Digital Signal Processors

2012, Course Overview

Introduction

- Digital Signal Processing Systems, History, Applications
- Processor Architectures and Design Philosophies
- CISC, RISC, ILP, Superscalar, VLIW, Sequential, Pipelined
- Von Neumann, Harvard and Modified Harvard Architectures
- Where do DSPs Stand? Why go DSP? DSP Design Principles
- Historical Evolution of DSPs

DSP System Elements and Related Issues

- DSP Processors Elements: Memory, ALU, Bus, Cache, Peripherals,...
- ADCs and DACs
- Quantization Effects, Overflow, Saturation Arithmetic, Signal Scaling, Guard Bits
- Numerical Representation, Fixed and Floating Point, Q Format
- Pipelining Methods, Data Stationary, Time Stationary, Interlocking
- DSP Performance

Digital Signal Processors

Course Overview ...

Real-Time Signal Processing

- DSP System Design, Development Tools, MATLAB, CCS
- DSP Chips Review
- Older DSPs
- Modern DSPs: TI: TMS320C5X, TMS320C6X, ADI: ADSP-21xx, ADSP-2116X, Shark, Tiger Shark, OMAP, Davinci...
- DSP Applications
- DSP Selection
- Code Optimization, C, Linear Assembly, Assembly on TI C6000

DSP Algorithms, Mapping, Implementation, Optimization

- Filtering, FIR, IIR, Multi-rate Filters, Adaptive Filters
- Arithmetic Operations, DFT, FFT, Goertzel, Spectral Estimation
- Signal Generation, Waveforms, Random Signals
- Speech and Image Processing
- Examples of Optimization on other DSPs

Digital Signal Processors

Course Overview ...

Dedicated DSP Design Examples

- System Level Design
- System Level Optimization
- Porting Operating Systems: Android, Angstrom Linux,...

Embedded DSP Systems Design

- Interfaces and Peripherals
- Interfacing to DSP, High Speed Interfaces, Serial/Parallel Interfaces, IO, Analog, Codec
- Memory and Power Consumption
- Hardware for DSP, Typical Applications
- DSP Board Design: Low Voltage Interfaces, Mixed Signal Problems, Grounding, Isolation,
- Power Supply Noise Reduction and Filtering, High Speed Logic

Digital Signal Processors

Text books

Chapters from:

1) "Embedded DSP Processor Design", D. LIU, Morgan Kaufmann, 2008, ISBN:978-0-12-374123-3.
2) "DSP Software Development Techniques for Embedded and Real-Time Systems", Elsevier 2006, R. Oshana, ISBN:10: 0-7506-7759-7.
3) "Real-time Digital Signal Processing", S. M. Kuo, B. H. Lee, W. Tian, John Wiley & Sons Ltd, 2006, ISBN: 0-470-01495-4.
4) "Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK",R. Chassaing, D. Reay, 2nd Edition, John Wiley & Sons Ltd, 2008, ISBN 978-0-470-13866-3.
5) "Programmable Digital Signal Processors", Y. H. Hu, 2001 by Marcel Dekker, Inc., ISBN: 0-8247-0647-1.
6) "Mixed-Signal and DSP Design Techniques", Analog Devices Inc. 2003, W. Kester, ISBN: 0750676116.
7) "Real-Time Digital Signal Processing Based on the TMS320C6000", Elsevier Inc. 2005, Nasser Kehtarnavaz, ISBN: 0-7506-7830-5.

Digital Signal Processors

Other References

1) "Guide to RISC Processors, S. P. Dandamudi, Springer Science+Business Media, Inc. 2005, ISBN 0-387-21017-2.
2) "Dedicated Digital Processor", F. Mayer-Lindenberg, John Wiley & Sons 2004, ISBN: 0-470-84444-2.
3) "Synthesis and Optimization of DSP Algorithms", G. A. Constantinides, P. Y. K. Cheung, W. Luk, Kluwer Academic Publishers 2004, ISBN:1-4020-7931-1
4) Selected papers
   1) "Programmable DSP Architectures, Part 1 and 2", E. A. Lee, IEEE ASSP Magazine Oct 1988 and Jan 1989.
   2) "DSP Processors Evolution", IEEE Signal Processing Magazine, March 2000.
   3) "How to Estimate DSP Processor Performance", IEEE Spectrum July 1996.

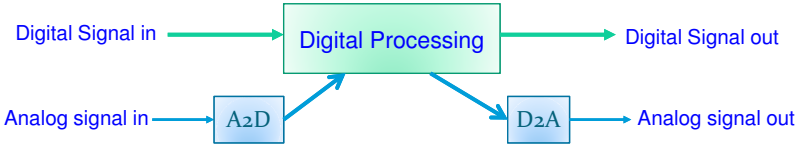
   ....
5) Web Sites and Teaching ROMs

   www.ti.com , www.analog.com , www.freescale.com

   www.bdti.com, ...

Digital Signal Processing Systems

Different Types Based on Inputs and Outputs



Digital Processing: Operation, Transformation, Filtering, Manipulation, ...

Examples

Speech/Speaker Recognizer
CD/DVD Players
Graphic Card
Mobile Phones

Digital Signal Processing Systems...

Analog Processing

Analog Computers
Fourier Optics

Why Go Digital Processing?

- Programmability
  - One hardware can perform several tasks
  - Upgradeability and flexibility
- Repeatability
  - Identical performance from unit to unit
  - No drift in performance due to temperature or aging
- Immune to noise
- Offering higher quality or performance

  (Compare CD players versus phonographic turntable)

Digital Signal Processing Systems...

DSP Systems Usually are

1) Embedded Systems
2) Real-Time Systems
3) High Computational Performance
4) Low Cost/Power/Size

Embedded Systems:

Computer systems designed to perform one or a few tasks
Examples: MP3 players, Traffic Control Systems, Radars, ...
Comparing to GP computers which are flexible to end user needs
Can be optimized for cost, size, power consumption, ...

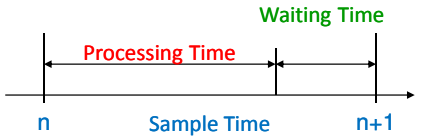
Real Time Systems:

Systems that are subjected to a "real-time constraint"
~ operational deadlines from event to system response
Examples: Video Recorder,
Comparing to non-real-time systems

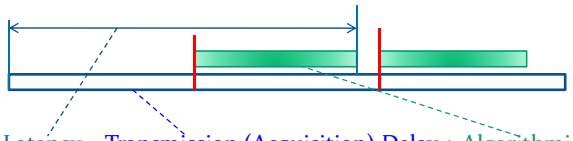
Digital Signal Processing Systems…

Real Time Systems…

Hard (Immediate) Real-time Systems
Correct execution of the main task depends on the duration of execution
Deadline Concept~ Real-Time Constraint (RTC)
Example: Car engine controller



Real-time → Waiting Time > 0



Latency = Transmission (Acquisition) Delay + Algorithmic Delay

Soft Real-time Systems
Completion after deadline is tolerated by losing QoS
Example: Dropping frames in video chat

Processors

History

First Commercial Microprocessor : 4-bit Intel-4004, 1971
→ 4-bit processor followed by 8, 16, 32, 64, … -bit
The most successful family, started with 16-bit Intel 8086
→ x86 / IA-32 (i386) *architecture* , 32-bit ones started with 80386

The "architecture" is the processor contents from the programmers vantage point

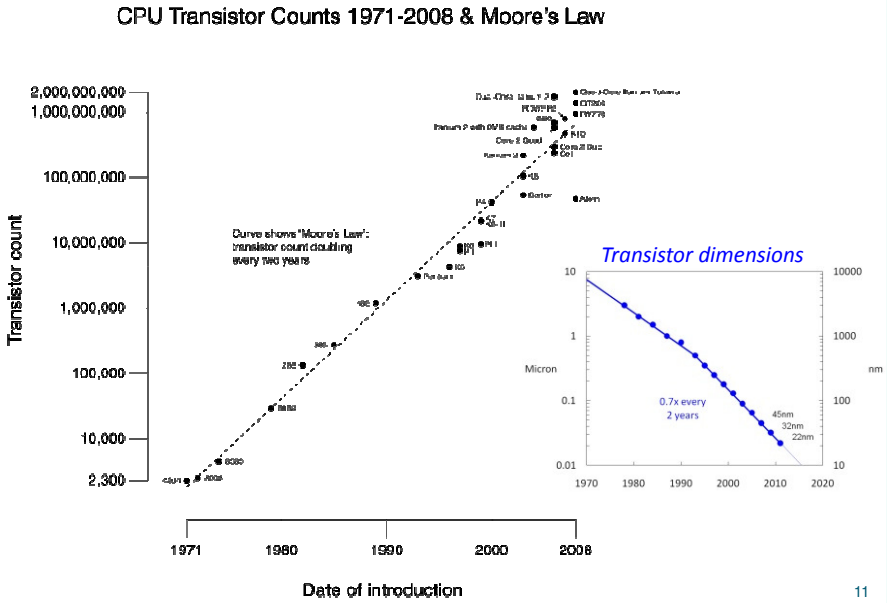
Moore's Law, 1965

The number of transistors that can be integrated on a single piece of silicon will double roughly every 18-24 months

Has held true for more than 45 years now may hold true for another decade

Roughly applies to both density and clock frequency → ***denser ~ faster***

Processors…



Processors…

Moore's Law…

Challenge of processor designers
→ Make Performance to follow at least (Moore's law)^2
density effect x clock effect
adding improvements thru innovations → micro-architectures, multi-core…

Performance has not gone up that fast! → *follows the Moore's law*
Orchestration problem *1971-2009* → $\times 2^{38/2} \sim \times 524{,}000$
Power consumption bottleneck
Heat dissipation → *not worth it to increase the clock*

Intel Rules:
Increasing clock rate by 25% will yield approximately 15% performance increase
But power consumption will be doubled

MIPS per Watts challenge → Change of view point

Processors…

Processor Design

Architecture (ISA)
programmer/compiler view
Functional structure, Interface to user/system programmer
Op-codes, Addressing modes, Registers, Number formats

Implementation (μArchitecture)
processor designer view
Logical structure, Pipelining
Functional units, Caches, Physical registers

Realization (Chip)
chip/system designer view
Physical structure for the implementation
Gates, Cells, Transistors, Interconnection

Processors…

Iron Law (Joel Emer)

$$\frac{Time}{Program} = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Time}{Cycle}$$

To be minimized

Architecture	Implementation	Realization
Code Size	CPI	Cycle time
Compiler Designer	Processor Designer	Chip designer

Instructions/Program → Instructions executed, not static code size
Determined by algorithm, compiler, ISA

Cycles/Instruction → Determined by ISA and CPU organization
Overlap among instructions reduces this term

Time/Cycle → Determined by technology, organization, clever circuit design

Processors…

Iron Law Examples

Processor A: clock 1ns, CPI 2.0, for program P (N instructions of Processor A)
Processor B: clock 2ns, CPI 1.2, for program P (N instructions of Processor B)

Time(A) = N x 2.0 x 1 = 2N
Time(B) = N x 1.2 x 2 = 2.4N

Time(B)/Time(A) = 2.4N/2N = 1.2 → A is 20% faster on program P

For performance of B to reach the performance of A:

- CPI(B) may be improved to 1
- Clock(B) may be changed to 1.66667ns
- ISA(B) may be redesigned to support golden instructions → 0.833N instructions to perform P

Processors…

Iron Law Examples…

Option:
stores can be executed in
1 cycle by slowing the clock down by 15%

Is it better to consider the modification?

oldCPI = 0.43 + 0.21 + 0.12 x 2 + 0.24 x 2 = 1.36
newCPI = 0.43 + 0.21 + 0.12 + 0.24 x 2 = 1.24

Speedup = oldtime/newtime
= {P x oldCPI x T}/{P x newCPI x 1.15 T}
= (1.36)/(1.24 x 1.15) = 0.95

→ Don't do it!

op	frequency in P	cycles
ALU	43%	1
Load	21%	1
Store	12%	2
Branch	24%	2

…	…	…
Store	12%	1

Processors…

Instruction Set Architecture (ISA)

The boundary between software and hardware

- Specifies the functional machine that is visible to the programmer
- Also, a functional spec for the processor designers

What needs to be specified by an ISA

- Operations : what to perform and what to perform next
- Temporary Operand Storage in the CPU : accumulator, stacks, registers
- Number of operands per instruction
- Operand location : where and how to specify the operands
- Type and size of operands
- Instruction-to-Binary Encoding

Processors…

Basic ISA Classification

- Stack Architecture (zero operand):
  - Operands popped from stack(s)
  - Result pushed on stack
- Accumulator (one-operand):
  - Special accumulator register is implicit operand
  - Other operand from memory
- Register-Memory (two-operand):
  - One operand from register, other from memory or register
  - Generally, one of the source operands is also the destination
  - A few architectures have allowed memory To memory operations
- Register-Register or Load/Store (three-operand):
  - All operands for ALU instructions must be registers
  - General format Rd ← Rs op Rt
  - Separate Load and Store instructions for memory access

Processors…

Important ISA Considerations

- Number of registers
- Data types/sizes
- Addressing modes
- Instructions complexity
- Branch/jump/function call
- Exception handling
- Instruction format/size/regularity

…

Data Type / Size

Fixed point → 8, 16, 24, 32,…-bit

Floating point → IEEE 754 Standard 32 and 64-bit

Processors…

Addressing Modes

- Register indirect: $M[Ri]$
- Indexed: $M[Ri+Rj]$
- Absolute: $M[\#n]$
- Memory indirect: $M[M[R_i]]$
- Auto-increment: $M[Ri]$; $Ri \mathrel{+}= d$
- Auto-decrement: $M[Ri]$; $Ri \mathrel{-}= d$
- Scaled: $M[Ri + \#n + Rj * d]$
- Update: $M[Ri = Ri + \#n]$

Immediate value: #n; Registers: Ri, Rj ; displacement Ri+#n; M :Memory block

Branches

Conditional/Unconditional

Function call is similar but needs parameter passing, saving state restoring state, Latency, …

Processors…

Modern ISAs

- Operations: simple ALU op's, data movement, control transfer
  Temporary Operand Storage in the CPU
- Large General Purpose Register (GPR) File
  Load/Store Architecture
- Three operands per ALU instruction (all registers)
  A ← B op C
- Addressing Modes
  Limited addressing modes, e.g. register indirect addressing only
- Type and size of operands
  32/64-bit integers, IEEE floats
- Instruction-to-Binary Encoding
  Fixed width, regular fields

- *Important Exception: Intel x86*

Processors…

MIPS ISA

The MIPS ISA was one of the first RISC instruction sets (1985)

Main characteristics:

- Load-Store Architecture
- Three operand format (Rd ← Rs op Rt)
- Simple instruction format
- 32 General Purpose Registers
- Only one addressing mode for memory operands: reg. indirect + displacement
- Limited, highly orthogonal instruction set: 52 instructions
- Simple branch/jump/subroutine call architecture

Processors…

x86 ISA, a CISC ISA

Was first introduced with Intel 8086 processor in 1978 → Evolved over the years

Main characteristics:

- Reg/Mem architecture →ALU instructions can have memory operands
- Two operand format → one source operand can be destination too
- Eight general purpose registers
- Seven memory addressing modes
- More than 500 instructions
- Instruction set is non-orthogonal
- Highly variable instruction size and format → instruction size varies from 1 to 17 bytes.

Processors…

ARM and Thumb ISAs

The ARM processor architecture provides support for:
The 32-bit ARM and 16-bit Thumb® Instruction Set Architectures
Reduced Instruction Set Computer (RISC) architecture

- Load/store architecture
- Simple addressing modes (determined from register contents and instruction)
- 16 /32-bit registers

- 8, 16, 32-bit data types
- Instructions that combine a shift with an arithmetic or logical operation
- Auto-increment and auto-decrement addressing modes to optimize loops
- Load and Store Multiple registers : instructions to maximize data throughput
- Conditional execution of almost all instructions to maximize execution throughput.
- ARM uses the Universal Assembly Language to provide a canonical form for all ARM and Thumb instructions

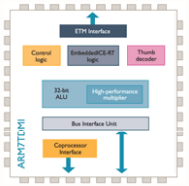
…

Processors…

ARM and Thumb ISAs…

Enhancements to a basic RISC architecture enable ARM processors to achieve a good balance of high performance, small code size, low power consumption, and small silicon area.

ARM6, ARM7, ARM9, ARM11, Cortex ,…



ARM7-TDMI

→ *3-stage pipeline*

→ *Von Newman bus structure (ARM9 Harvard)*

www.arm.com

→ *TDMI → Thumb-Multiplier-Debug (JTAG) Interface-ICE*

CPI ~ 1.9

20 billion chips created, 10 millions shipped everyday!
About 60 instructions

Processors…

Performance → Depends on What is Important!

Execution time, **throughput**, cost, area, power,…

Execution time or elapsed time is the time to finish a job

Throughput is completion counts per second or number of jobs done per sec

Faster CPUs or more CPUs to improve performance

Performance Metrics, MIPS, MFLOPS

MIPS = instruction count/(execution time x 10^6) = clock rate/(CPI x 10^6)

MIPS has serious shortcomings

MFLOPS = FP ops in program/(execution time x 10^6)
Assuming FP ops independent of compiler and ISA
However, not always safe:
→ Missing instructions (e.g. FP divide, sqrt/sin/cos), and Optimizing compilers

Relative MIPS and normalized MFLOPS, normalized to some common baseline machine

Processors…

Performance…

Program Selection

A Set of programs → Benchmarks

- ❑ Covering different aspects
- ❑ Tested on different processors

www.BDTi.com

Berkeley Design Technology Inc.

Example → DSP Kernel Benchmarks
Each kernel is implemented in hand-optimized assembly language on the target processor.

Video , OFDM,
DQPSK Benchmarks

The BDTI DSP Kernel Benchmarks™		
Benchmark	**Description**	**Example Application**
Real Block FIR	Finite impulse response filter that operates on a block of real (not complex) data.	Speech processing (e.g., G.728 speech coding).
Complex Block FIR	FIR filter that operates on a block of complex data.	Modem channel equalization.
Real Single-Sample FIR	FIR filter that operates on a single sample of real data.	Speech processing, general filtering.
LMS Adaptive FIR	Least-mean-square adaptive filter; operates on a single sample of real data.	Channel equalization, servo control, linear predictive coding.
IIR	Infinite impulse response filter that operates on a single sample of real data.	Audio processing, general filtering.
Vector Dot Product	Sum of the pointwise multiplication of two vectors.	Convolution, correlation, matrix multiplication, multi-dimensional signal processing.
Vector Add	Pointwise addition of two vectors, producing a third vector.	Graphics, combining audio signals or images.
Vector Maximum	Find the value and location of the maximum value in a vector.	Error control coding, algorithms using block floating-point.
Viterbi Decoder	Decode a block of bits that has been convolutionally encoded.	Error control coding.
Control	A sequence of control operations (test, branch, push, pop, and bit manipulation).	Virtually all DSP applications include some control code.
256-Point In-Place FFT	Fast Fourier Transform converts a time-domain signal to the frequency domain.	Radar, sonar, MPEG audio compression, spectral analysis.
Bit Unpack	Unpacks variable-length data from a bit stream.	Audio decompression, protocol handling.

Processors…

Going Deeper into Implementation

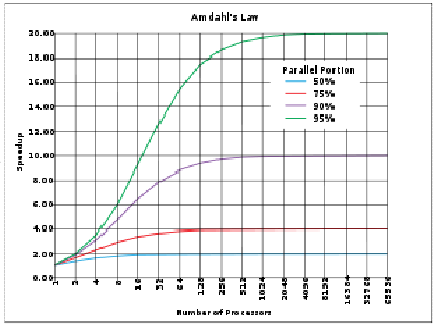
How to improve the implementation → Increasing CPI

Amdahl's Law (Gene Amdahl, IBM)

Expected speed-up of partial program improvement $speedup = \frac{1}{(1-p) + p/S}$

Expected speed-up of partial parallel implementation $speedup = \frac{1}{(1-p) + p/N}$

$$\%improvement = (1 - 1/speedup) \times 100$$



For parallel implementation:

$$speedup_{\max} = \lim_{N \to \infty} \frac{1}{(1-p) + p/N} = \frac{1}{1-p}$$

For sequential case:

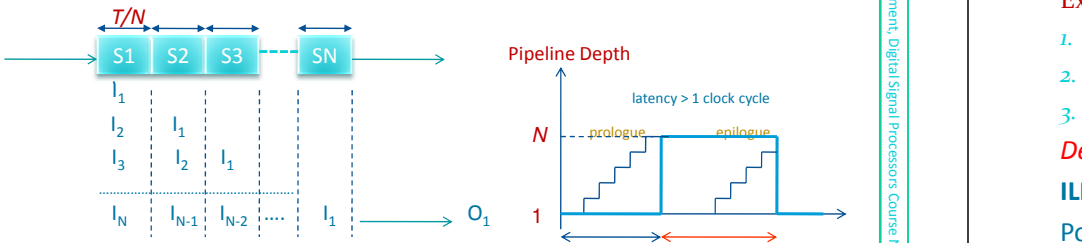
$$speedup_{\max} = \frac{S_{\max}}{(1-p)(S_{\max} - 1) + 1}$$

Processors…

Going Deeper into Implementation… Pipelining, 1980s

Overlap the execution of instructions

N Independent tasks being done in independent *N* modules → *N*-Stage Pipeline



Amdahl's Law

if *p* is the fully pipelined portion, for *K* In/Out:

$$speedup = \frac{KT}{(K+N-1)T/N} = \frac{1}{(1-p)+p/N} \rightarrow N \quad (K >> N)$$

$$p = (K-1)/K$$

Processors…

Going Deeper into Implementation…

Instruction Level Parallelism (ILP), 1990s

A measure of how many operations in a computer program can be performed simultaneously.

Example Program → ILP=3/2 [IPC (Instruction per Cycle) in CPU level]

1. $x = a + b$
2. $y = c - d$
3. $z = x \times f$

Design Problem → Compiler or Processor must increase ILP

ILP Processors

Possibility of having overlap among instructions

Pipelining is what can be done in a basic single block IPC_{max}= 1

Substantial improvement can be achieved by having multiple blocks, IPC>1

$$speedup = \frac{1}{(1-p)/S_1 + p/(NS_2)}$$

Processors…

Going Deeper into Implementation… ILP…

Single Issue Architecture → Multi Issue Architecture

Parallelism Expands in Space (1990's) → Using Multiple Basic Units

- ❑ VLIW (Very long Instruction Word) Processors

  Static / no code compatibility, recompile needed for different processors

- ❑ Superscalar Processors

  Dynamic/ code compatibility between processors family members

Dynamic Static Interface (DSI) → Gap between HW and SW

Placement of the DSI determines how the gap is bridged

Processors…

The Role of the Compiler

- ❑ Phases to manage complexity
- ❑ Parsing → intermediate representation
- ❑ Loop Optimizations
- ❑ Common Sub-Expression
- ❑ Procedure inlining
- ❑ Jump Optimization
- ❑ Register Allocation
- ❑ Code Generation → Assembly code + Problems with Optimization
- ❑ Constant Propagation
- ❑ Strength Reduction → Simpler equivalent code replacement
- ❑ Pipeline Scheduling

More important in VLIW Processors Case

Directing Compiler + Hand optimization are needed for full optimization

Digital Signal Processors

Microprocessors Optimized for Digital Signal Processing Algorithms

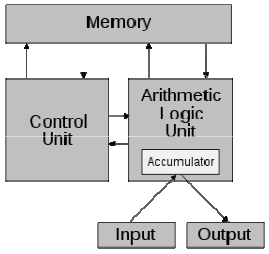
DSPs are shaped by DSP Algorithms

Let's Start with Von Neumann Architecture:

And implementing an FIR Filter

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

```
for(;;) {
    ReadNewSample(&xn);
    UpdateInputArray(xn);
    sum = 0;
    for (int k=0; k<N; k++)
        sum += h[k]*x[n-k];
    yn=sum;
    n++; }
```



Separate Memories/Paths for Code and Data

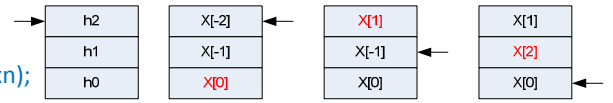
MAC

Faster Memories

Digital Signal Processors…

Modify the C code and see how to match a hardware

```
for(;;)
{
    ReadNewSample( &xn);
    UpdateInputArray(xn);
    sum = 0;
     a = &x[n-N+1];
     b = &h[N-1];
    for (int k=0; k<N; k++)
        sum += (*a++)*(*b++);
    yn=sum;
}
```



$$y0 = h[0]*x[0] + h[1]*x[-1] + h[2]*x[-2]$$
$$y1 = h[0]*x[1] + h[1]*x[0] + h[2]*x[-1]$$
$$y2 = h[0]*x[2] + h[1]*x[1] + h[2]*x[0]$$

Same/Similar ideas for efficiently perform in FFT, Convolution, IIR filtering

→ inner (dot) product