EDISP (NWL2)
(English) Digital Signal Processing
Implementing signal processing

December 7, 2016

Scalar product

$$\mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

- Convolution
- FIR filter (direct)
- IIR filter (direct - use for low orders only!!)

$$w(n) = x(n) - \langle a[k], w[n-k] \rangle$$

$$y(n) = b_0 w(n) + \langle b[k], w[n-k] \rangle$$

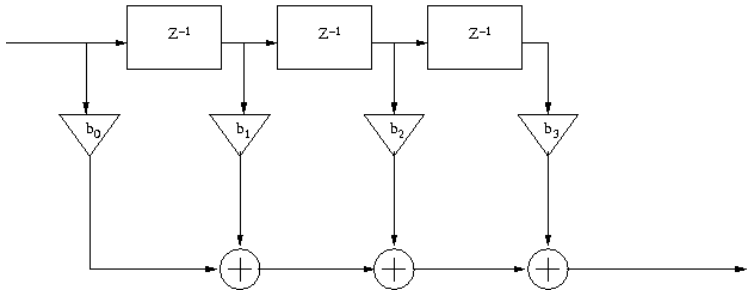
- matrix multiply $y = a \times b \longrightarrow y_{km} = \langle a_{k,:}, b_{:,m} \rangle$
- We need to repeat (length times)
  - fetch a
  - fetch b
  - multiply and accumulate
- Do it in parallel or sequentially?

Arithmetics

- Floating point problems:
  - Speed
  - Conversion from A/D native format
- Fixed point problems:
  - (overflows) 16 bits + 16 bits= 17 bits

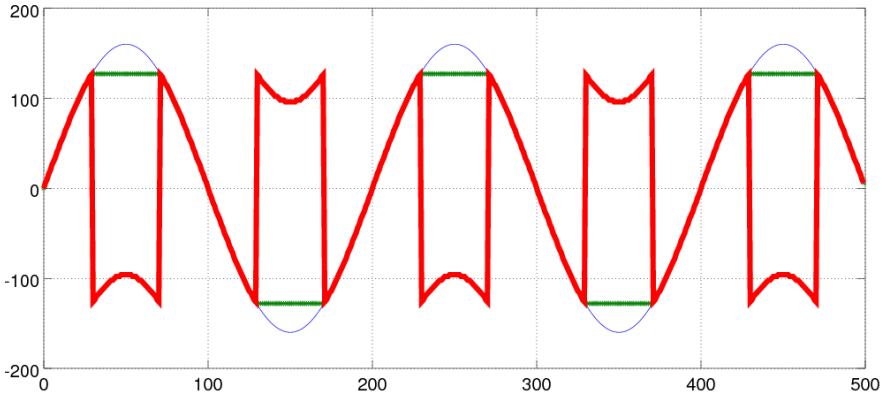
  $\longrightarrow$ partial solution with saturation arithmetic
  - (rounding) 16 bits * 16 bits= 32 bits

  $\longrightarrow$ this is hidden with FP, but it can also hurt there!



Saturation arithmetic

- blue – original
- red – 2's complement wrapped at 8 bits
- green – 2's complement saturated at 8 bits

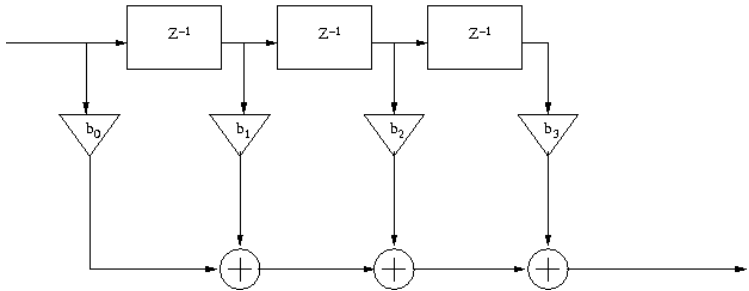


Find the error power in both modes.

Dynamic range

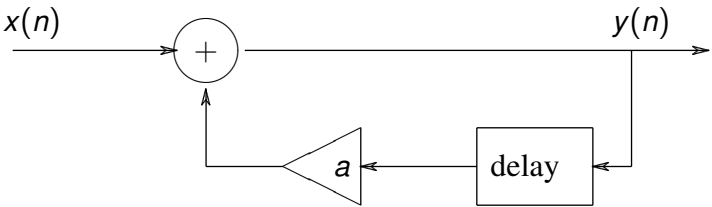
- Strongest signal: “rail to rail” (magnitude: 2^{N-1})
- Weakest signal: “one LSB” (magnitude: 1)
- Noise may hide the weak signal

Noise (at the output) may come also from rounding **inside** the system.



Resonance behavior

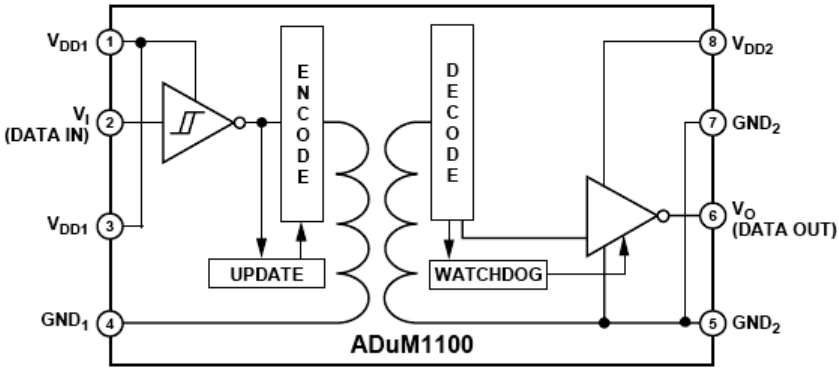
Maximum of $y(n) = x(n) + 0.9y(n-1)$?
Let $y(n) = y(n-1) = 1$ Then $x(n) = 0.1 \longrightarrow$ amplified 10x!



What happens if you scale it down?

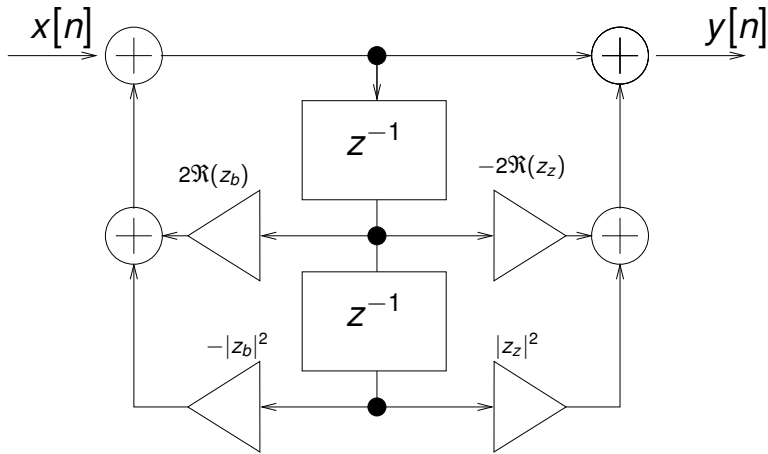
Serial I/O

Galvanic isolation:

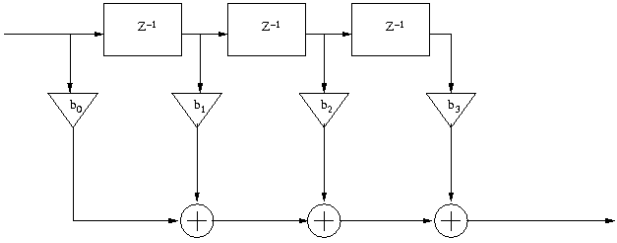


Implementation tricks

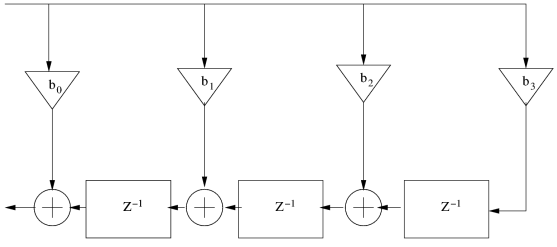
- Multiplication by 2^N – by shift or add
- Save on storage – inverted IIR structure
- Cascade biquads
- Use CIC filters



Implementation tricks – speed



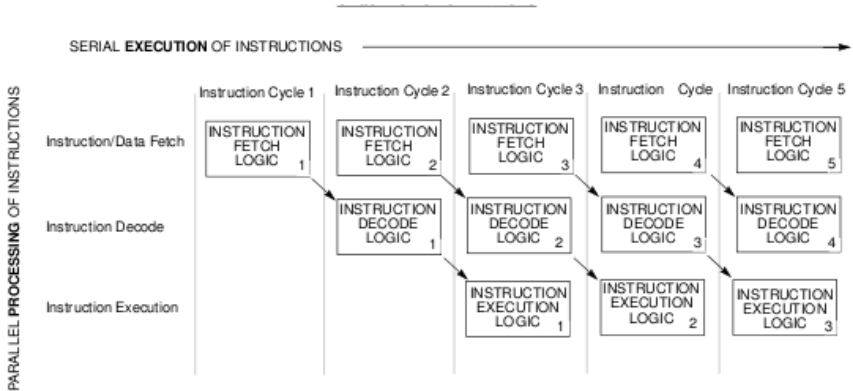
Cascaded adders – long propagation time



Digital Signal Processors (DSP)

- Specialization for scalar product (and FFT)
- Single-cycle processing (memory throughput):
  - parallelism (pipelining)
  - Harvard architecture (program (P), data (X), data (Y) memories)
- Desing for embedding – I/O & host interfaces etc.
- Special arithmetic modes: rounding, saturation
- ALU in fraction mode
- ALU overflow space
- butterfly implementation (+ and -)
- Special addressing modes:
  - circular buffer,
  - bit-reversal,
  - matrix address
- Hardware speedups
  - Hardware loop
  - Hardware stack
  - Fast interrupts
  - Single-cycle instructions
  - RISC..

Instruction pipeline

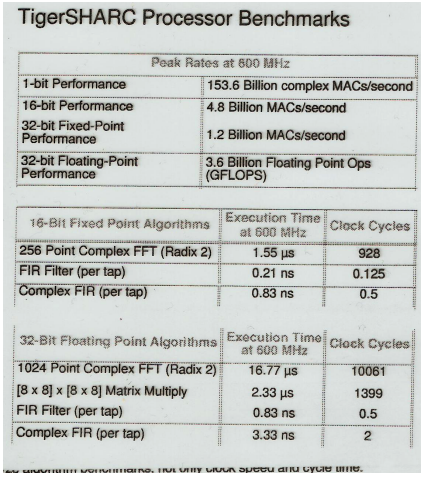


Manufacturers

- Texas Instruments (TMS family)
  - From simple to multiprocessor
  - fixed point and floating point
- Analog Devices (ADSP family, codename “SHARC” etc.)
- Motorola $\longrightarrow$ now Freescale Semiconductors (DSP56K family)
- vector units of general-purpose μP
- graphic processors: IBM Cell BE, Nvidia CUDA

Also “DSP cores” - VHDL or silicon IP blocks to be embedded into VLSIs.

TS benchmarks



TigerSHARC Processor Benchmarks

Peak Rates at 600 MHz	
1-bit Performance	153.6 Billion complex MACs/second
16-bit Performance	4.8 Billion MACs/second
32-bit Fixed-Point Performance	1.2 Billion MACs/second
32-bit Floating-Point Performance	3.6 Billion Floating Point Ops (GFLOPS)

16-Bit Fixed Point Algorithms	Execution Time at 600 MHz	Clock Cycles
256 Point Complex FFT (Radix 2)	1.55 µs	928
FIR Filter (per tap)	0.21 ns	0.125
Complex FIR (per tap)	0.83 ns	0.5

32-Bit Floating Point Algorithms	Execution Time at 600 MHz	Clock Cycles
1024 Point Complex FFT (Radix 2)	16.77 µs	10061
[8 x 8] x [8 x 8] Matrix Multiply	2.33 µs	1399
FIR Filter (per tap)	0.83 ns	0.5
Complex FIR (per tap)	3.33 ns	2

algorithm benchmarks. Not only clock speed and cycle time.

Speedups

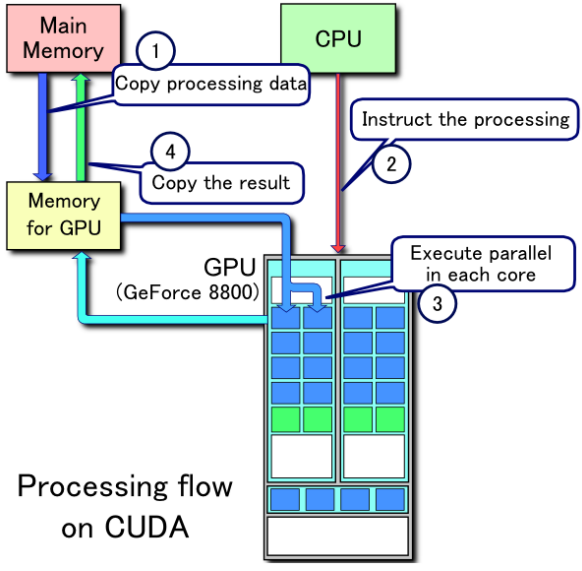
- SIMD mode: the same operation on different data (e.g. different rows of a matrix): multiple ALU's
- Multiprocessing (with separate RAMs)
- DMA I/O

Alternatives: GPGPU

- massive multicore, massive pipelining, SIMT
- stream processing: applying the same operation (“kernel”) to each element of the stream of data (vector element or vector fragment)
  a `for (i=0;i++;i<N)` loop where each *i*-th loop content is executed in parallel
- linear algebra libraries (CUBLAS)

Main usage: speeding up offline calculations, simulations etc., with `main()` running on a PC
For realtime work in embedded environment – DSP or FPGA is more predictable!

GPU CUDA

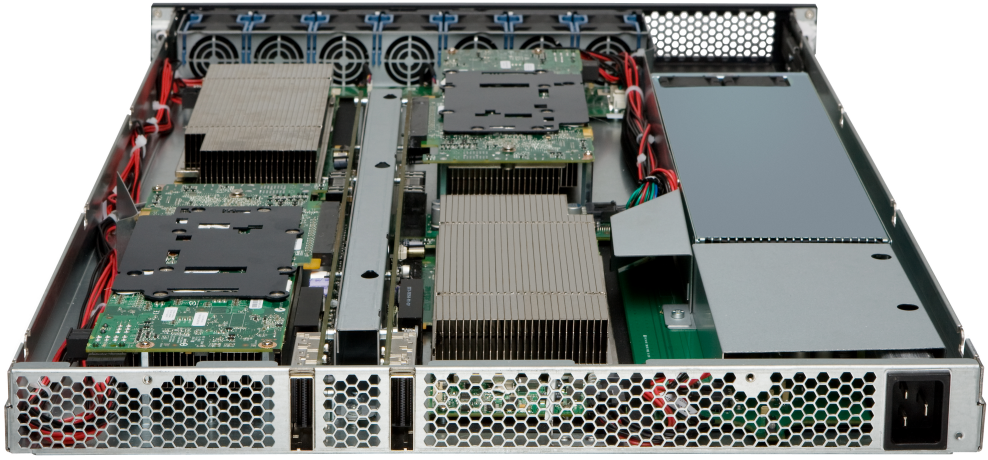

Main Memory
CPU
1
Copy processing data
Instruct the processing
2
4
Copy the result
Memory for GPU
GPU
(GeForce 8800)
Execute parallel in each core
3
Processing flow on CUDA

GPU CUDA

Technical Specifications

Product	Tesla S1070
# of Tesla Processors	4
# of Computing Cores	960 (240 per processor)
Floating Point Precision	IEEE 754 single & double
Total Dedicated Memory	16 GB (organized as 4.0 GB per GPU)
Memory Interface	4x 512-bit GDDR3 memory interface (organized as a 512-bit interface per GPU)
Memory Bandwidth	408 GB/sec (102 GB/s per GPU to local memory)
Typical Power Consumption	700 W
System Interface	PCIe x16 or x8
Programming environment	CUDA

GPU CUDA

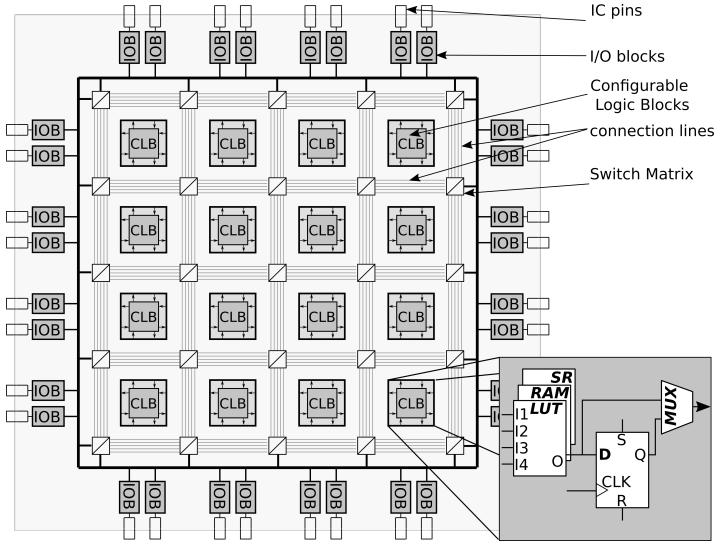


FPGA

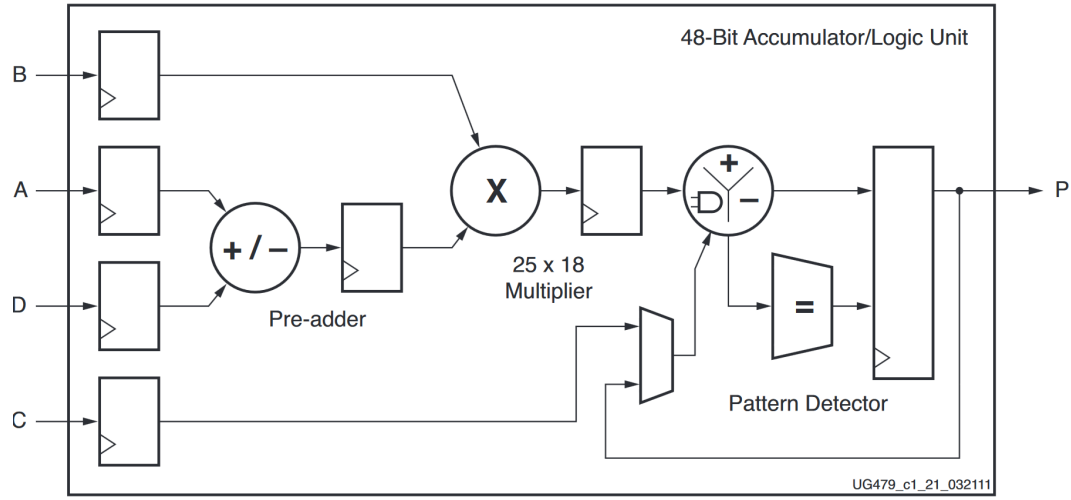
Field Programmable Gate Array

- an array of logic blocks (simple or complex)
- configuration and connections stored in RAM (sometimes ROM)
- Additionaly – specialized processing blocks (e.g. 48-bit ALU)
- Sometimes – an option to convert to mask-programmable version
- Complexity from thousands to millions of gates
- Cost from few $ to few k$

FPGA


IC pins
I/O blocks
Configurable
Logic Blocks
connection lines
Switch Matrix
IOB
CLB
SR
RAM
LUT
I1
I2
I3
I4
O
MUX
S
D
Q
CLK
R

FPGA – DSP48E1

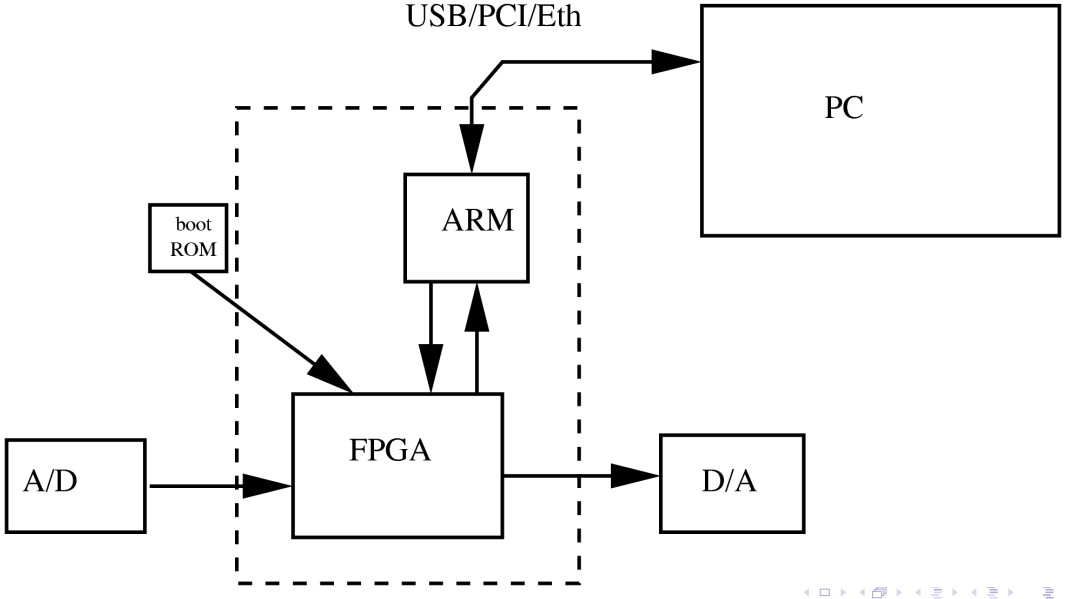


FPGA – programming

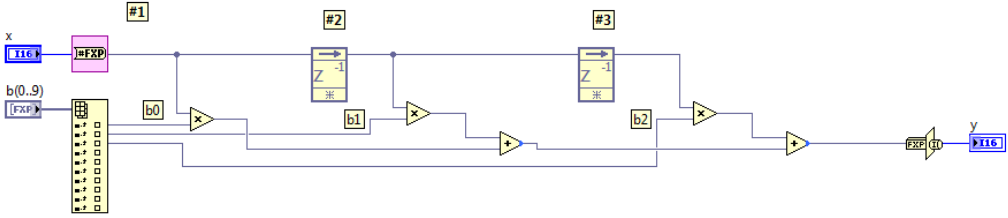
FPGA programming

- graphical (draw the schematic)
- VHDL (usually it is an intermediate form)

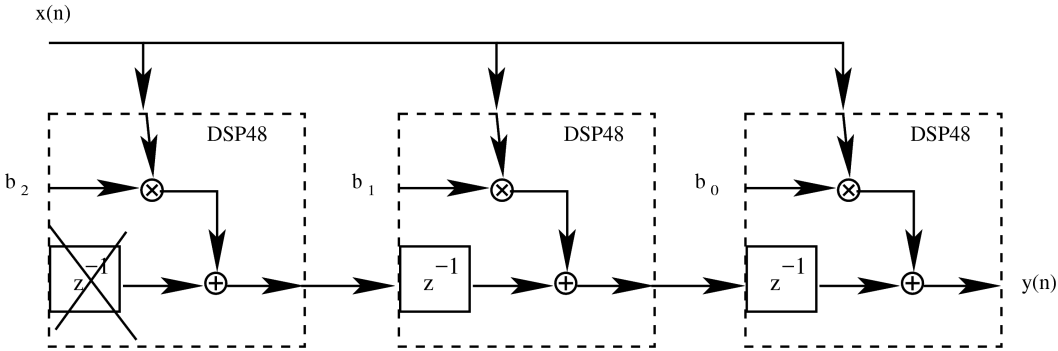
FPGA – environment



Signal processing with an FPGA



Signal processing with an FPGA – DSP48



FPGA – detailed programming of DSP48