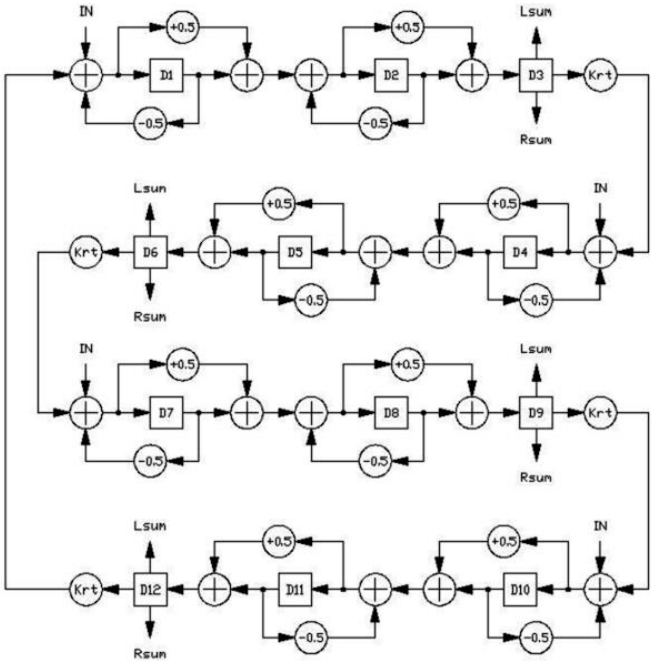
Very Low Power

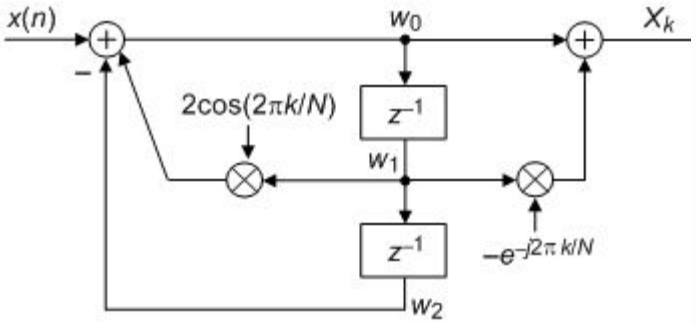
High performance computing

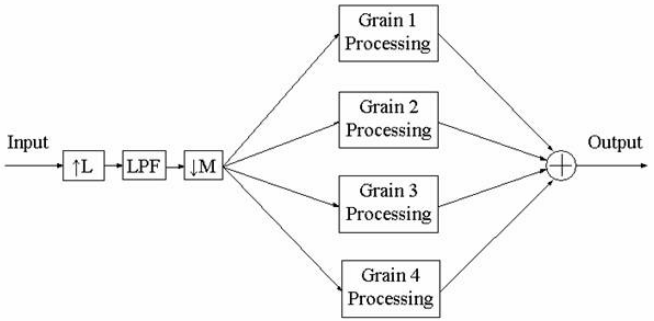
Low Power

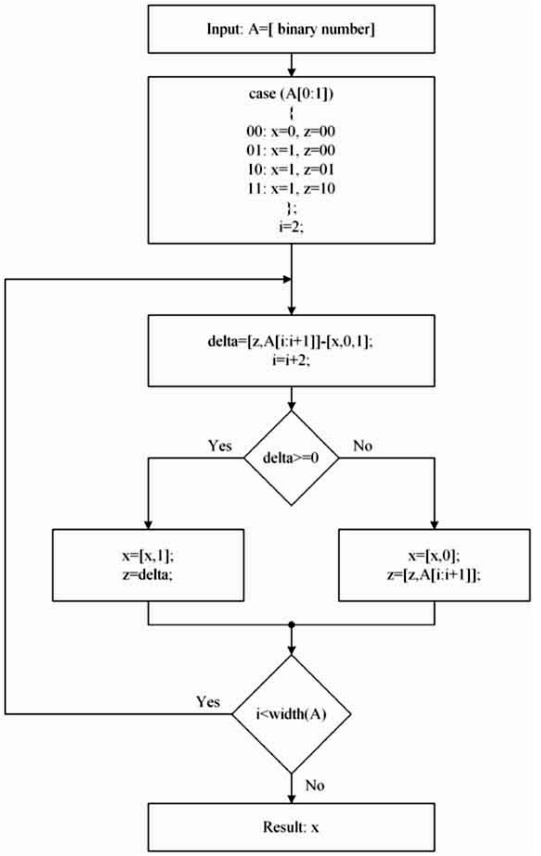
What's happening inside the DSP?

In the DSP an (DSP) algorithm is executed!









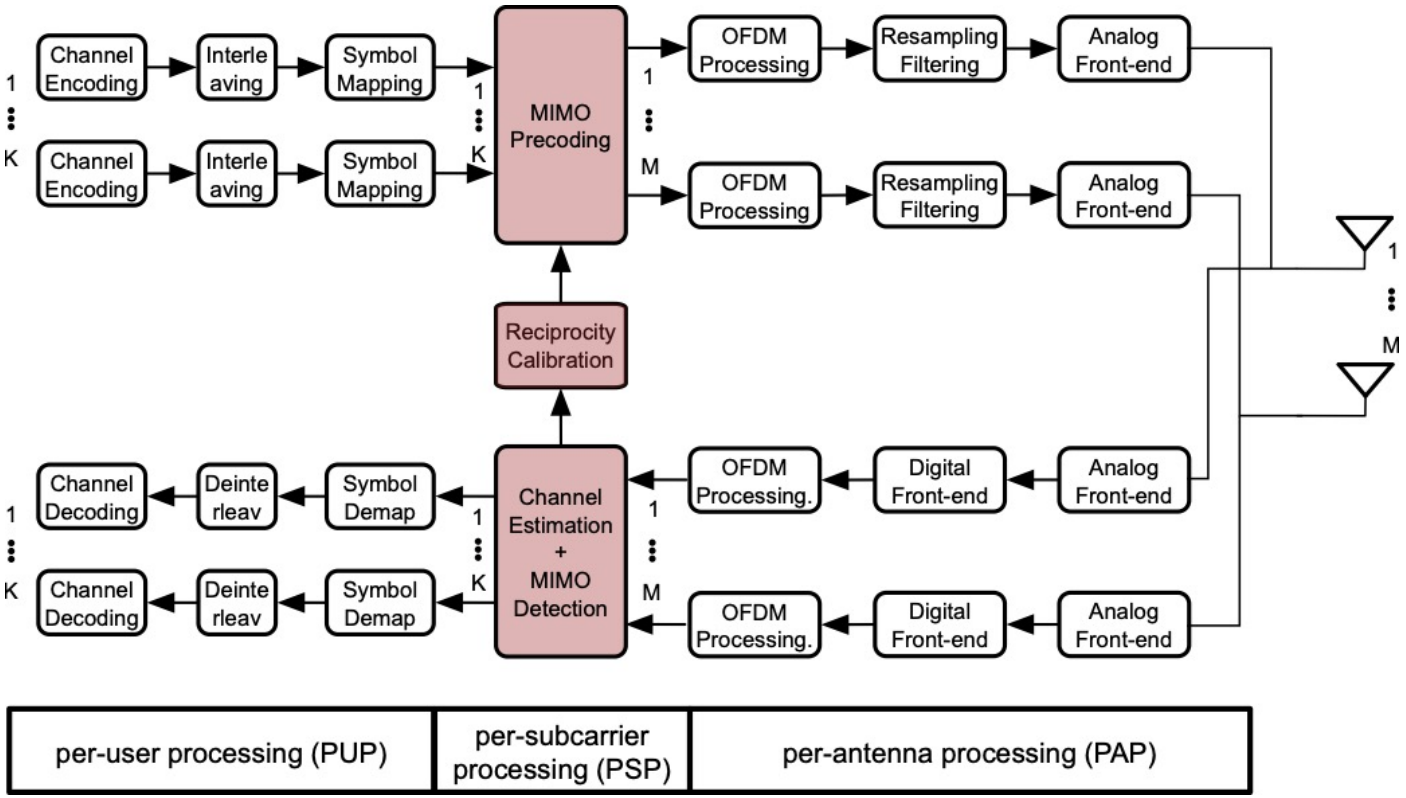
The heart of DSP algorithms are usually DSP ”Primitives”

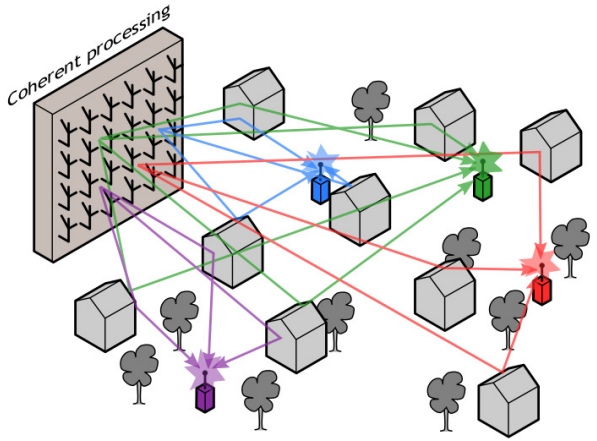
Examples:

- Convolutions
- Filters
  - FIR
  - IIR
  - Wave digital
- Correlation
- FFT - fast Fourier transform
- DCT - discrete cosine transform
- LMS – Least Mean Square
- etc...

An application is often comprised of many different blocks (incl. DSPs)

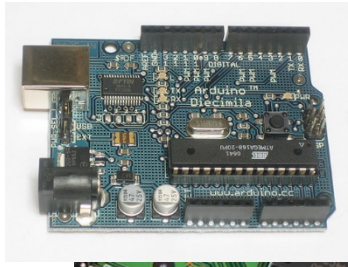
Example: Massive MIMO

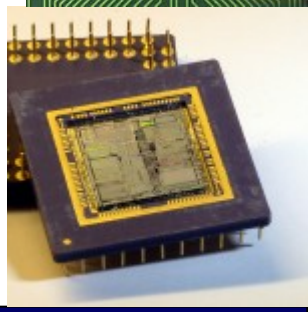




Hardware Implementation Techniques

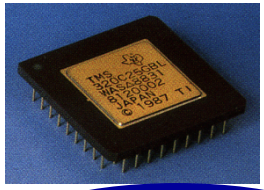
There are several ways of implementing a DSP-algorithm in hardware.





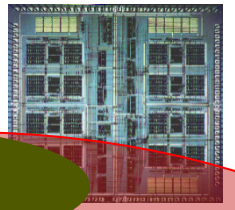
- **Microprocessor/*µ*controller** – is a small computer on a single integrated circuit which may contain a processor core, memory, and programmable input/output peripherals.
- **Digital Signal Processor (DSP)** – a specialized microprocessor, with its architecture optimized for the operational needs of digital signal processing.
- **Field-Programmable Gate Array (FPGA)** - an integrated circuit designed to be configured after manufacturing.
- **Application-Specific Intergated Circuit (ASIC)** - is an integrated circuit customized for a particular use.

Architectural Options –
Standard Processor vs. Special Purpose



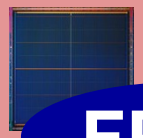
Algorithm

Standard Processor



Special Purpose

- Programable/Flexible
- Short design time/TTM
- Low price?



FPGA

"ASIC"

- High calculation capacity
- Low power consumption
- Low price at volume

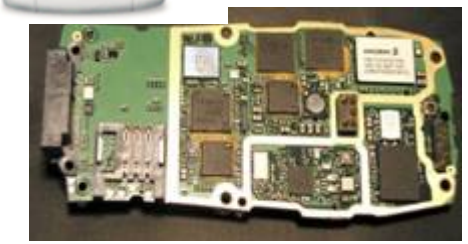
Main focus of this Course

Different applications, different demands...

(a simplified view)

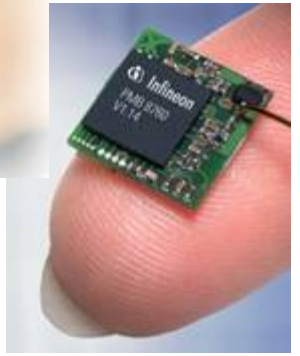
Flexibilty
Complexity

Processors
FPGAs



Low power
Low cost
Flexibilty

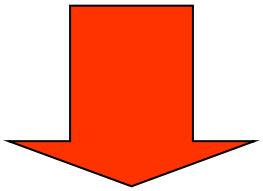
Processors
ASICs



Lower power
Lower cost

ASICs
Processors

This course mainly looks at specialized architectures



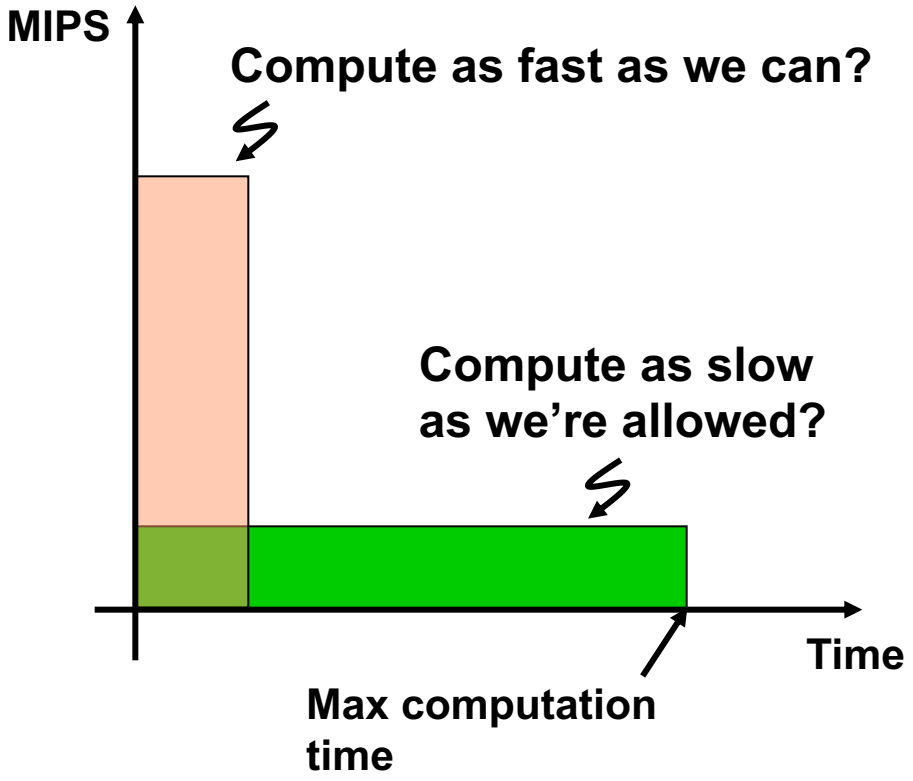
Could be used for either FPGA or ”ASIC”

Energy Efficiency

One of the key design issues!

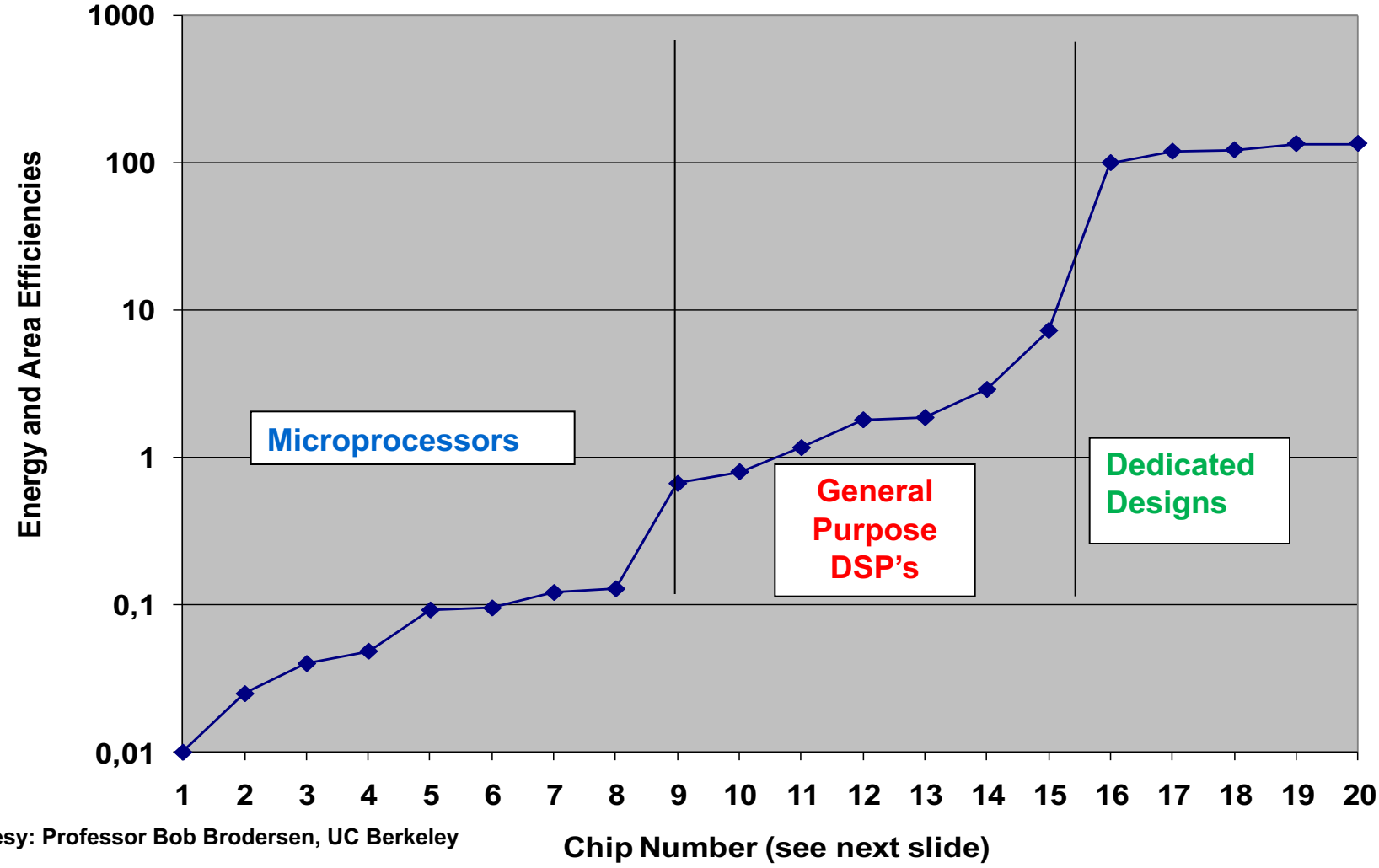
Utilizing the computation time

- Can we control the clock frequency?
- What power down options do we have?
  - clock gating
  - various sleep modes
- Can we scale the power supply?
  - Dynamic
  - How many levels
- What cell library can we choose?
  - Low power
  - High speed



Energy efficiency (MOPS/mW)

depends on type of application



Courtesy: Professor Bob Brodersen, UC Berkeley

Energy efficiency (MOPS/mW) depends on type of application

ISSCC Chips (0.18µm –0.25µm)

- Chips published at ISSCC over a 5-year span

Chip	Year	Paper	Description
1	1997	10.3	S/390
2	2000	5.2	PPC
3	1999	5.2	G5
4	2000	5.1	Alpha
6	1998	15.4	P6
7	1998	18.4	Alpha
8	1999	5.6	PPC

Chip	Year	Paper	Description
9	1998	18.6	Strong-Arm
10	2000	4.2	Comm.
11	1998	18.1	Graphics
12	1998	18.2	Multimedia
13	2000	14.6	Multimedia
14	2002	22.1	MPEG Dec.
15	1998	18.3	Multimedia
16	2001	21.2	Encryption
17	2000	14.5	Hearing Aid
18	2000	4.7	FIR
19	1998	2.1	MPEG Dec.
20	2002	7.2	802.11a

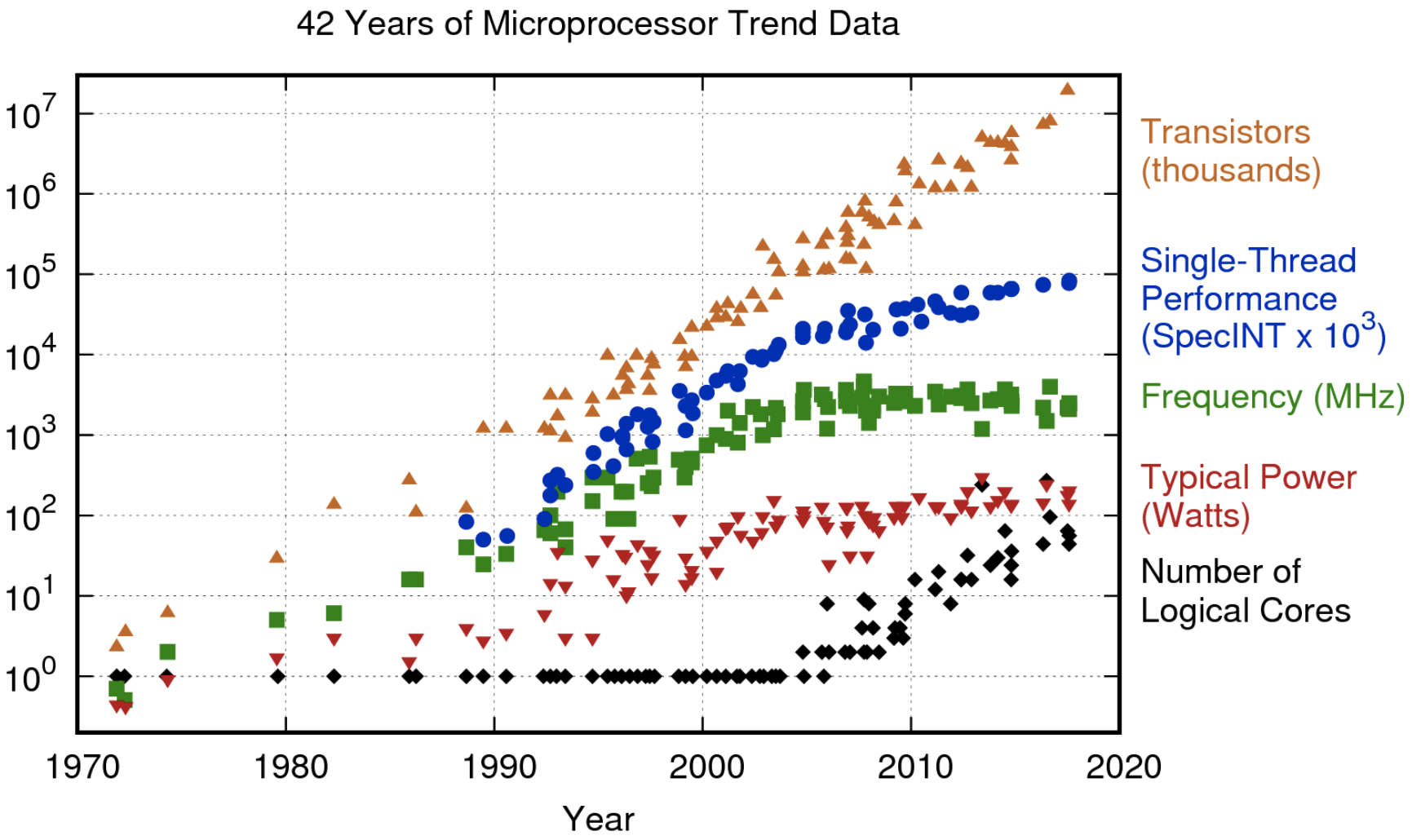
Chip type:

Microprocessor

General purpose DSP

Dedicated design

Trends in processors



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Complexity

A constant challenge!

Complexity

Complexity of Algorithms are increasing with new systems

Number of transistors possible to implement on a die was always increasing (Moore's law)

Often mature algorithms (systems) go to non-custom solutions.

But there is always new algorithms and there is power and price...

Evolution

- **New systems**
  - i.e. high performance
  - use non-standard architectures and components
  - e.g. 5G

- **Mature systems**
  - i.e. low performance compared to state of the art
  - implemented on standard platforms
  - mature technologies
  - e.g. GSM, 3G

Important questions when designing hardware architectures

Which structure gets the job done?

Which structure use the least amount of energy?

Which structure use the least amount of area?

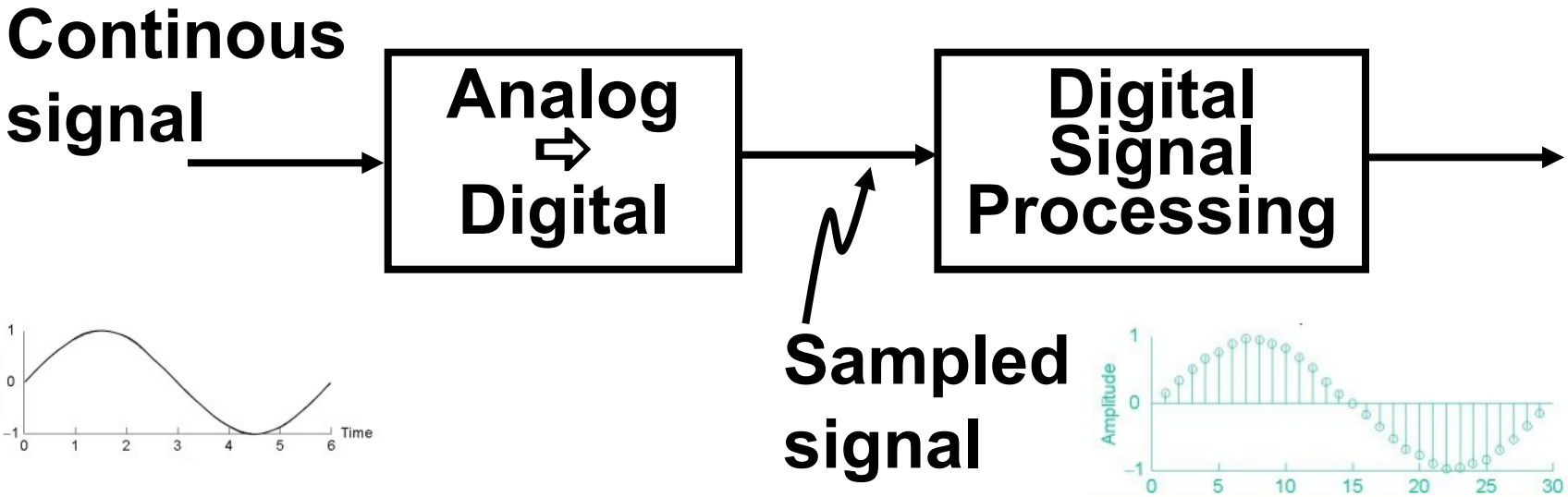
Etc, etc, etc...

How do we design architectures to achieve it?

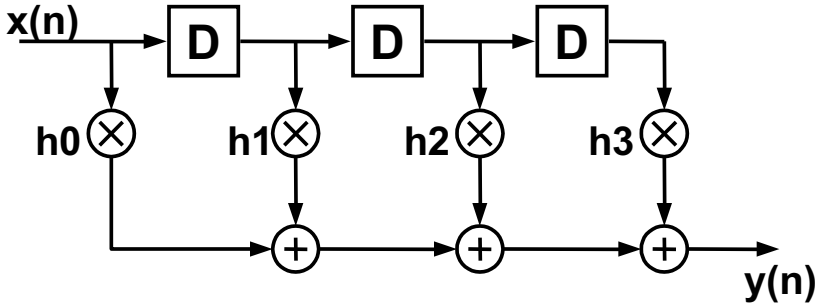
DSP Basics
Filters

Digital signal processing algorithms works on samples of a continous signals.

Sampling rate = nr. of samples processed/second



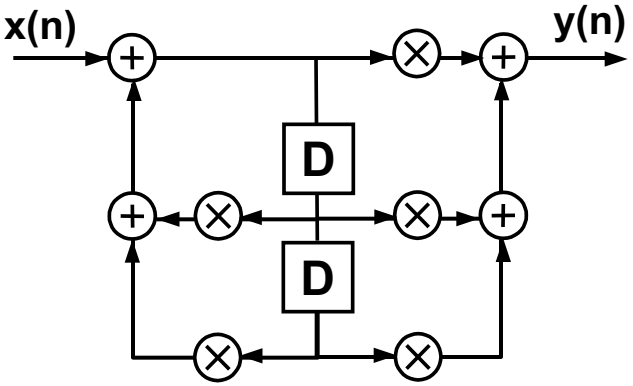
Two Basic DSP Structures



FIR – Finite Impulse Response

4-tap FIR filter

No feedback



IIR – Infinite Impulse Response

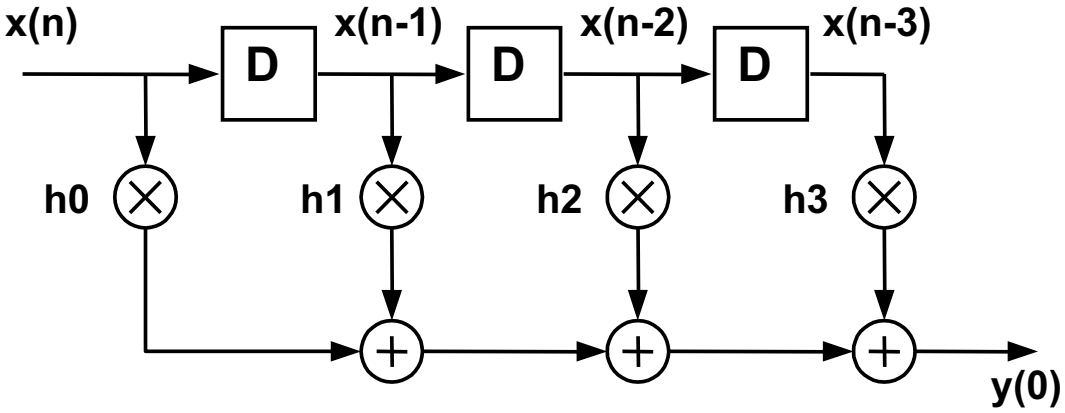
Biquad section

Feedback

The FIR filter

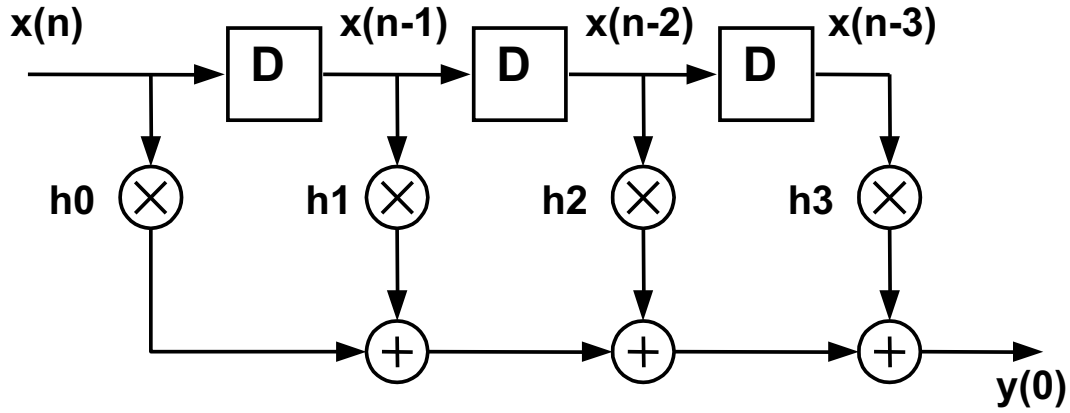
$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

h(.) **is the impulse response which defines the filter response, e.g. low- or highpass.**



The FIR filter - Definitions

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$



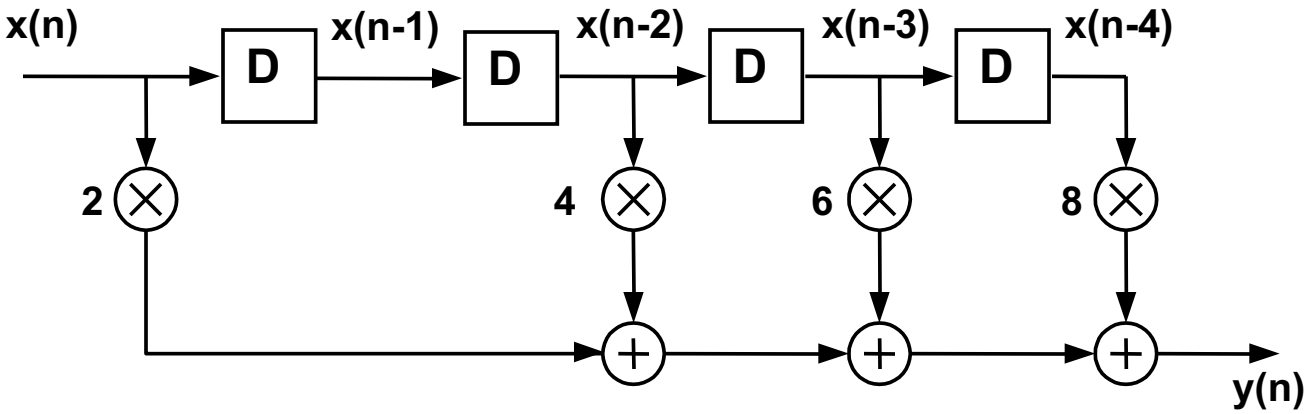
- The filter length is = *N*
- The filter order is = *N-1*
- The number of filter taps = the filter length

A higher order filter, more taps, will result in a steeper filter function but has higher complexity!

Quick look at filter order, length and taps

Suppose that we have the following filter:

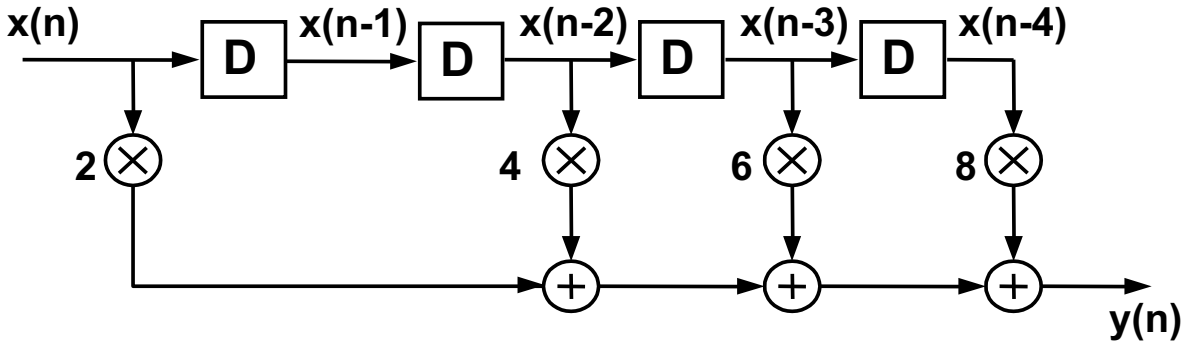
$$y[n]=2x[n]+4x[n-2]+6x[n-3]+8x[n-4]$$



What is the filter length, filter order and number of taps?

Quick look … answer

$$y[n]=2x[n]+4x[n-2]+6x[n-3]+8x[n-4]$$

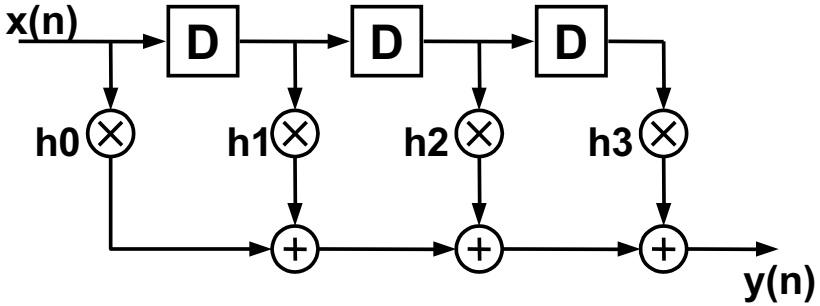


Filter length: the filter length is 5, i.e. the filter extends over 5 input samples [x(n),x(n−1),x(n−2),x(n−3),x(n−4)].

Filter order: The order of an FIR filter is filter length minus 1, i.e. the filter order in the example is 4. (The filter order is the max. delay needed, so if your filter is y(n)+y(n−10)=x(n) you have a filter order of 10)

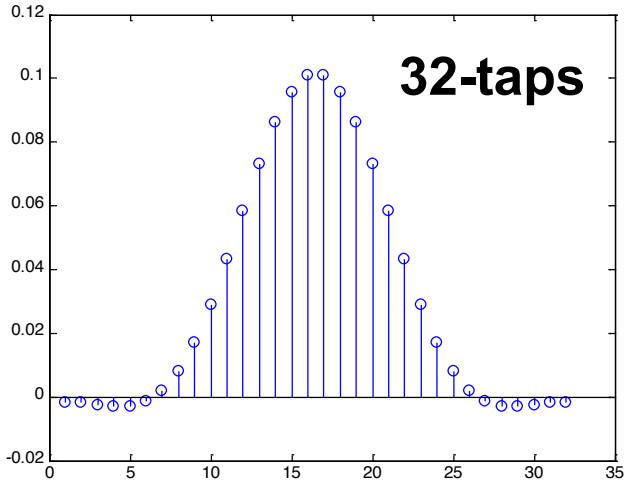
 # filter taps: The number of taps is the same as the filter length. In this case you have one tap equal to zero (the coefficient for x(n−1)), so there is 4 non-zero taps. Still, the filter length is 5.

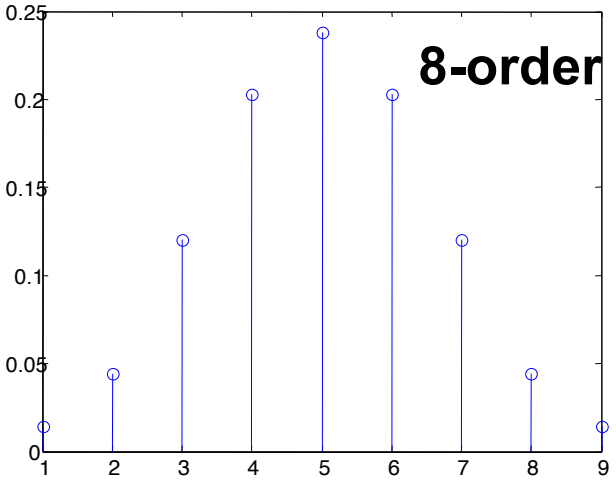
Example: FIR filter in Matlab



FIR-filters can be designed with the built-in filter function

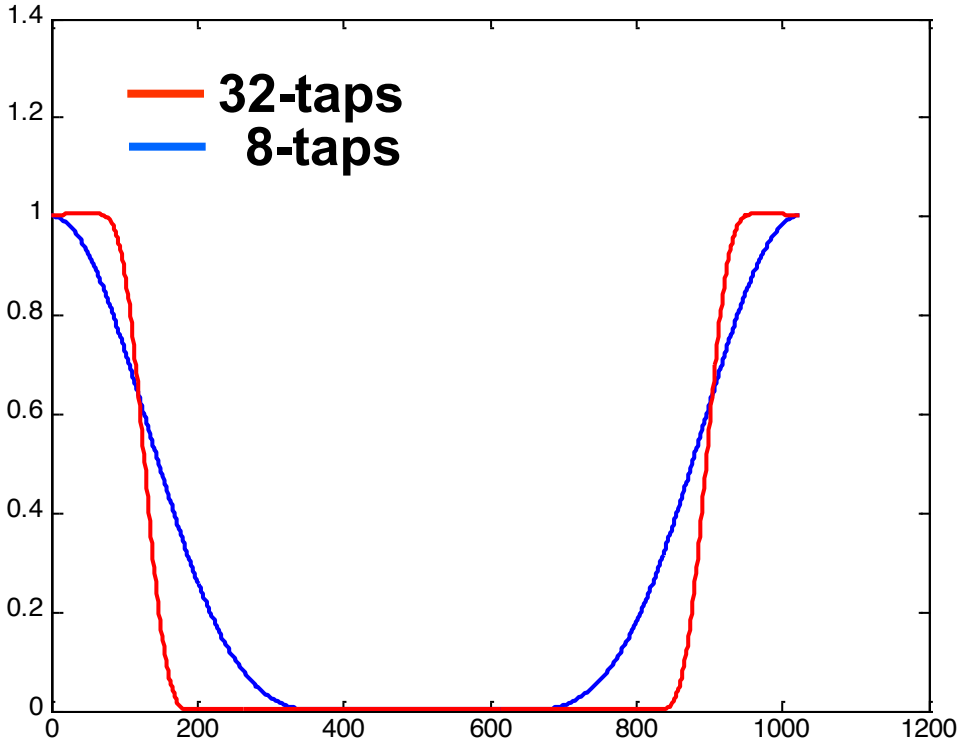
***fir1(N,Wn)* – *N*'th order filter with the cut-off frequency *Wn* must be between *0 < Wn < 1.0*, with *1.0* corresponding to half the sample rate.**





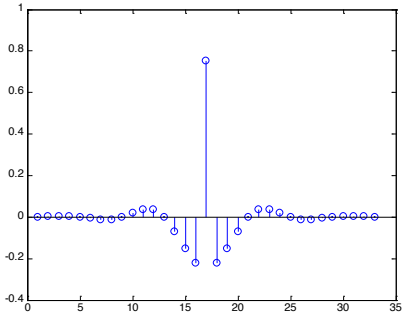
FIR-filter frequency response

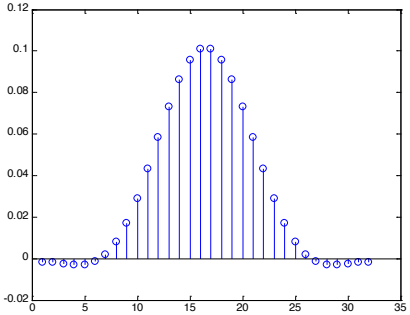
Use *fft* to transform *h(.)* to frequency domain and plot.



Symmetry when real input to *fft*.

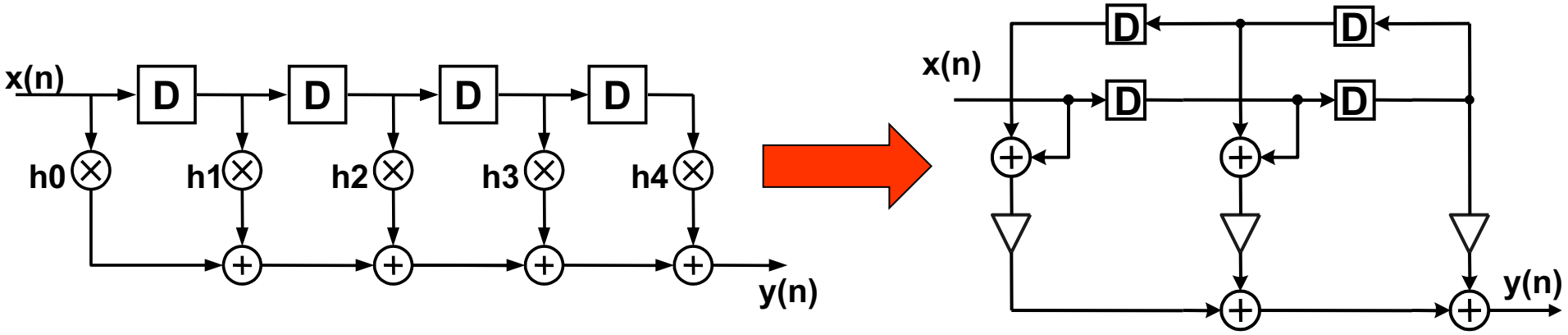
Linear phase FIR filters





Linear phase filters has a constant group delay in the passband, i.e. all frequency components are delayed equally ⇨ no phase distortion!

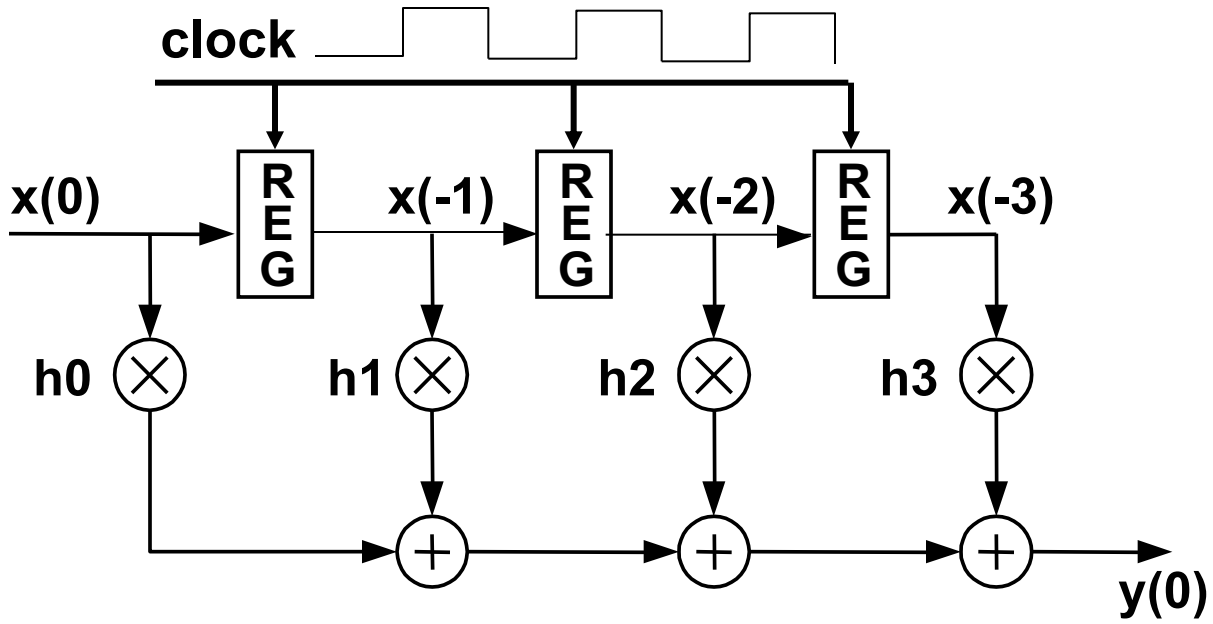
Linear phase filters, e.g. from *fir1()*, has symmetric coefficients.
This can be used to simplify the filter structure.



The FIR filter, *hardware mapped, first clock cycle*

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

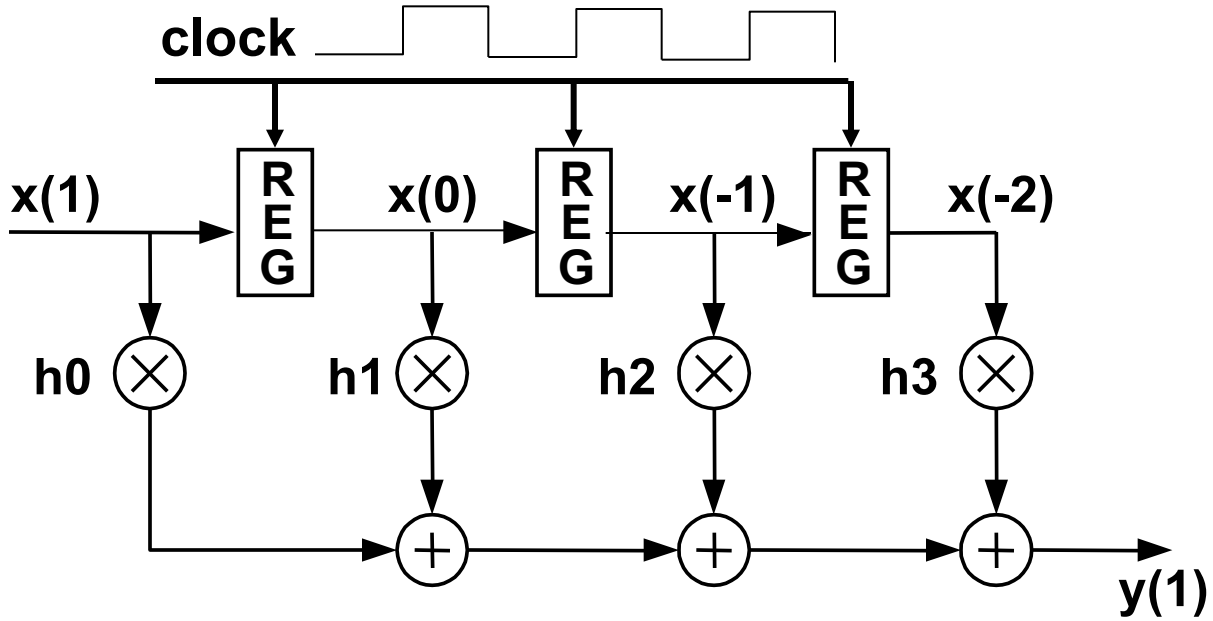
$$y(0) = h_0 x(0) + h_1 x(-1) + h_2 x(-2) + h_3 x(-3)$$



The FIR filter, *second clock cycle*

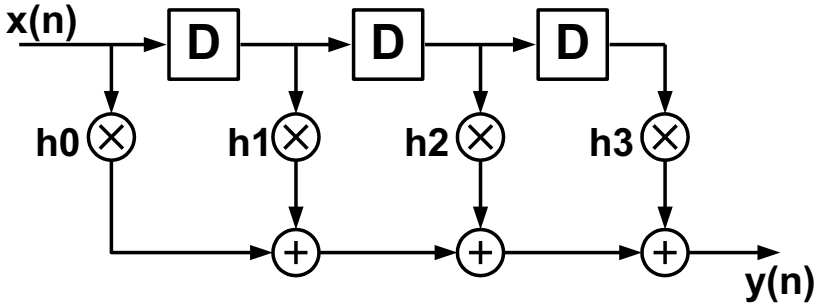
$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

$$y(1) = h_0 x(1) + h_1 x(0) + h_2 x(-1) + h_3 x(-2)$$

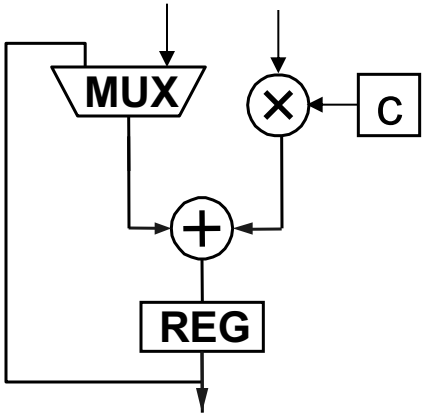


Time multiplexed to save hardware

$$FIR: y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

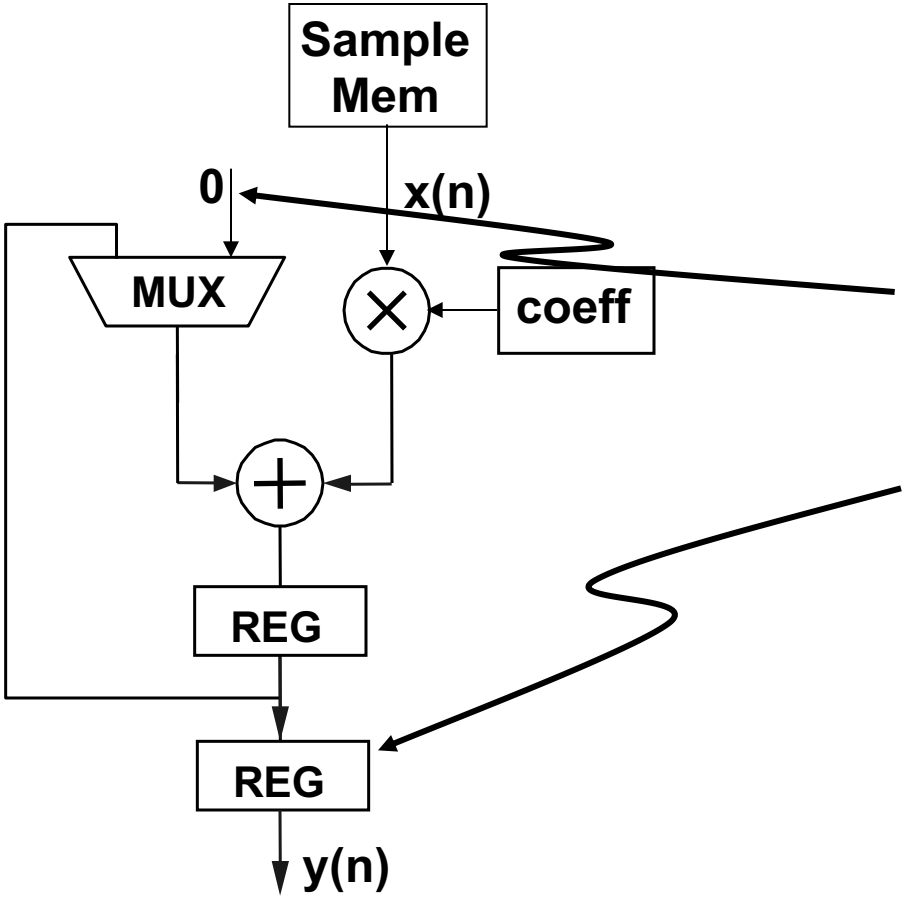


1	sample/cc
N	fixed multipliers
N-1	adders



N	cc/sample
1	generalized multiplier
1	adders
1	coefficient memory
+	control

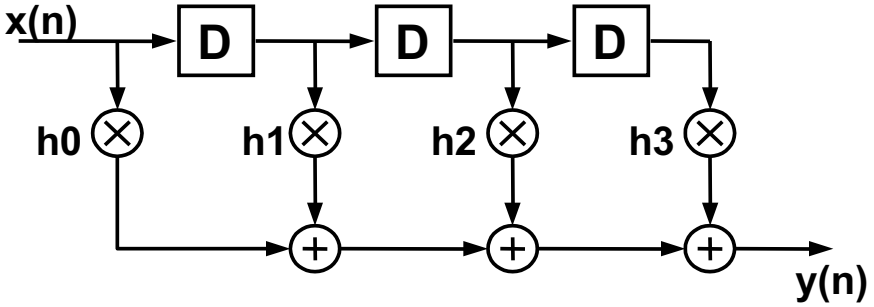
Time multiplexed to save hardware



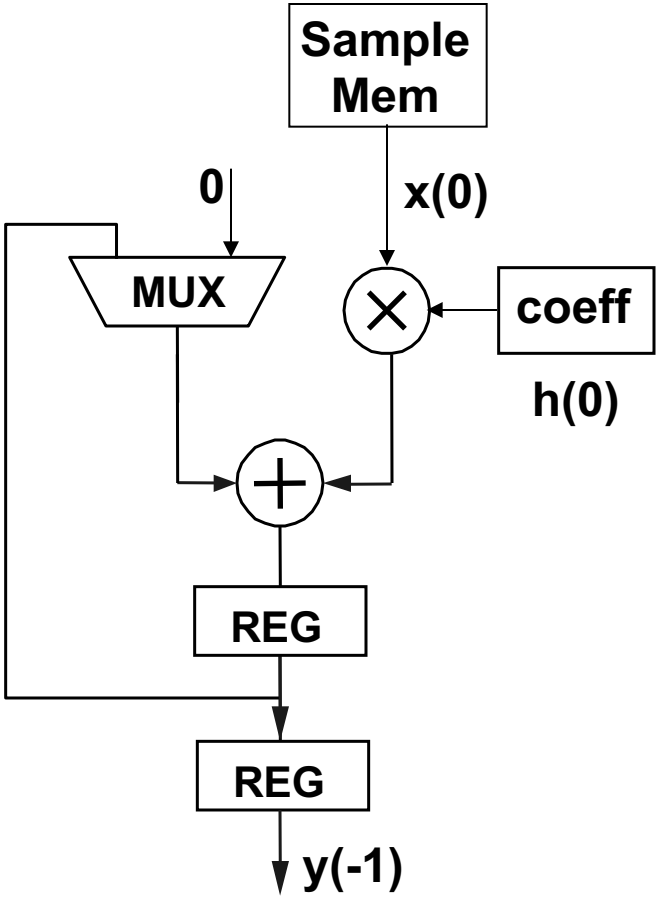
How many clock cycles?

Why the ”0”?

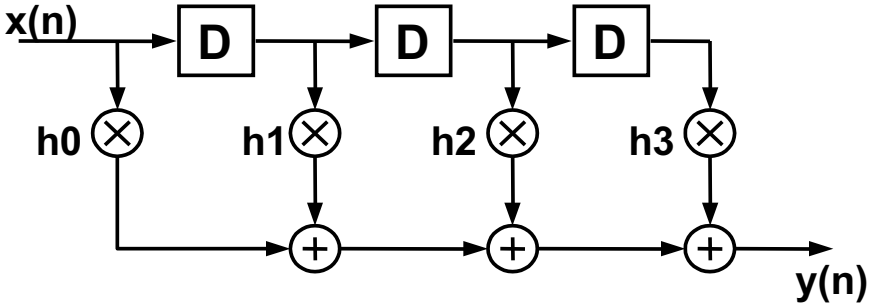
Why the extra reg?



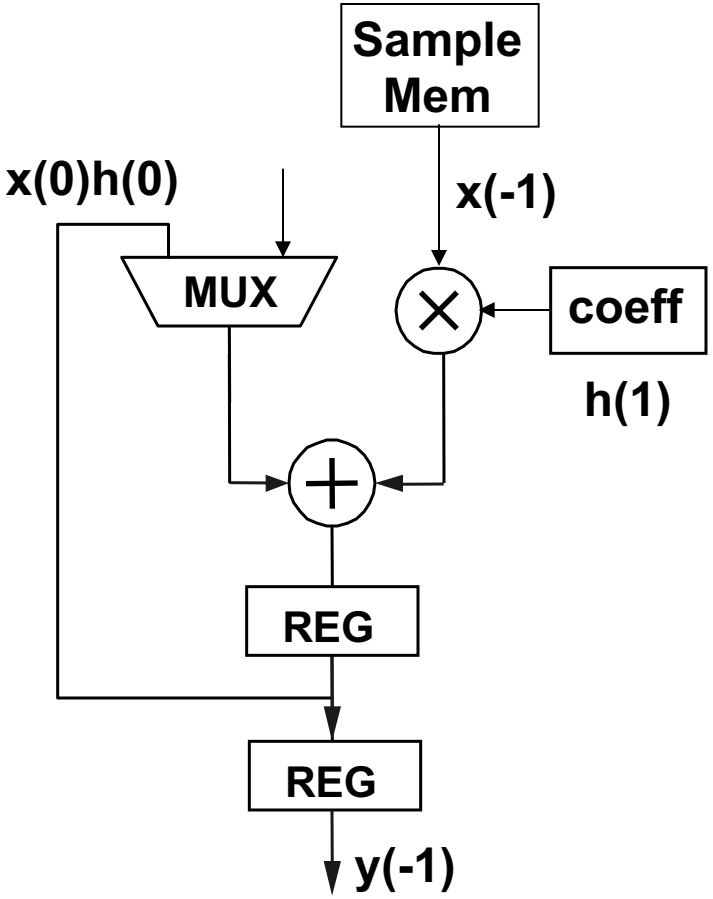
Time multiplexed to save hardware



cc0: x(0)h(0)+0

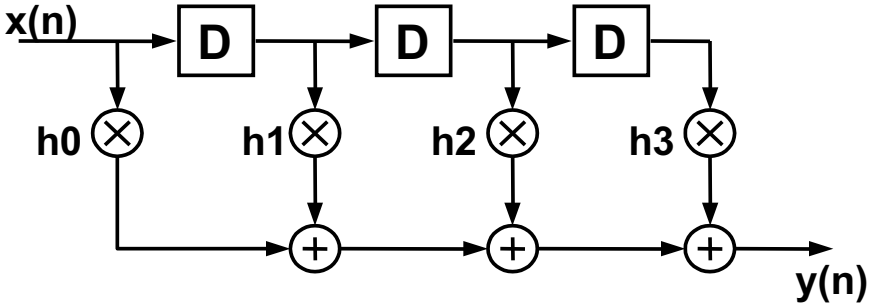


Time multiplexed to save hardware

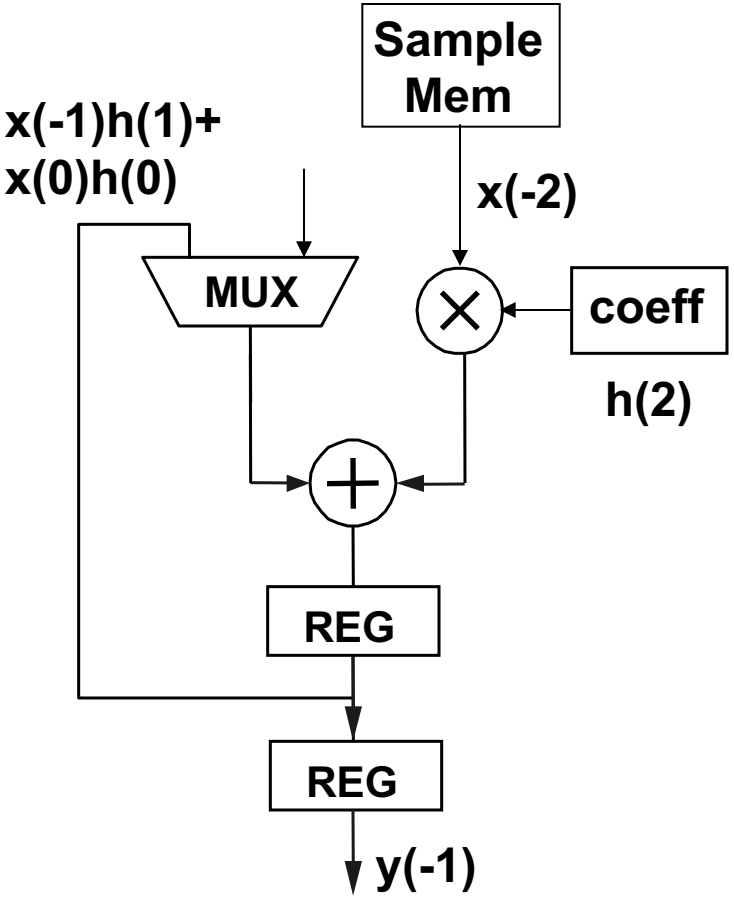


cc0: x(0)h(0)+0

cc1: x(-1)h(1)+x(0)h(0)



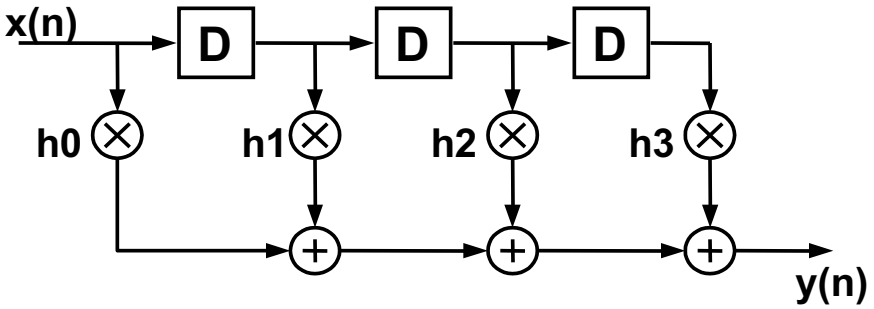
Time multiplexed to save hardware



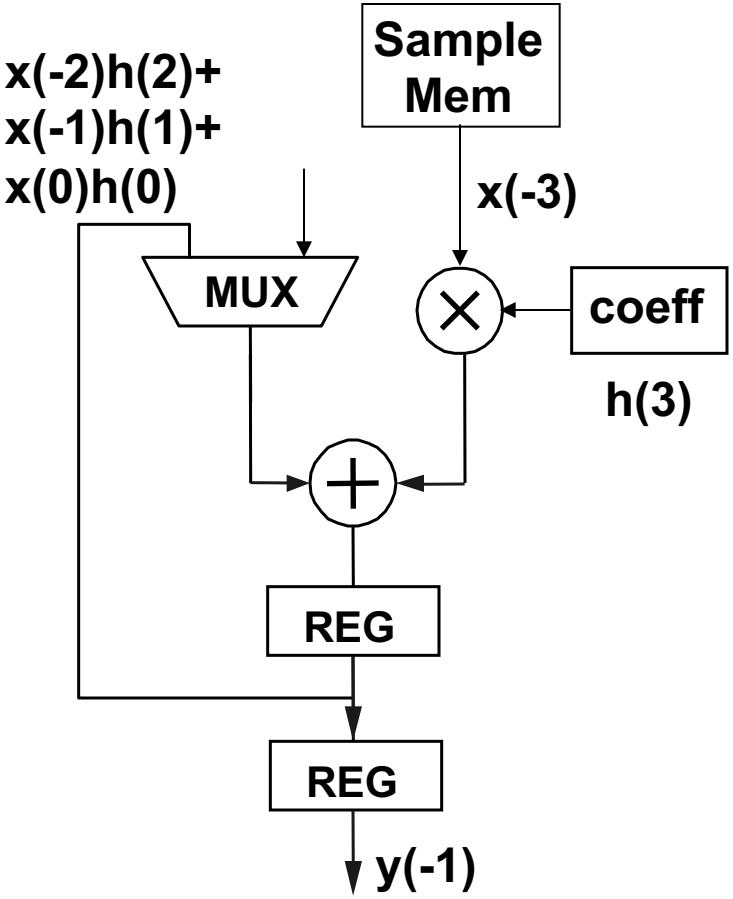
cc0: x(0)h(0)+0

cc1: x(-1)h(1)+x(0)h(0)

cc2: x(-2)h(2)+ x(-1)h(1)+x(0)h(0)



Time multiplexed to save hardware

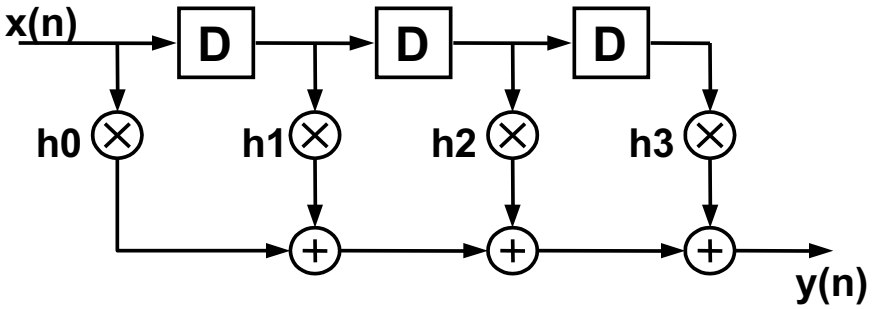


cc0: x(0)h(0)+0

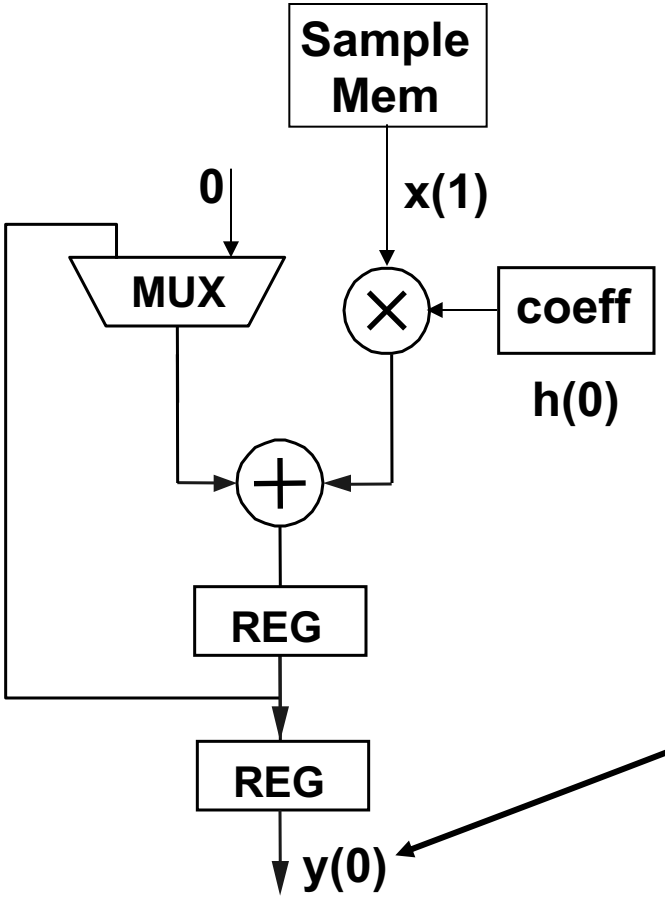
cc1: x(-1)h(1)+x(0)h(0)

cc2: x(-2)h(2)+ x(-1)h(1)+x(0)h(0)

cc3: x(-3)h(3)+ x(-2)h(2)+ x(-1)h(1)+x(0)h(0)



Time multiplexed to save hardware



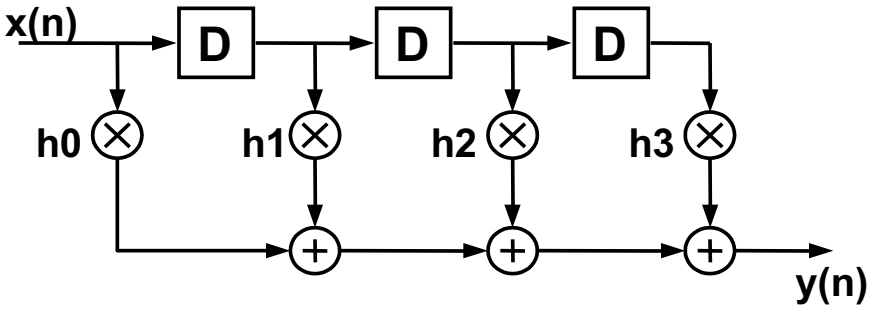
cc0: x(0)h(0)+0

cc1: x(-1)h(1)+x(0)h(0)

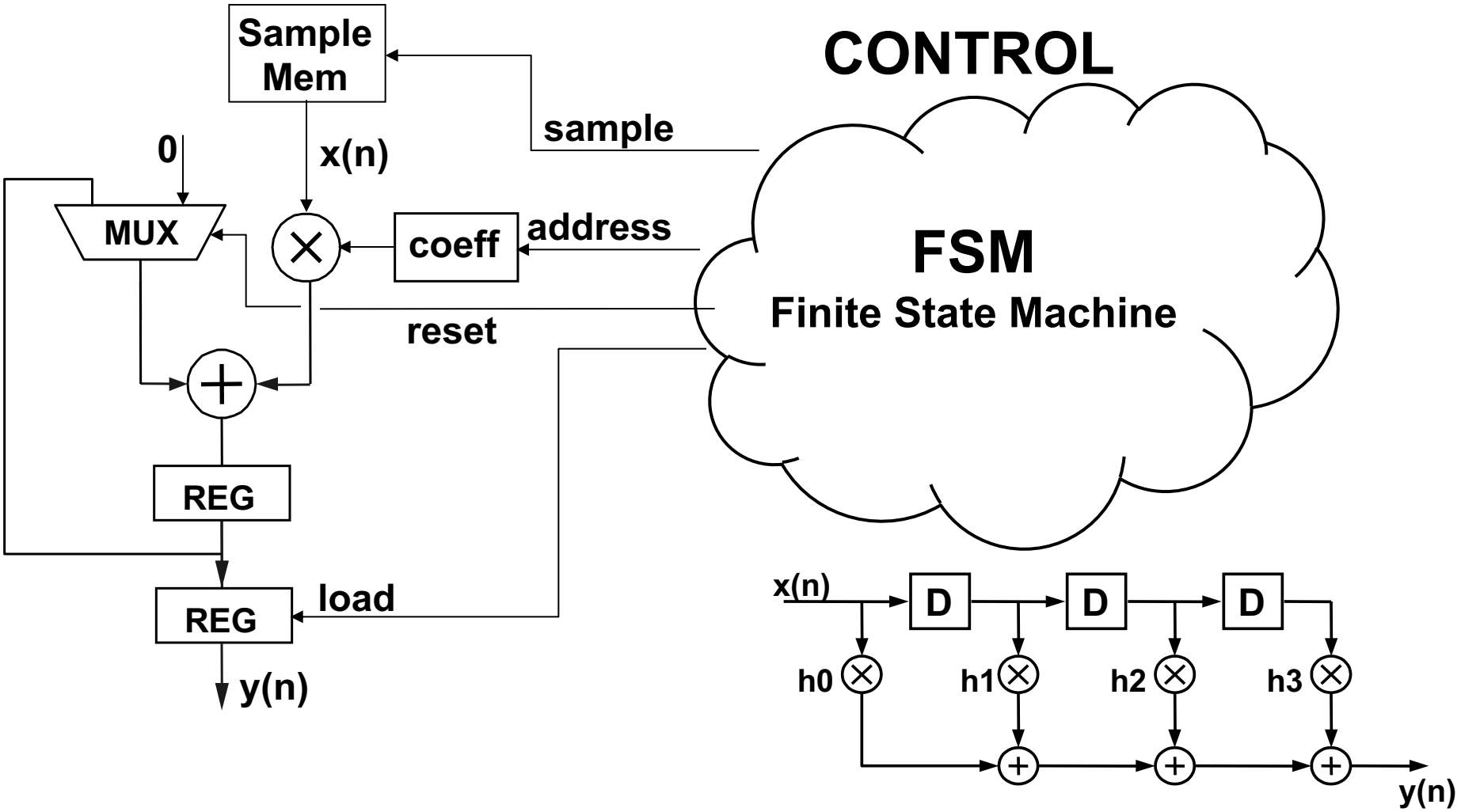
cc2: x(-2)h(2)+ x(-1)h(1)+x(0)h(0)

cc3: x(-3)h(3)+ x(-2)h(2)+ x(-1)h(1)+x(0)h(0)

cc4: x(1)h(0)+0; new iteration



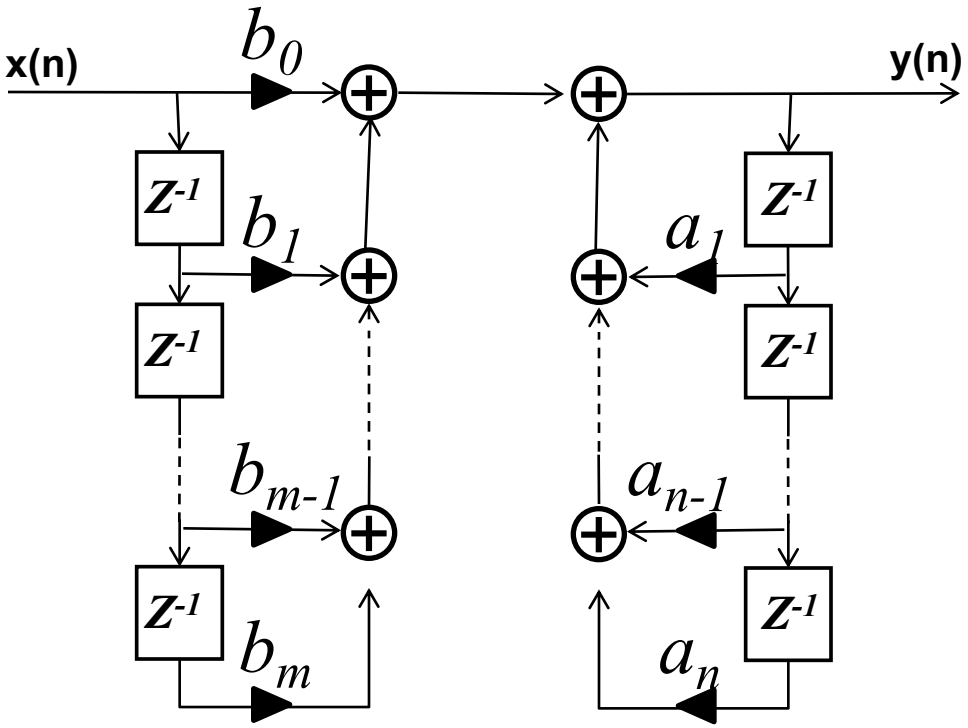
Time multiplexed to save hardware



The IIR filter, direct form I

The impulse response also includes feedback terms.

$$y(n) = \sum_{i=0}^{m} b_i x(n-i) + \sum_{j=1}^{n} a_j y(n-j)$$

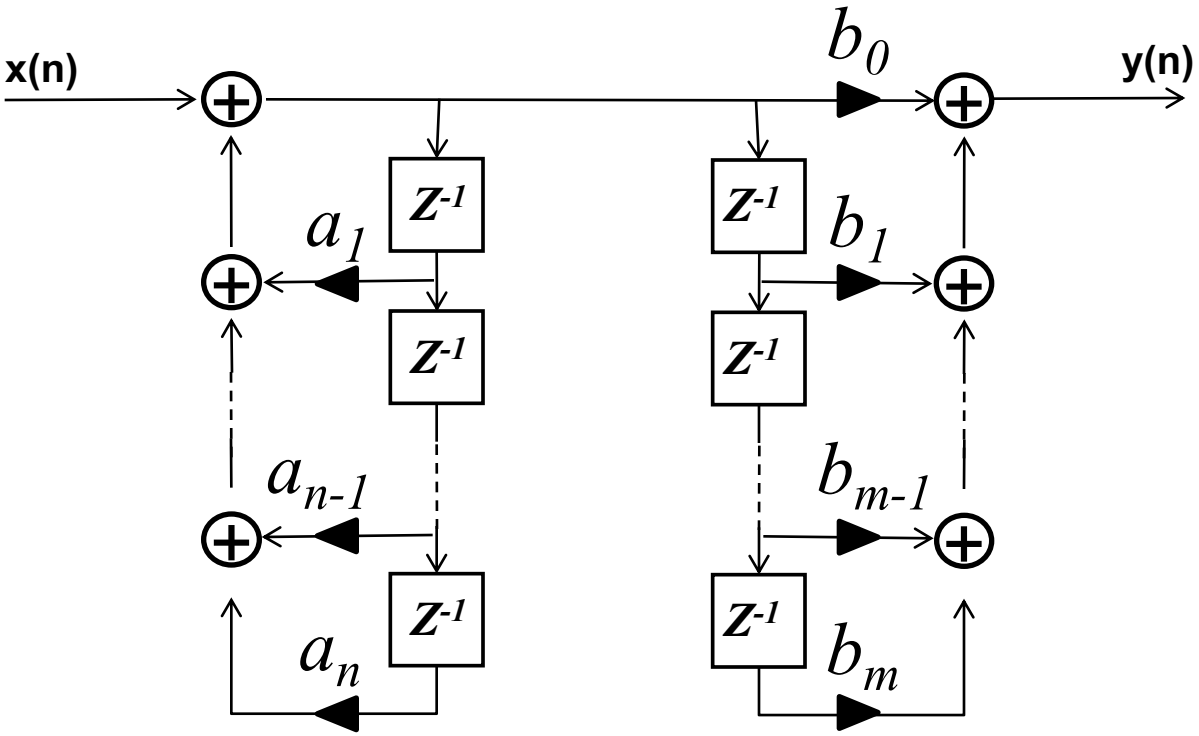


- **Steeper impulse response but possibility for instability**

The IIR filter, direct form II

$$y(n) = \sum_{i=0}^{m} b_i x(n-i) + \sum_{j=1}^{n} a_j y(n-j)$$

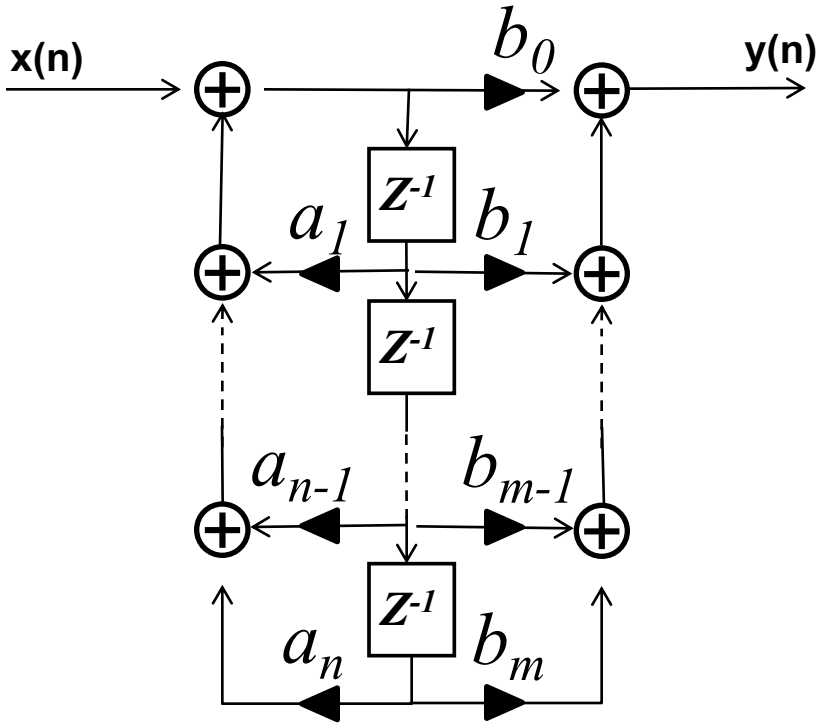
Each part is a linear time-invariant system and the order can be reversed.



The IIR filter, direct form II

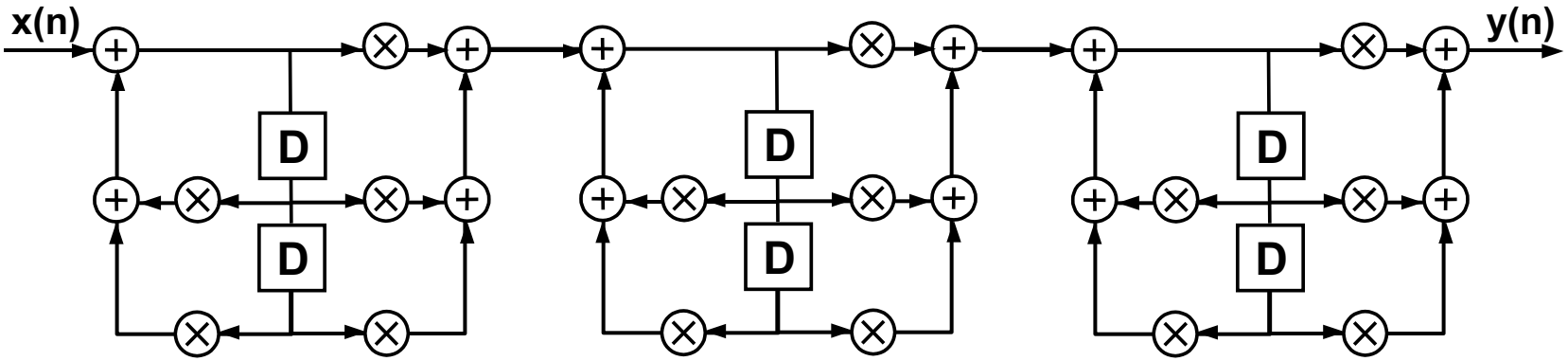
$$y(n) = \sum_{i=0}^{m} b_i x(n-i) + \sum_{j=1}^{n} a_j y(n-j)$$

The two parts can be collapsed into one with a minimum number of delay elements.



The IIR filter, cascade form

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}}; \; Ns = \lfloor (N+1)/2 \rfloor$$



- Often cascaded with shorter sections which are combined, easier to design when fixed-point arithmetic.
- The above is often referred to as biquad sections.

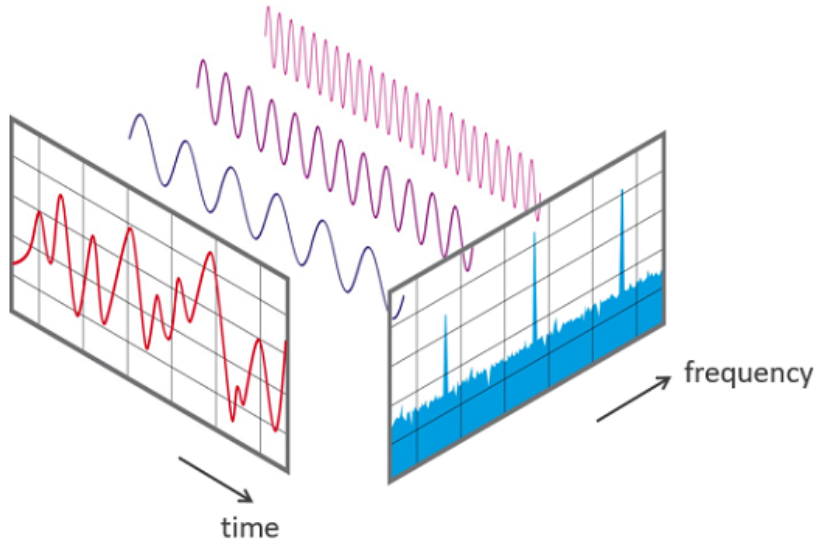
DSP Basics

DFT - FFT

DFT - FFT

- The Discrete Fourier Transform (DFT) is a mathematical operation. The Fast Fourier Transform (FFT) is an efficient algorithm for the evaluation of that operation (actually, a family of such algorithms).

- The fast Fourier transform (FFT) samples a signal over a period of time (or space) and divides it into its frequency components.



DFT - FFT

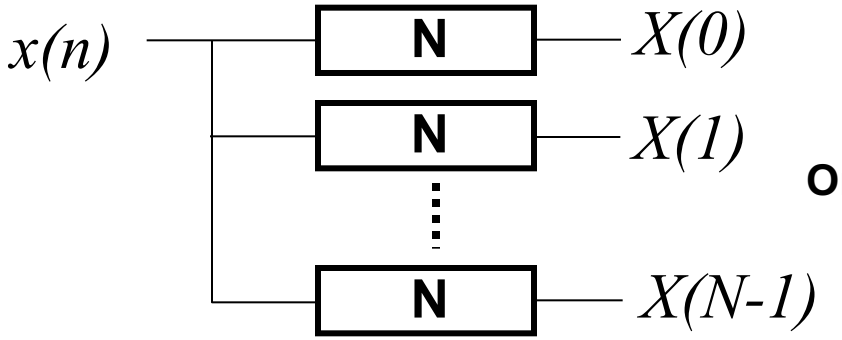
- The DFT/FFT is one of the most common digital signal processing algorithms.
- Used to determine frequency content of a discrete signal sequence.
- Transform between time and frequency domains.
- The FFT is a low complexity way of computing the DFT.

N-point DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0,1,\ldots,N-1 \qquad W_N^{kn} = e^{-j2\pi kn/N}$$

Complex

N filters of length N ⇨ $O(N^2)$



Only every N^th sample

- **The DFT determines spectral content at N equally spaced frequency points, i.e. coorelates with different frequencies,**

$$f_{analysis}(m) = \frac{mf_{sample}}{N}$$

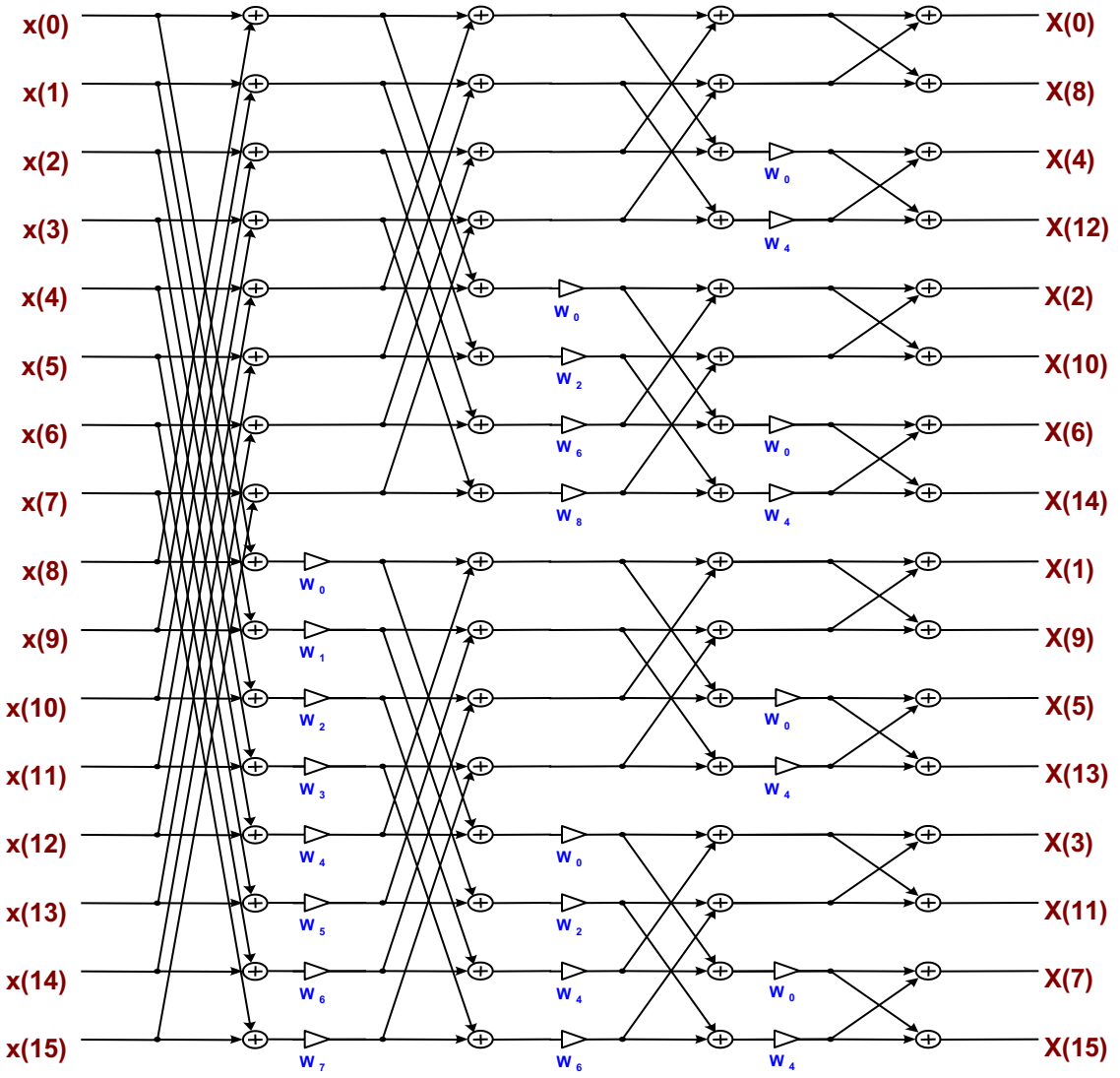
- **N samples are needed.**

FFT is low complexity DFT

$\log_2(N)$ **stages**

DFT $O(N^2)$

FFT $\frac{N}{2}\log_2(N)$



End of Lecture 1

See you on Thursday