

VLSI Programming: Lecture 1

Course 2IN35

Course: Kees van Berkel c.h.v.berkel@tue.nl

Rudolf Mak r.h.mak@tue.nl

Lab: Kees van Berkel, Rudolf Mak
Wei Tong, Hrishikesh Salunkhe

www: <http://www.win.tue.nl/~cberkel/2IN35/>

Lecture 1 Introduction

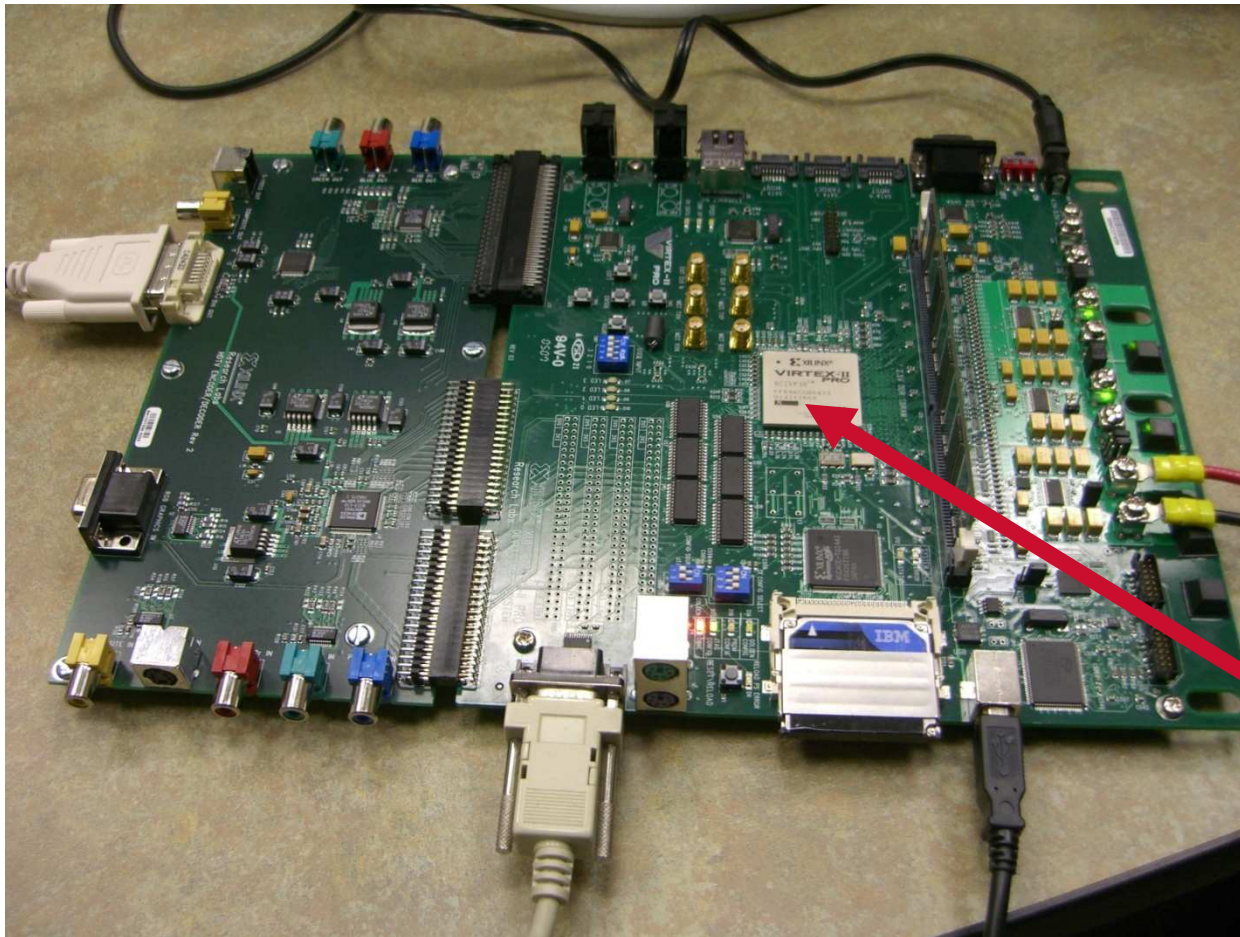
Introduction to VLSI Programming: goals

- to acquire insight in the description, design, and optimization of fine-grained parallel computations;
- to acquire insight in the (future) capabilities of VLSI as an implementation medium of parallel computations;
- to acquire skills in the design of parallel computations and in their implementation on FPGAs.
-

Contents

- Massive parallelism is needed to exploit the huge and still increasing computational capabilities of Very Large Scale Integrated (VLSI) circuits:
- we focus on fine-grained parallelism (not on networks of computers);
- we assume that parallelism is by design (not by compilation);
- we draw inspiration from consumer applications, such as digital TV, 3D TV, image processing, mobile phones, etc.;
- we will use Field Programmable Arrays (FPGA) as fine-grained abstraction of VLSI for practical implementation.

FPGA IC on the Xilinx XUP Board



FPGA



XC2VP30

Lab work prerequisites

- Notebook
- Exceed (can be obtained through the software distribution of the university)
- Access to UNIX server Dept. W&I (can be obtained through BCF, HG floor 8)
- Lab work is by teams of two students.
- Have FPGA tools (SW) installed on your machine by Feb 28

VLSI Programming: time table 2012

date	class lab	subject (provisional)
Feb 7	2 0 hours	introduction, DSP representations, bounds
Feb 14	2 0 hours	pipelining, retiming, transposition, J-slow, unfolding
Feb 21		no lecture (carnaval) (have FPGA tools installed)
Feb 28	2 0 hours	unfolding (cntd), look-ahead, strength reduction
Mar 6	1 3 hours	Introductions FPGA, Verilog, Lab (A1)
Mar 13	2 2 hours	systolic computation (A1)
Mar 20	1 3 hours	folding; FPGAs: pipelining, retiming (A2)
Mar 27	1 3 hours	DSP processors; FPGAs: parallelism, strength reduc. (A3)
Apr 3	1 3 hours	FPGAs: sample-rate conversion (A4)
Apr 10,17		2x no lecture (examinations)
Apr 24	1 3 hours	FPGAs: video scaler (A5); hand-in report A3
May 1	1 3 hours	FPGAs: video scaler (A5 cntd)
May 22		deadline final report A4, A5

Course grading (provisional)

Your course grade is based on:

- the quality of your programs/designs [30%];
- your final report on the design and evaluation of these programs (guidelines will follow) [30%];
- a concluding discussion with you on the programs, the report and the lecture notes [20%];
- intermediate assignments [20%].
- Credits: 5 points = based on 140 hours on your side

Note on course literature

First half of VLSI programming course is based on:

- Keshab K. Parhi. *VLSI Digital Signal Processing Systems, Design and Implementation*. Wiley Inter-Science 1999.
- This book is recommended.

Accompanying slides can be found on:

- <http://www.ece.umn.edu/users/parhi/slides.html>
- <http://www.win.tue.nl/~cberkel/2IN35/>

Mandatory reading:

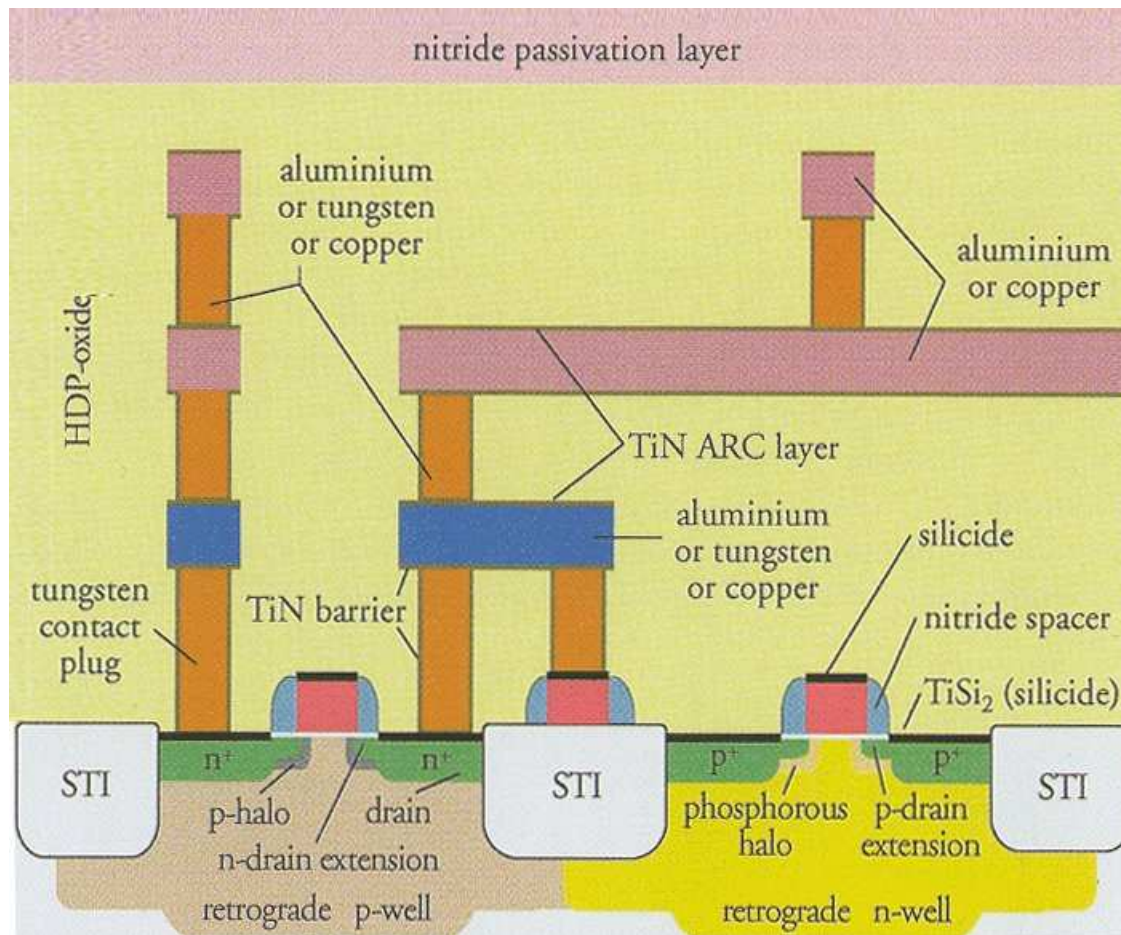
- Keshab K. Parhi. *High-Level Algorithm and Architecture Transformations for DSP Synthesis*. Journal of VLSI Signal Processing, 9, 121–143 (1995), Kluwer Academic Publishers.

Introduction

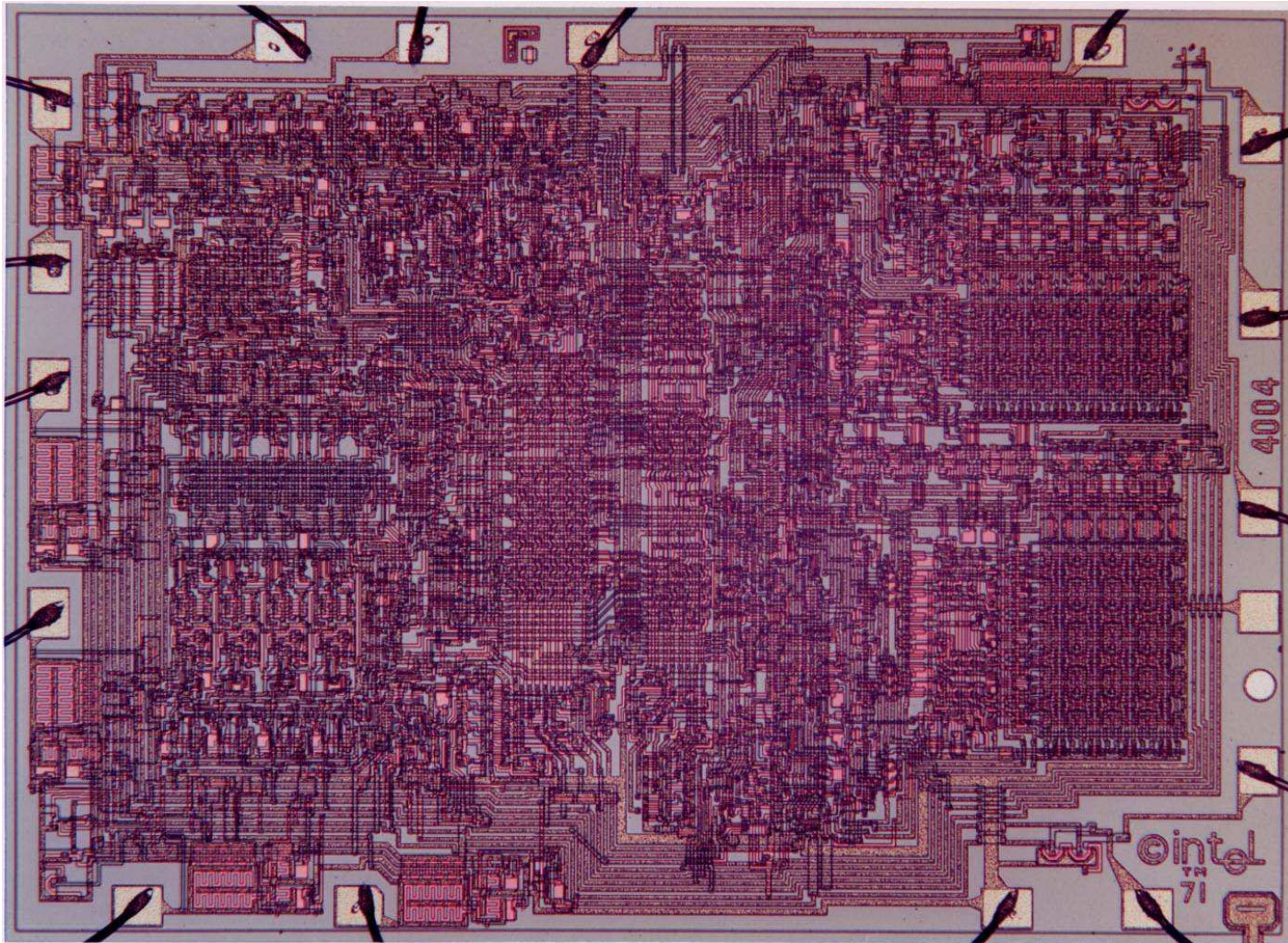
- Some inspiration from the technology side
 - VLSI
 - FPGAs
- Some inspiration from the application side
 - Machine Intelligence
 - Bee, SKA, SETI
 - Digital Signal Processing (Software Defined Radio)
- Parhi, Chapters 1, 2
 - DSP Representation Methods
 - Iteration bounds

Some inspiration from the technology side

Vertical cut through VLSI circuit

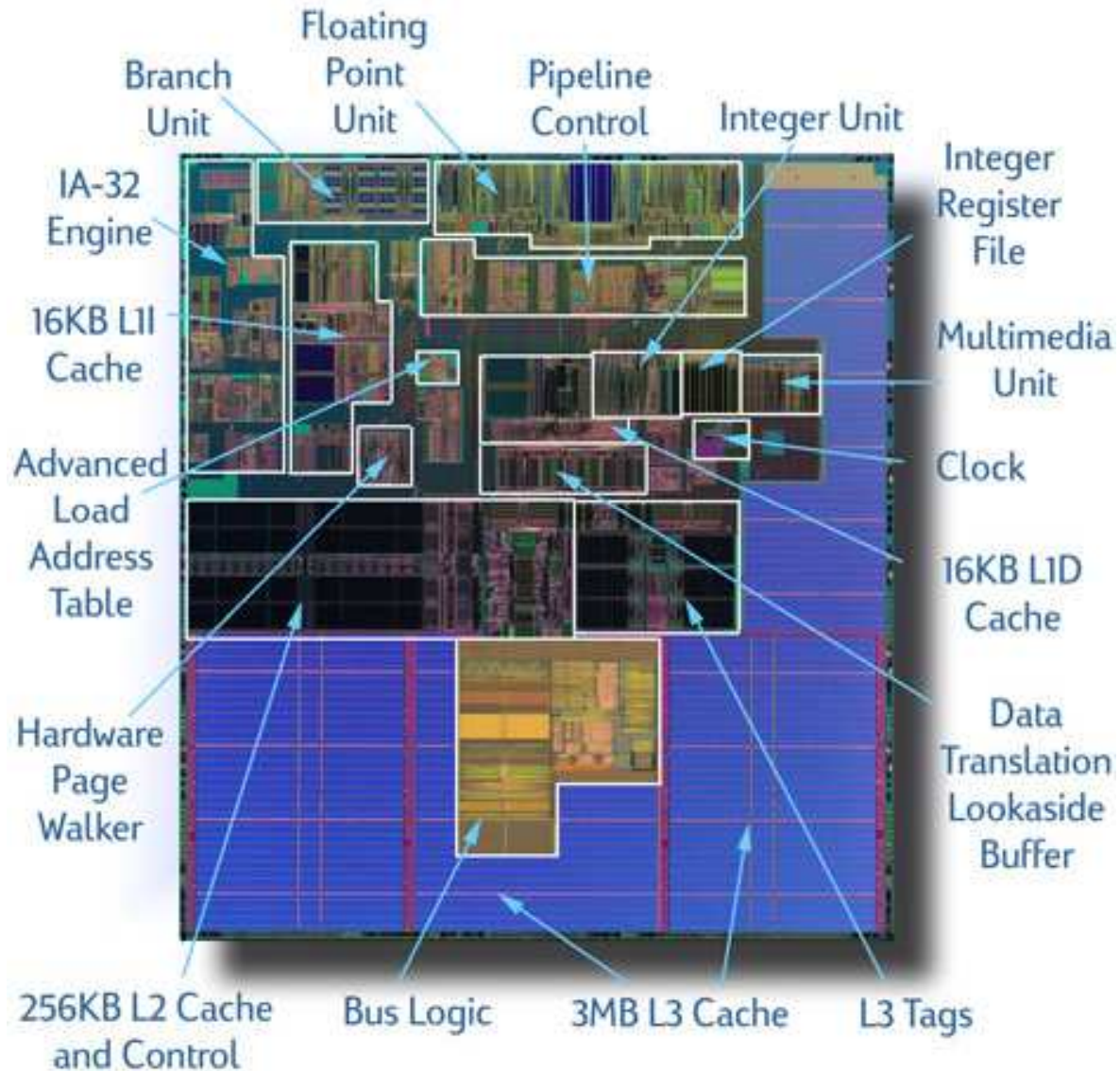


Intel 4004 processor [1970]



- 1970
- 4-bit
- 2300 transistors

Intel Itanium 2 6M Processor [2004]

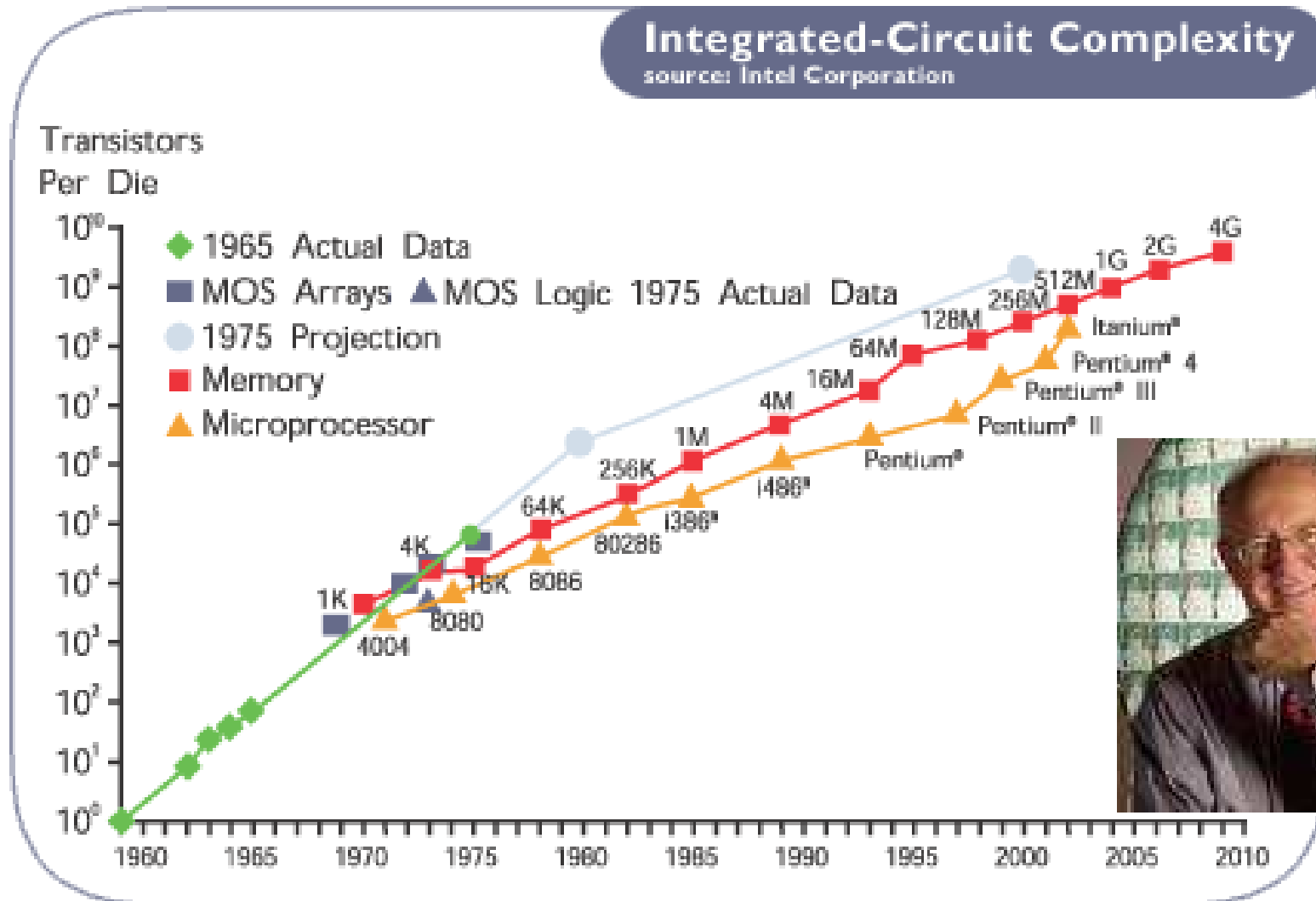


Intel Itanium 2 6M Processor [2004]

- Power dissipation 130 W (107 W typical)
- 3 Levels of Cache: 16k + 16k, 256K, 6M
- CMOS technology: 130nm
- Clock frequency 1.5 GHz
- 7,877 pins, 95 percent of the are for power
- 1,322 Specint base2000 for a single processor!
- Next generation: 1.7GHz and 9MByte cache!

- Rusu et. al [Intel], Itanium 2 Processor 6M: Higher Frequency and Larger L3 Cache, IEEE Micro April 2004, vol. 24, Issue 2, pp. 10–18.

Moore's Law

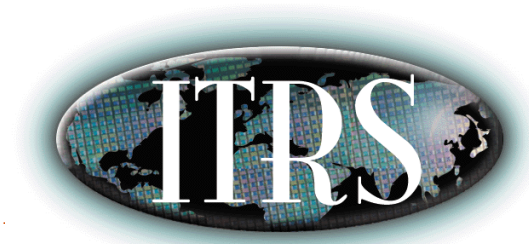


Rule of two [Hu, 1993]

- Every 2 generations of IC technology (6 years)
- device feature size 0.5 x
- chip size 2 x
- clock frequency 2 x (no longer true)
- number of i/o pins 2 x
- DRAM capacity 16 x
- logic-gate density 4 x

ITRS: INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS

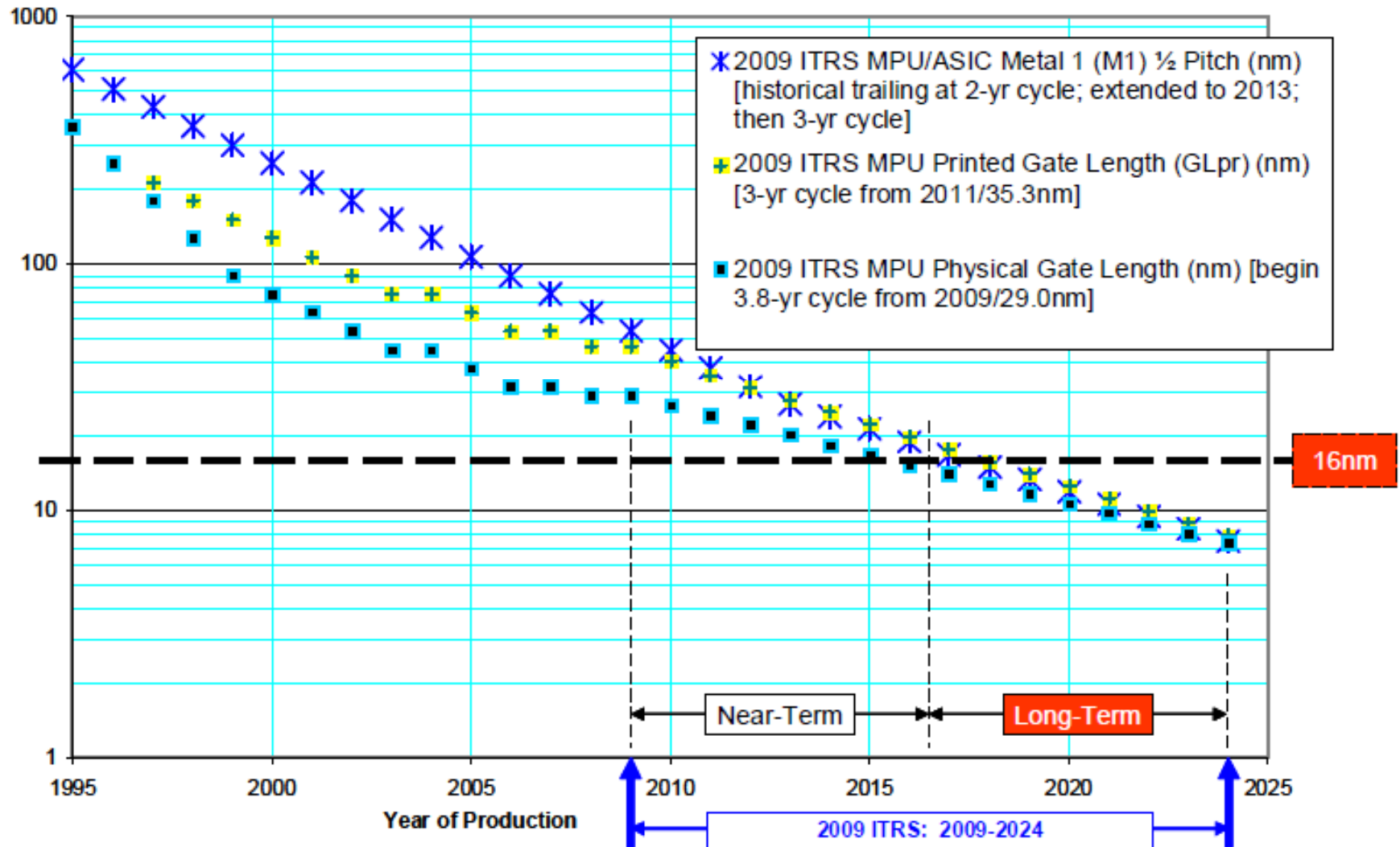
- The overall objective of the ITRS is to present industry-wide consensus on the “best current estimate” of the industry’s research and development needs out to a 15-year horizon.
- As such, it provides a guide to the efforts of companies, universities, governments, and other research providers/funders.
- The ITRS has improved the quality of R&D investment decisions made at all levels and has helped channel research efforts to areas that most need research breakthroughs.
- Involves over 1000 technical experts, world wide.
- a self-fulfilling prophecy? ... or wishful thinking?



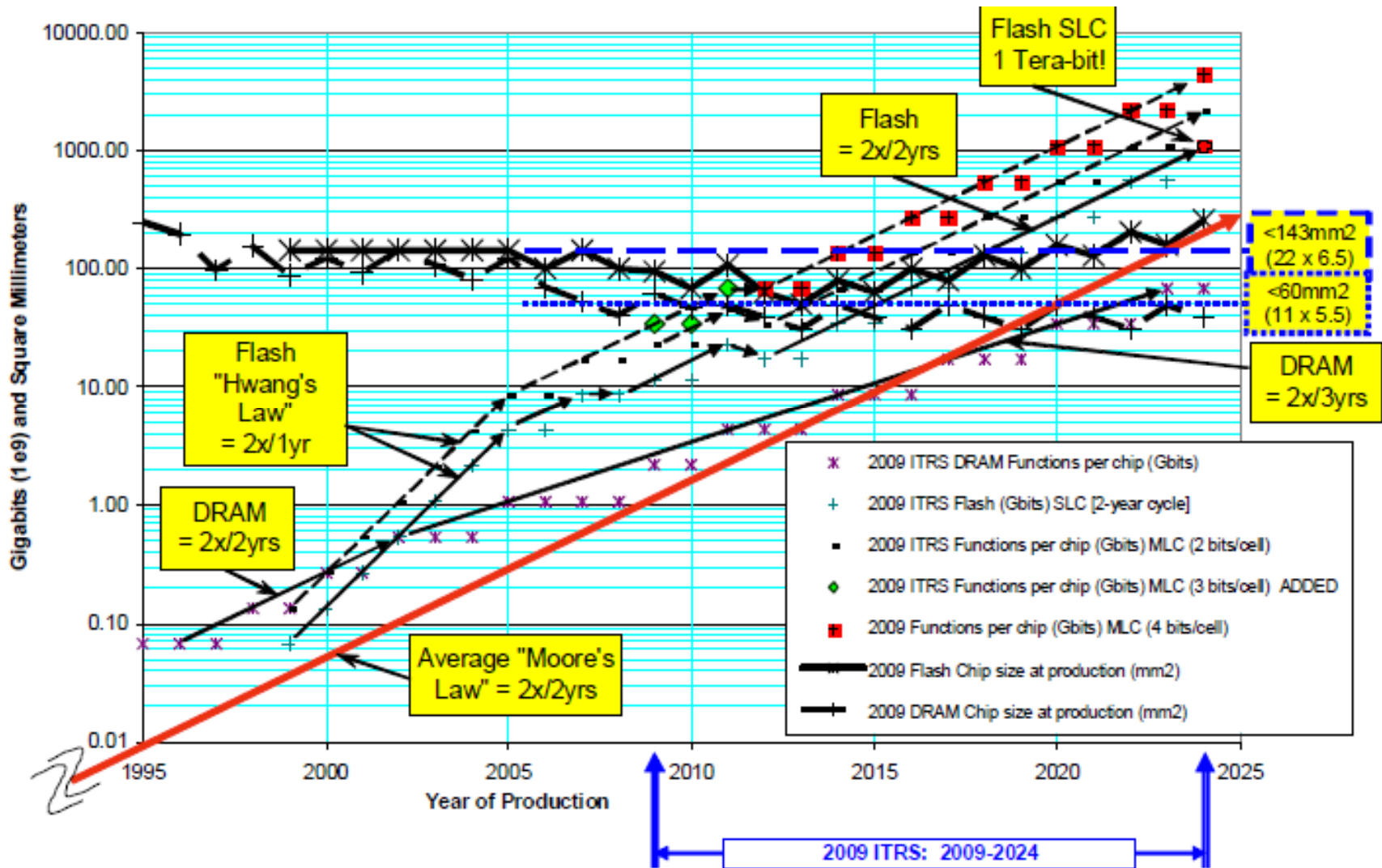
2/7/2012



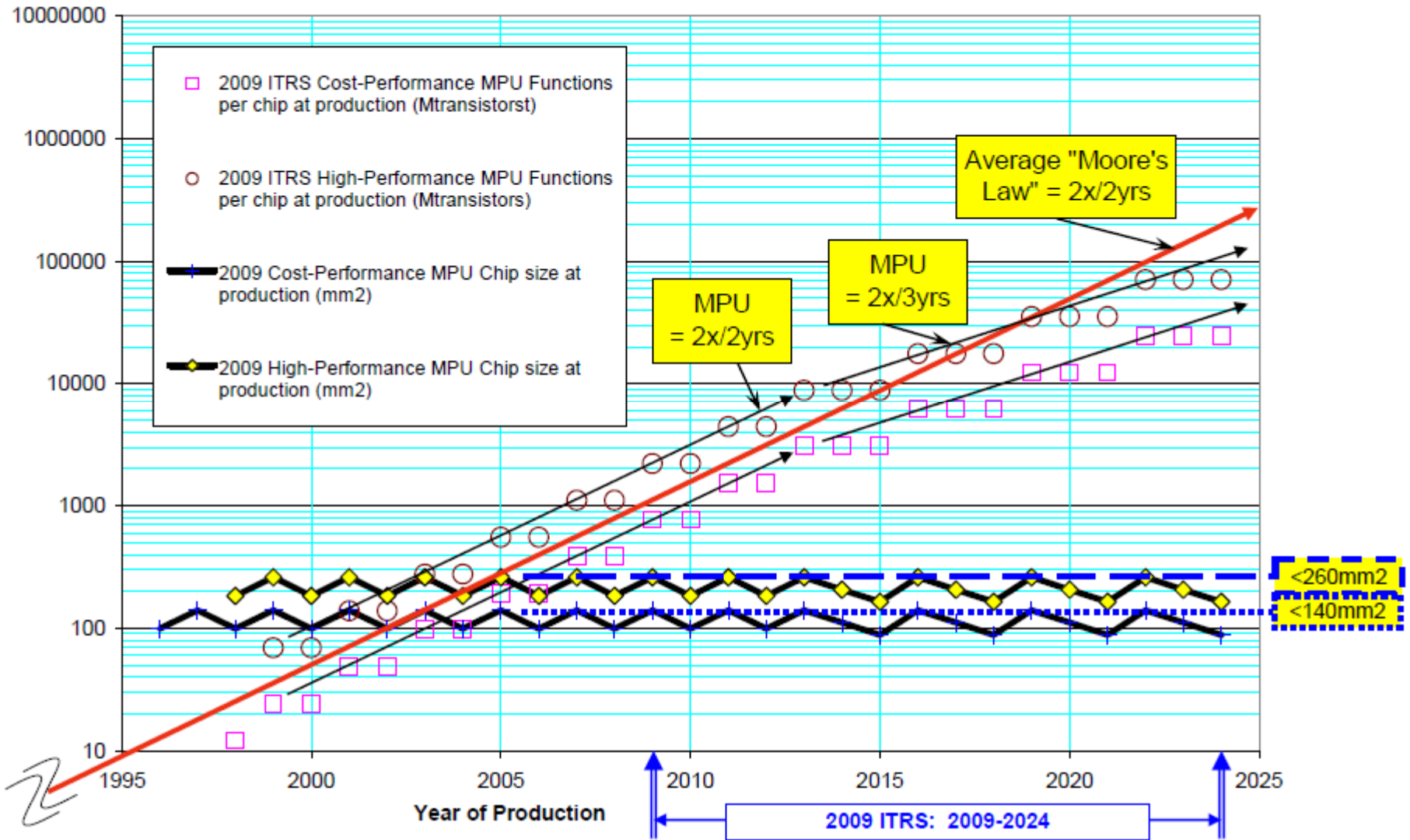
2009 ITRS—MPU/high-performance ASIC Half Pitch and Gate Length Trends



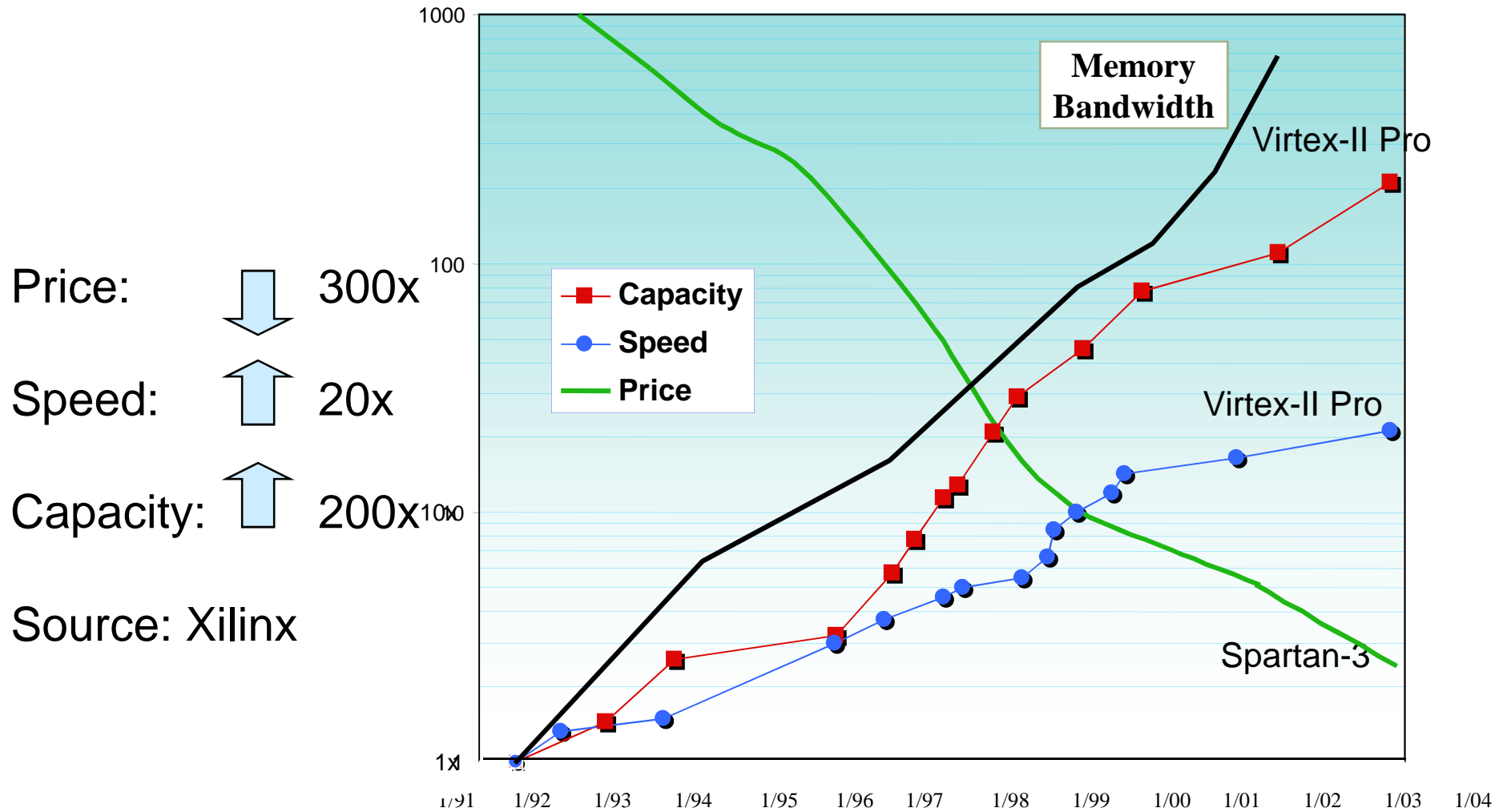
2009 ITRS— Functions/chip and chip size



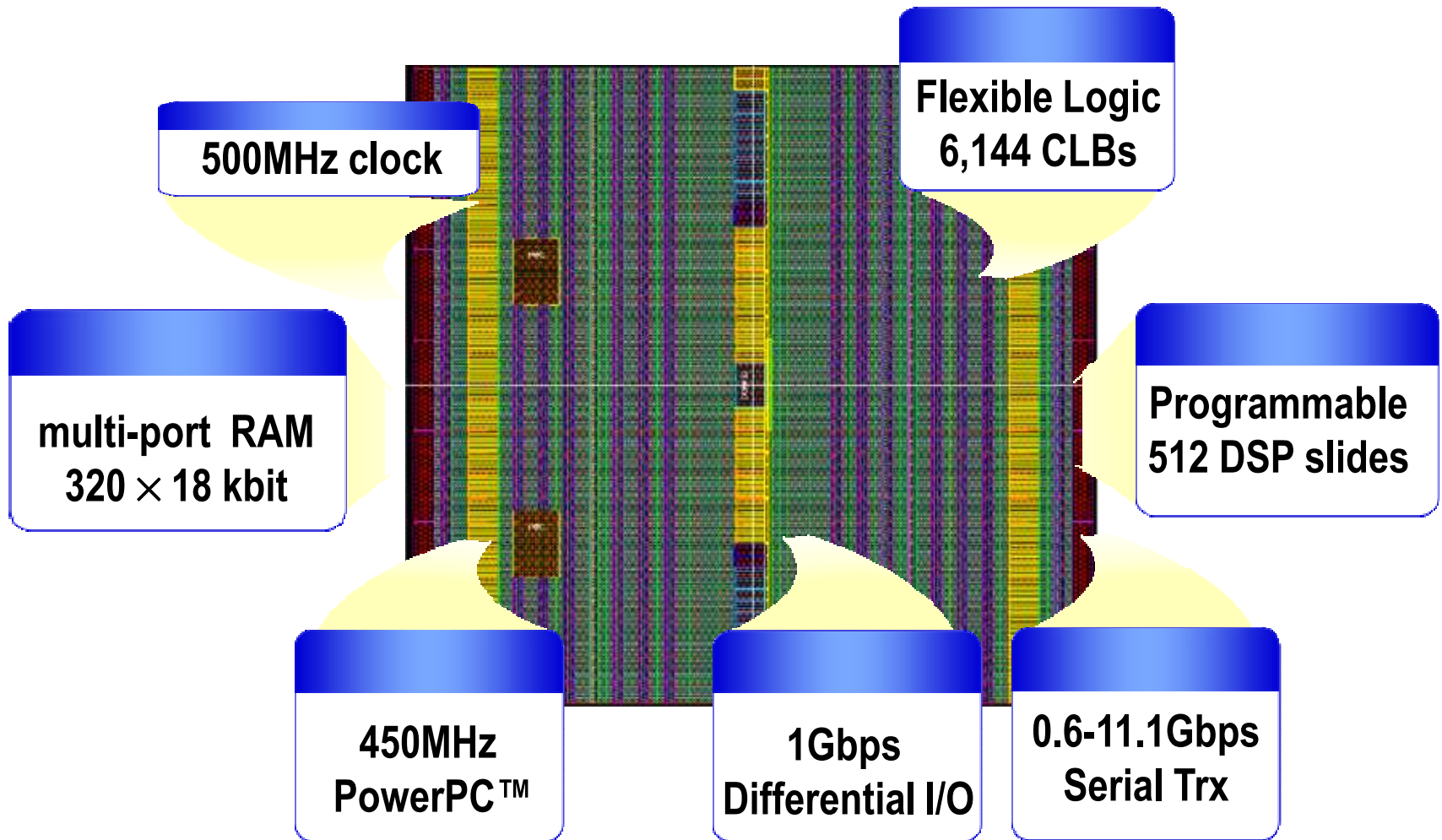
Transistor counts [M]/ Micro Processor Unit



FPGA, the last 10 years



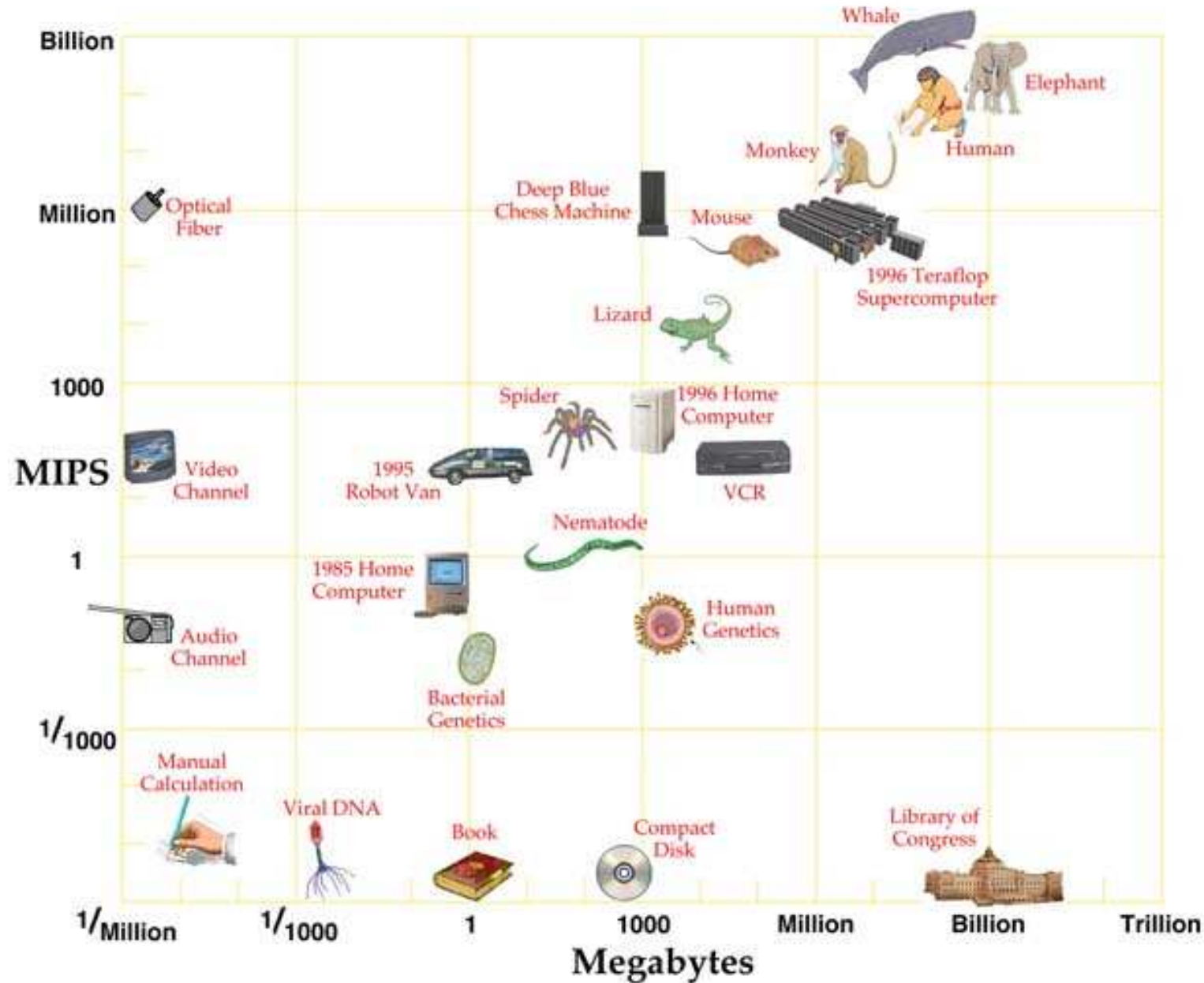
Virtex 4 FPGA: 4VSX55



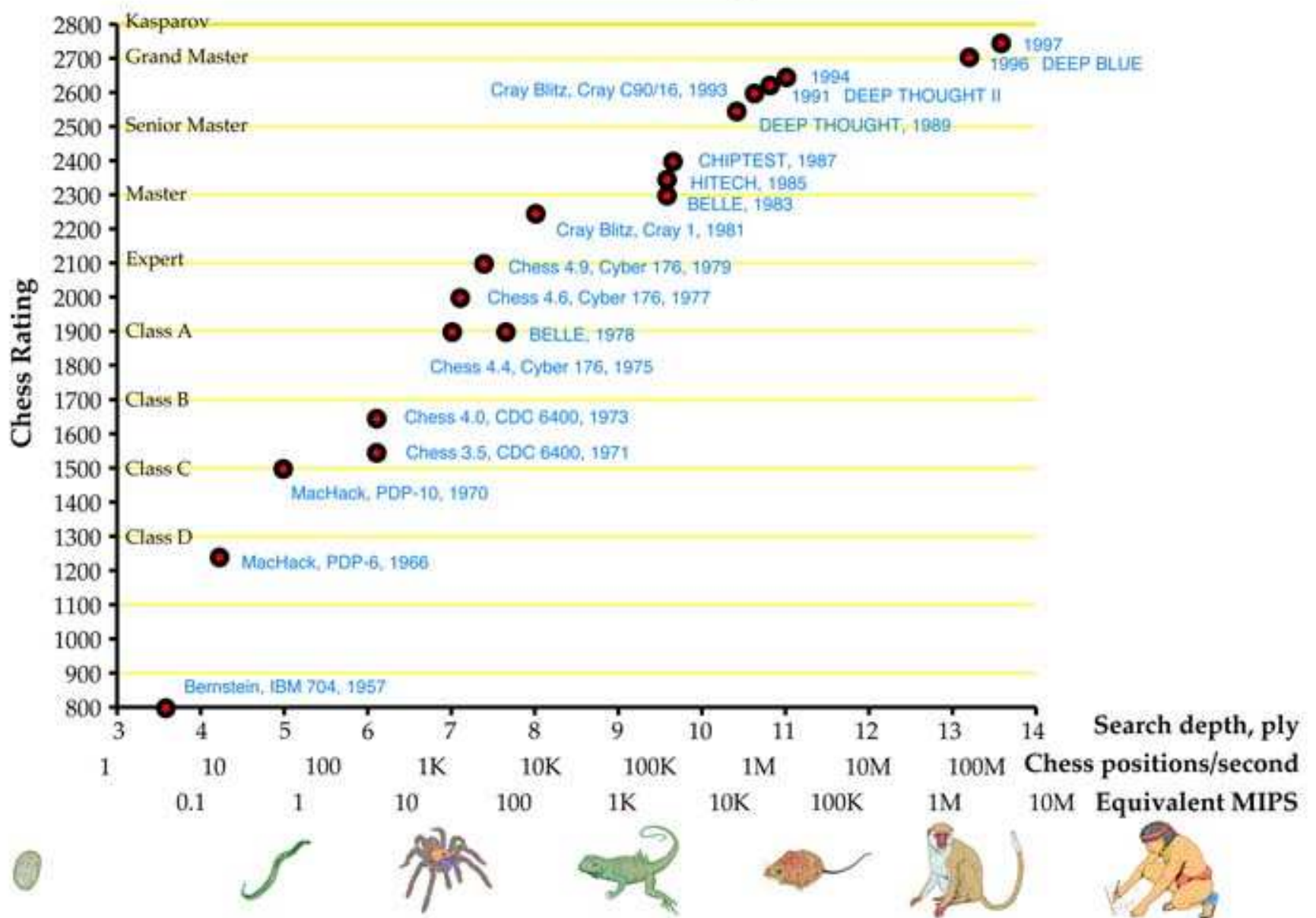
Some inspiration from the application side

All things grand and small [Moravec '98]

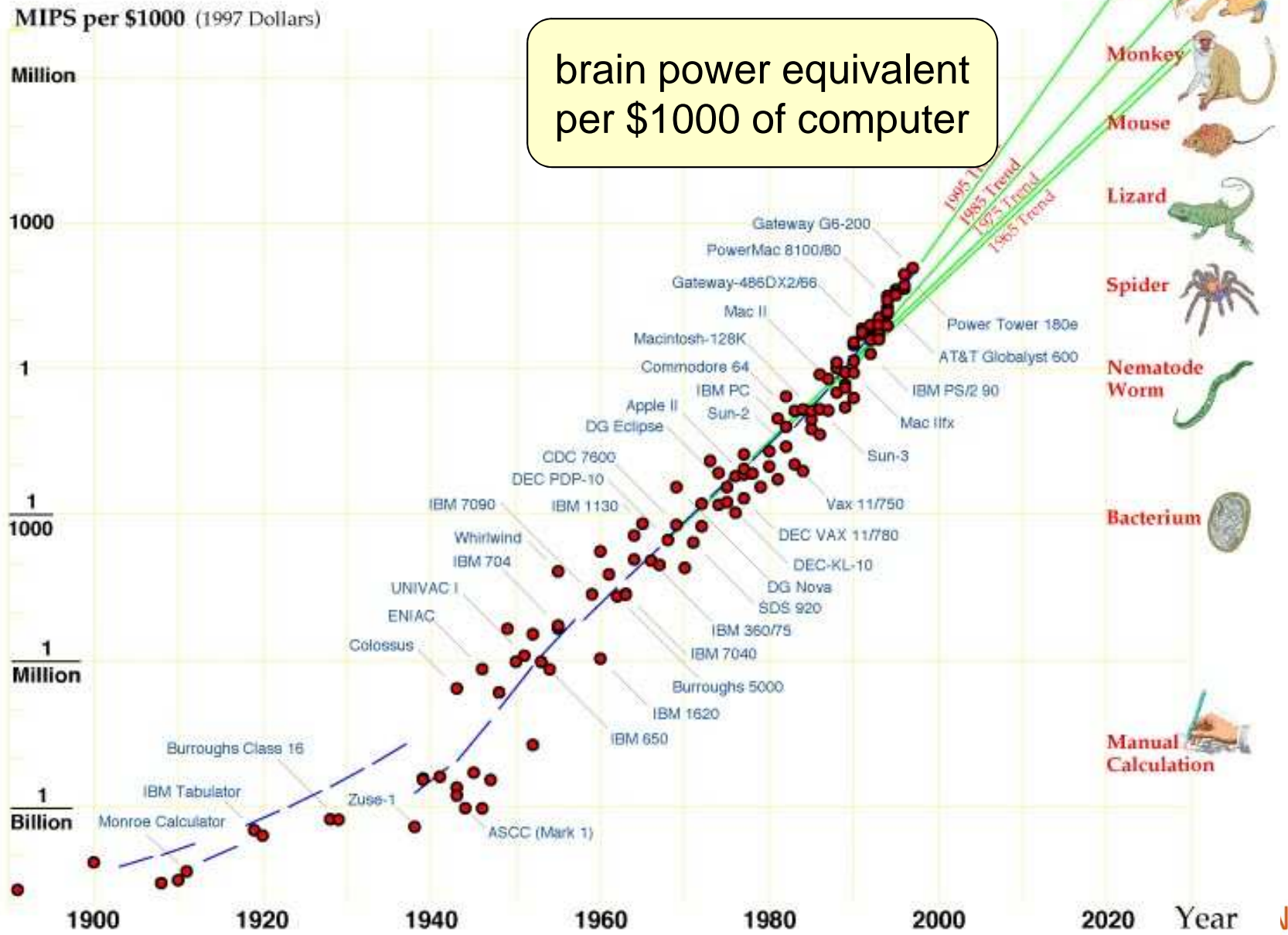
All Things, Great and Small



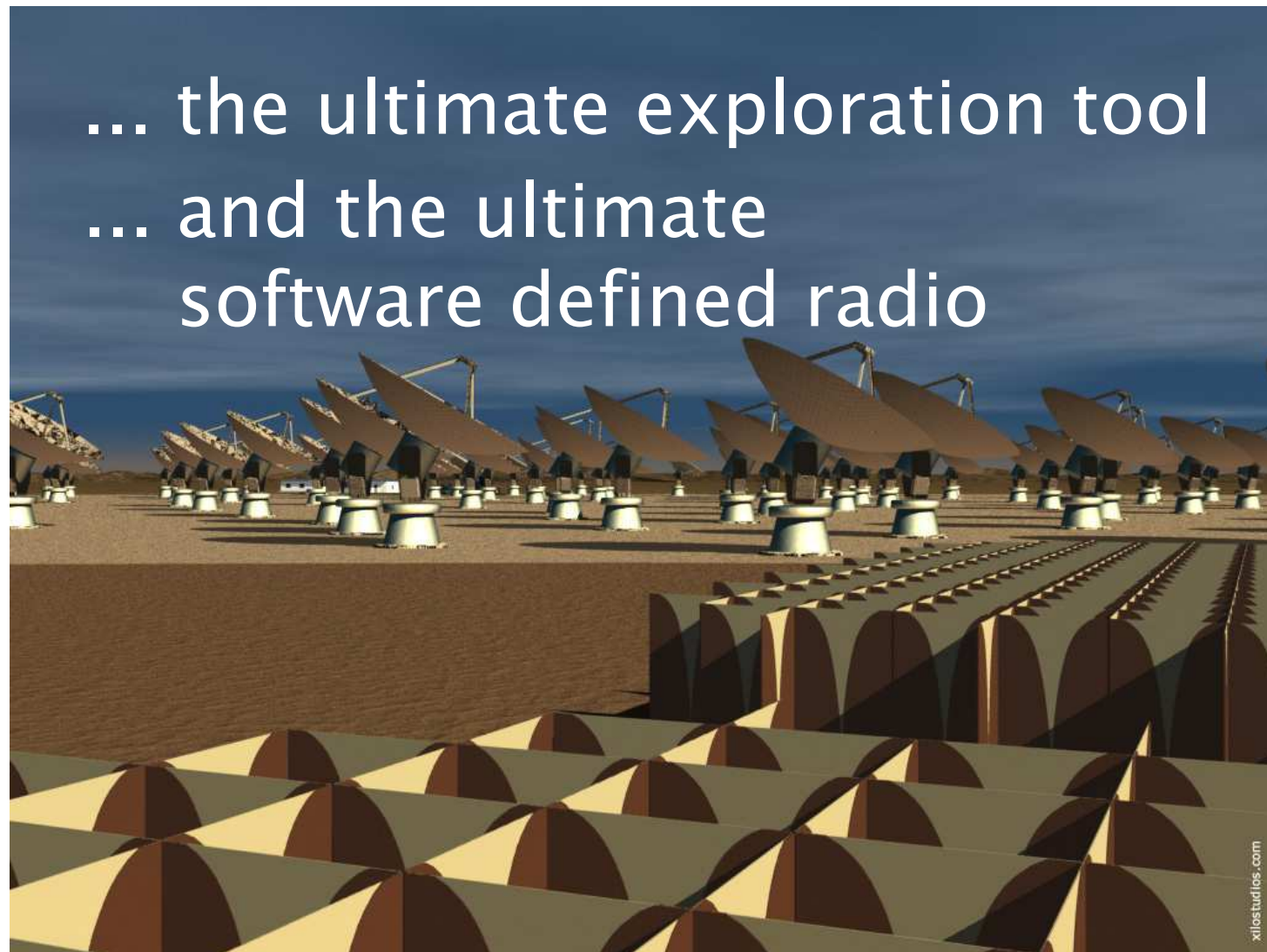
Chess Machine Performance [Moravec '98]



Evolution computer power/cost [Moravec '98]



The Square Kilometer Array (SKA)



The Square Kilometer Array (SKA)

- antenna surface: 1 km² (sensitivity 50×)
- large physical extent (3000+ km)
- wide frequency range: 70 MHz – 30 GHz
- full design by 2010; phase 1: 2017; phase 2: 2022
- 1000– 1500 dishes (15m) in the central 5 km (2000–3000 total)
- + dense and/or sparse aperture arrays
- connected to a massive data processor by an optical fibre network
- **Software Defined Radio Astronomy**
- computational load (on-line) \approx 1 exa MAC (10¹⁸ MAC/s)
- power budget = 30 MW
- \approx 30 pJ/MAC “all-in”

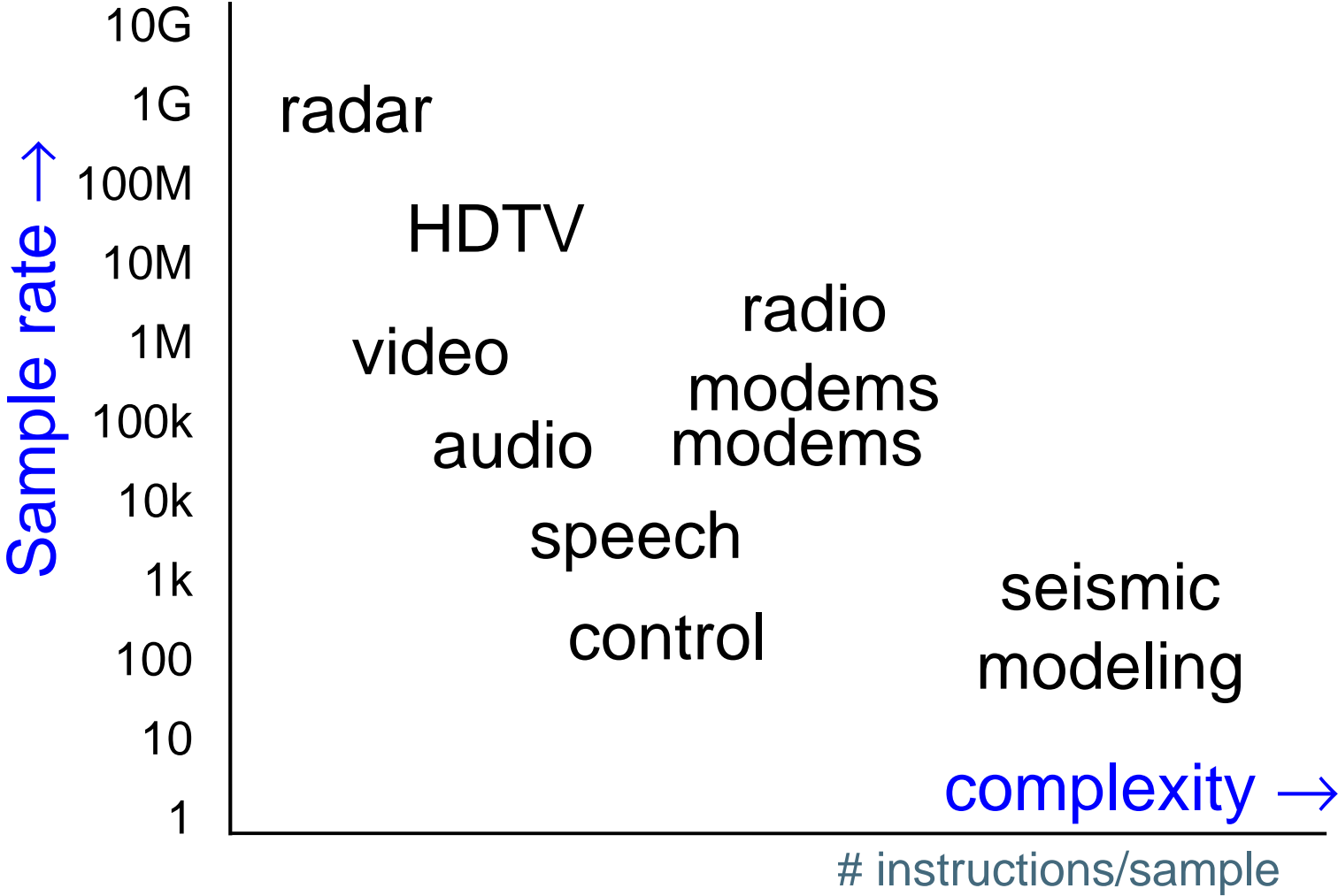
References

- Chip fotos:
- <http://www-vlsi.stanford.edu/group/chips.html>
- ITRS Roadmap
- <http://www.itrs.net/Links/2005ITRS/ExecSum2005.pdf>
- When will computer hardware match the human brain?
- <http://www.jetpress.org/volume1/moravec.htm>
- BEE & Square Kilometer Array
- <http://bwrc.eecs.berkeley.edu/Research/BEE/>
- http://seti.berkeley.edu/casper/papers/BEE2_ska2004_poster.pdf
- <http://www.skatelescope.org/>

VLSI Digital Signal Processing Systems

Parhi, Chapters 1&2

DSP applications classes



Typical DSP algorithms

- speech (de-)coding
- speech recognition
- speech synthesis
- speaker identification
- Hi-fi audio en/decoding
- noise cancellation
- audio equalization
- ambient acoustic emulation.
- sound synthesis
- echo cancellation
- modem: (de-)modulation
- vision
- image (de-)compression
- image composition
- beam cancellation
- spectral estimation
- etc.

Typical DSP algorithms: FIR Filters

- Filters reduce signal noise and enhance image or signal quality by removing unwanted frequencies.
- Finite Impulse Response (FIR) filters compute $y(n)$:

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

- where
 - x is the input sequence
 - y is the output sequence
 - h is the impulse response (filter coefficients)
 - N is the number of taps (coefficients) in the filter
- Output sequence depends only on input sequence and impulse response.

Typical DSP algorithms: IIR Filters

- Infinite Impulse Response (IIR) filters compute:

$$y(i) = \sum_{k=1}^{M-1} a(k)y(i-k) + \sum_{k=0}^{N-1} b(k)x(i-k)$$

- Output sequence depends on input sequence, impulse response, *as well as previous outputs*
- Adaptive filters (FIR and IIR) update their coefficients to minimize the distance between the filter output and the desired signal.

Typical DSP Algorithms: DFT and FFT

The **Discrete Fourier Transform** (DFT) supports frequency domain (“spectral”) analysis:

$$y(k) = \sum_{n=0}^{N-1} W_N^{nk} x(n) \quad W_N = e^{\frac{-2j\pi}{N}} \quad j = \sqrt{-1}$$

for $k = 0, 1, \dots, N-1$, where

- x is the input sequence in the time domain (real or complex)
- y is an output sequence in the frequency domain (complex)

The **Inverse Discrete Fourier Transform** (IDFT) is computed as

$$x(n) = \sum_{k=0}^{N-1} W_N^{-nk} y(k), \quad \text{for } n = 0, 1, \dots, n-1$$

The **Fast Fourier Transform** (FFT) and its inverse (IFFT) provide an efficient method for computing the DFT and IDFT.

Typical DSP Algorithms: DCT

The Discrete Cosine Transform (DCT) and its inverse IDCT are frequently used in video (de-) compression (e.g., MPEG-2):

$$y(k) = e(k) \sum_{n=0}^{N-1} \cos\left[\frac{(2n+1)k\pi}{2N}\right] x(n), \quad \text{for } k=0,1,\dots,N-1$$

$$x(n) = \frac{2}{N} \sum_{k=0}^{N-1} e(k) \cos\left[\frac{(2n+1)k\pi}{2N}\right] y(k), \quad \text{for } k=0,1,\dots,N-1$$

where $e(k) = 1/\sqrt{2}$ if $k = 0$; otherwise $e(k) = 1$.

A N -Point, 1D-DCT requires N^2 MAC operations.

Typical DSP Algorithms: distance calcul.

- Distance calculations are typically used in pattern recognition, motion estimation, and coding.
- Problem: chose the vector r_k whose distance (see below) from the input vector x is minimum.

Mean Absolute Difference (MAD)

$$d = \frac{1}{N} \sum_{i=0}^{N-1} |x(i) - r_k(i)|$$

Mean Square Error (MSE)

$$d = \frac{1}{N} \sum_{i=0}^{N-1} [x(i) - r_k(i)]^2$$

Typical DSP Algorithms: Matrix Computations

Matrix computations are typically used to estimate parameters in DSP systems.

- Matrix vector multiplication
- Matrix–matrix multiplication
- Matrix inversion
- Matrix triangulization

Matrices often have (band) structures or may be sparse.

Computation Rates

- To estimate the hardware resources required, we can use the equation:

$$R_C = R_S \cdot N_S$$

- where
 - R_C is the computation rate
 - R_S is the sampling rate
 - N_S is the (average) number of operations per sample
- For example, a 1-D FIR has $N_S = 2N$ and a 2-D FIR has $N_S = 2N^2$.

Computational Rates for FIR Filtering

Signal type	Frequency	# taps	Performance
Speech	8 kHz	$N = 128$	20 MOPs
Music	48 kHz	$N = 256$	240 MOPs
Video phone	6.75 MHz	$N * N = 81$	1,090 MOPs
TV	27 MHz	$N * N = 81$	4,370 MOPs
HDTV	144 MHz	$N * N = 81$	23,300 MOPs

Typical DSP Programs

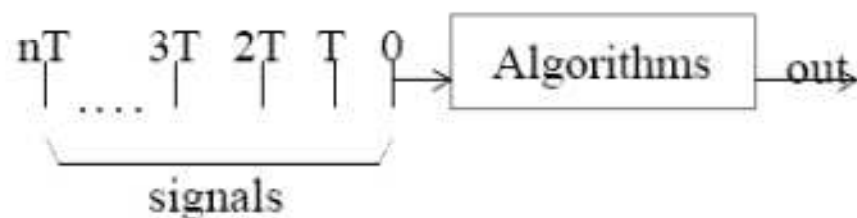
- Usually highly real-time, design hardware and/or software to meet the application speed constraint



- Non-terminating

– Example:

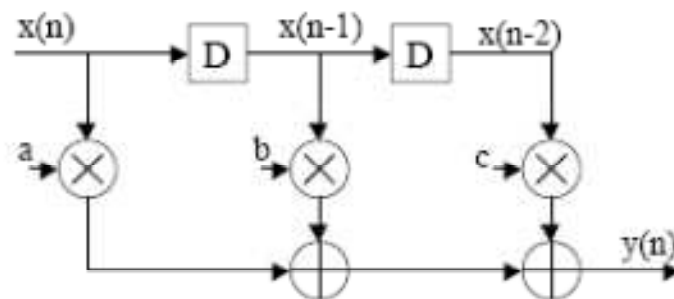
```
for n = 1 to ∞  
    y(n) = a · x(n) + b · x(n - 1) + c · x(n - 2)  
end
```



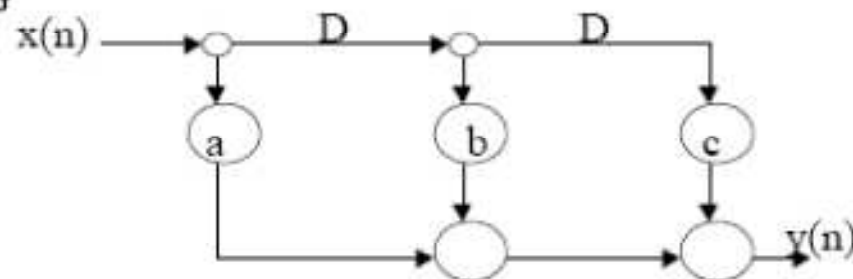
Representation Methods of DSP systems

$$\text{Example: } y(n) = a \cdot x(n) + b \cdot x(n-1) + c \cdot x(n-2)$$

- Graphical Representation Method 1: Block Diagram
 - Consists of functional blocks connected with directed edges, which represent data flow from its input block to its output block

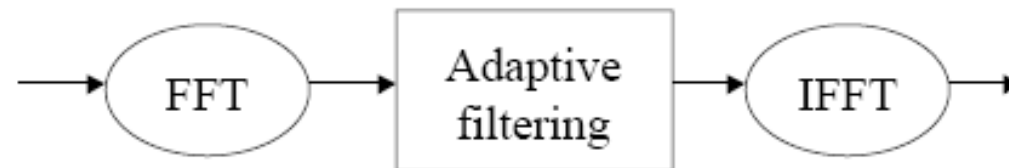


- Graphical Representation Method 3: Data-Flow Graph
 - DFG: nodes represent computations (or functions or subtasks), while the directed edges represent data paths (data communications between nodes), each edge has a nonnegative number of delays associated with it.
 - DFG captures the data-driven property of DSP algorithm: any node can perform its computation whenever all its input data are available.
 - Each edge describes a precedence constraint between two nodes in DFG:
 - Intra-iteration precedence constraint: if the edge has zero delays
 - Inter-iteration precedence constraint: if the edge has one or more delays
 - DFGs and Block Diagrams can be used to describe both linear single-rate and nonlinear **multi-rate** DSP systems
 - Fine-Grain DFG

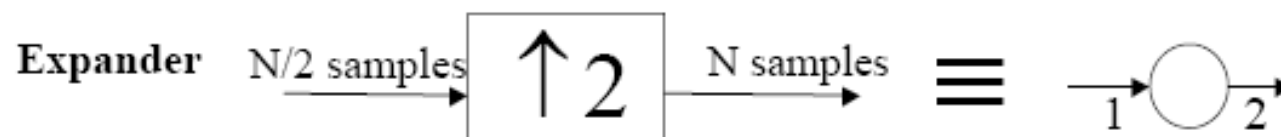
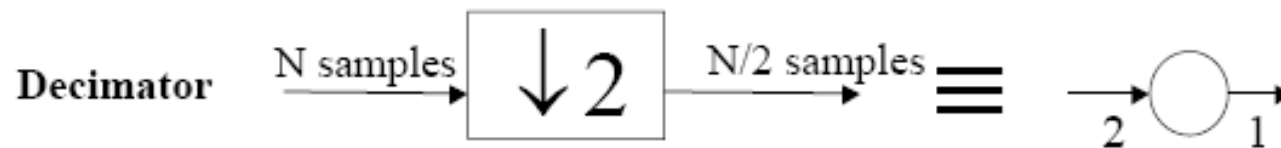


Examples of DFG

- Nodes are complex blocks (in Coarse-Grain DFGs)

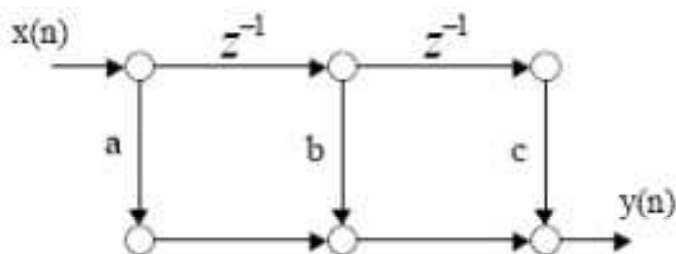


- Nodes can describe expanders/decimators in Multi-Rate DFGs

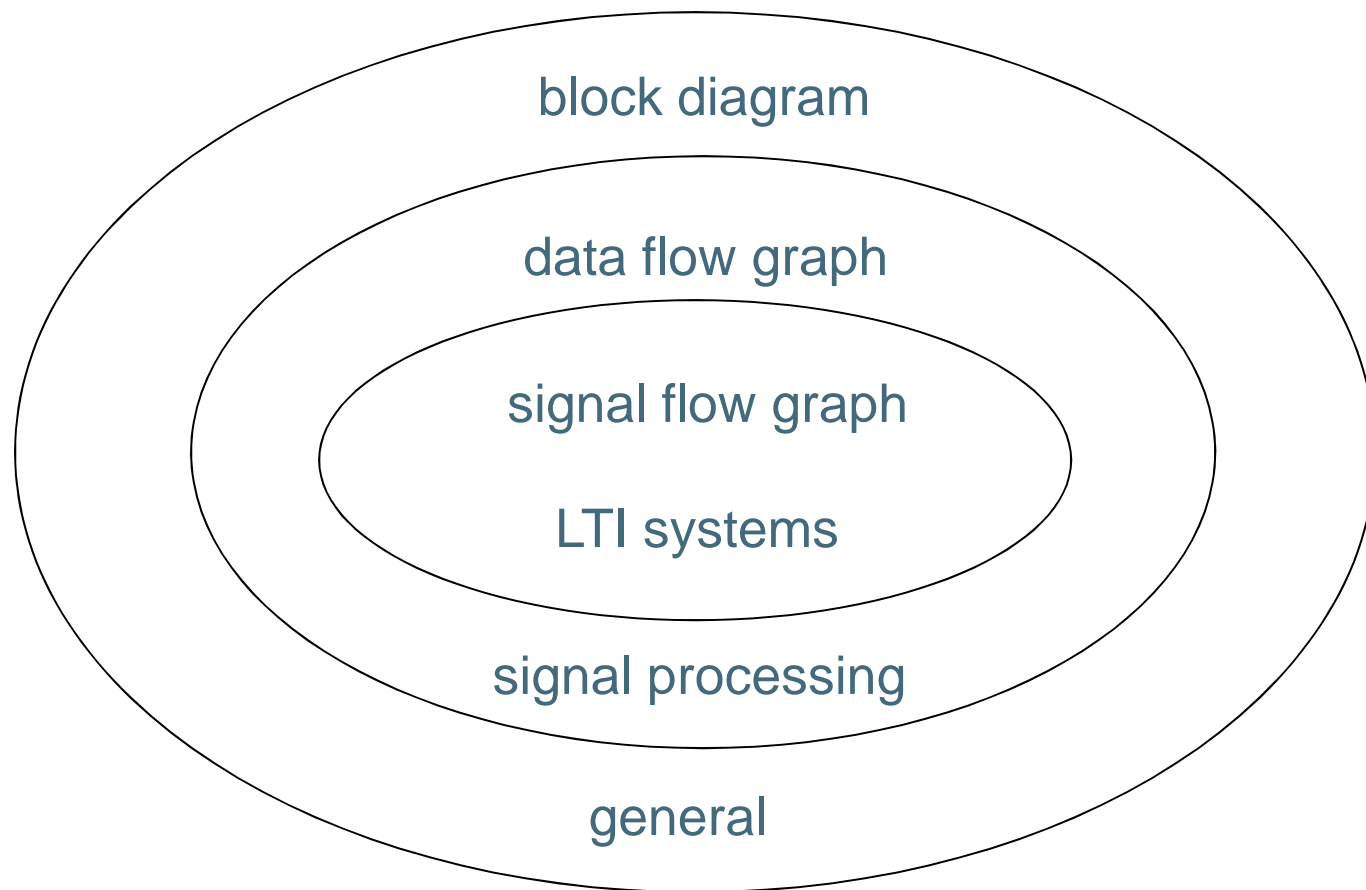


- Graphical Representation Method 2: Signal-Flow Graph
 - SFG: a collection of nodes and directed edges
 - Nodes: represent computations and/or task, sum all incoming signals
 - Directed edge (j, k): denotes a linear transformation from the input signal at node j to the output signal at node k
 - Linear SFGs can be transformed into different forms without changing the system functions. For example, *Flow graph reversal* or *transposition* is one of these transformations (Note: only applicable to single-input-single-output systems)
 - Usually used for linear time-invariant DSP systems representation

- z^{-k} = k units delay
- edge labeled n = multiplication by n



Graphical representations: typical usage



Linear Systems

input x , output y :

discrete system:

- $x(n)$ $\xrightarrow{\text{results in}}$ $y(n)$

linear system:

- $x_1(n) + x_2(n)$ $\xrightarrow{\text{results in}}$ $y_1(n) + y_2(n)$
- $c_1 x_1(n) + c_2 x_2(n)$ $\xrightarrow{\text{results in}}$ $c_1 y_1(n) + c_2 y_2(n)$

for arbitrary c_1 and c_2

Most of our examples will be linear systems

Linear Time-Invariant Systems

input x , output y :

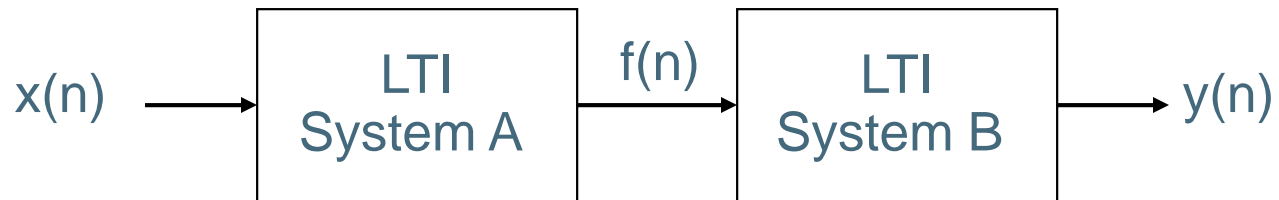
- $x(n+k) = x(n)$ shifted by integer k sample periods

time-invariant system

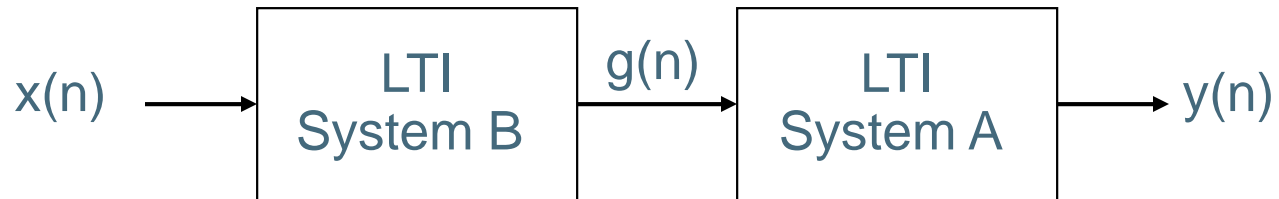
- $x'(n) = x(n+k)$ $\xrightarrow{\text{results in}}$ $y'(n) = y(n+k)$

Most of our examples will be linear time-invariant systems, or LTI systems

Commutativity of LTI systems



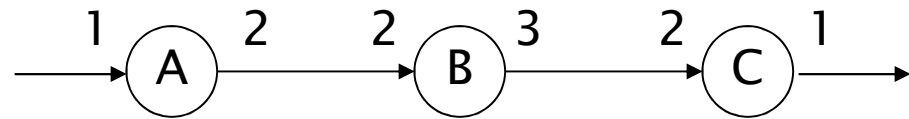
is equivalent to



Iteration of a Synchronous Flow Graph

- Each actor fires the minimum number of times to return the graph to a particular state

- Example of a **multi-rate** DFG:



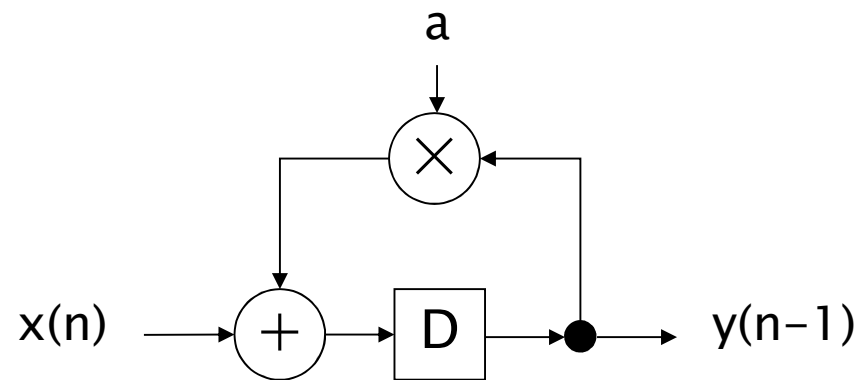
# firings for 1 iteration		
A	B	C
2	2	3

# tokens per edge for 1 iteration			
→ A	A → B	B → C	C →
2	4	6	3

Iteration period

Iteration period =
the time required for the execution of one iteration of the SFG

Example:



Let

- $T_m = 10$ = multiplication time
- $T_a = 4$ = addition time

Iteration period = $T_m + T_a = 14$

= minimum sample period T_s ; that is: $T_s \geq T_m + T_a$

Introduction (cont'd)

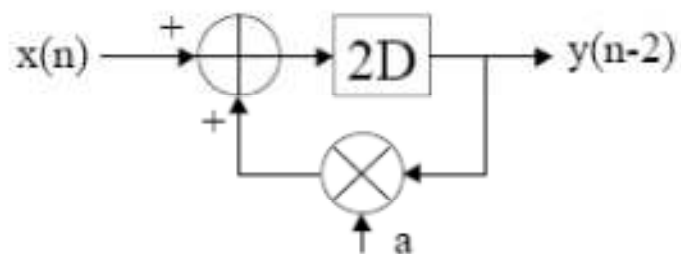
- Assume the execution times of multiplier and adder are T_m & T_a , then the iteration period for this example is $T_m + T_a$ (assume 10ns, see the red-color box). so for the signal, the sample period (T_s) must satisfy:

$$T_s \geq T_m + T_a$$

- Definitions:
 - Iteration rate: the number of iterations executed per second
 - Sample rate: the number of samples processed in the DSP system per second (also called throughput)

Iteration Bound

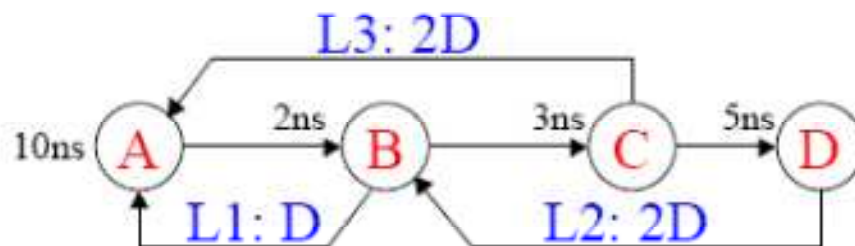
- Definitions:
 - Loop: a directed path that begins and ends at the same node
 - Loop bound of the j -th loop: defined as T_j/W_j , where T_j is the loop computation time & W_j is the number of delays in the loop
 - **Example 1:** $a \rightarrow b \rightarrow c \rightarrow a$ is a loop (see the same example in Note 2, PP2), its loop bound: $T_{loopbound} = T_m + T_a = 10ns$
 - **Example 2:** $y(n) = a*y(n-2) + x(n)$, we have:



$$T_{loopbound} = \frac{T_m + T_a}{2} = 5ns$$

Iteration Bound (cont'd)

- **Example 3:** compute the loop_bounds of the following loops:



$$T_{L1} = (10 + 2)/1 = 12ns$$

$$T_{L2} = (2 + 3 + 5)/2 = 5ns$$

$$T_{L3} = (10 + 2 + 3)/2 = 7.5ns$$

- **Definitions (Important):**

- **Critical Loop:** the loop with the maximum loop bound
- **Iteration bound of a DSP program:** the loop bound of the critical loop, it is defined as

$$T_{\infty} = \max_{j \in L} \left\{ \frac{T_j}{W_j} \right\}$$

where L is the set of loops in the DSP system,

T_j is the computation time of the loop j and

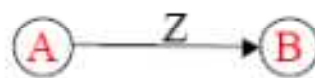
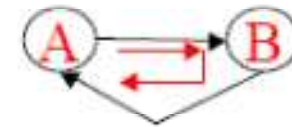
W_j is the number of delays in the loop j

- **Example 4:** compute the iteration bound of the example 3:

$$T_{\infty} = \max_{l \in L} \{12, 5, 7.5\}$$

Iteration bound (cont'd)

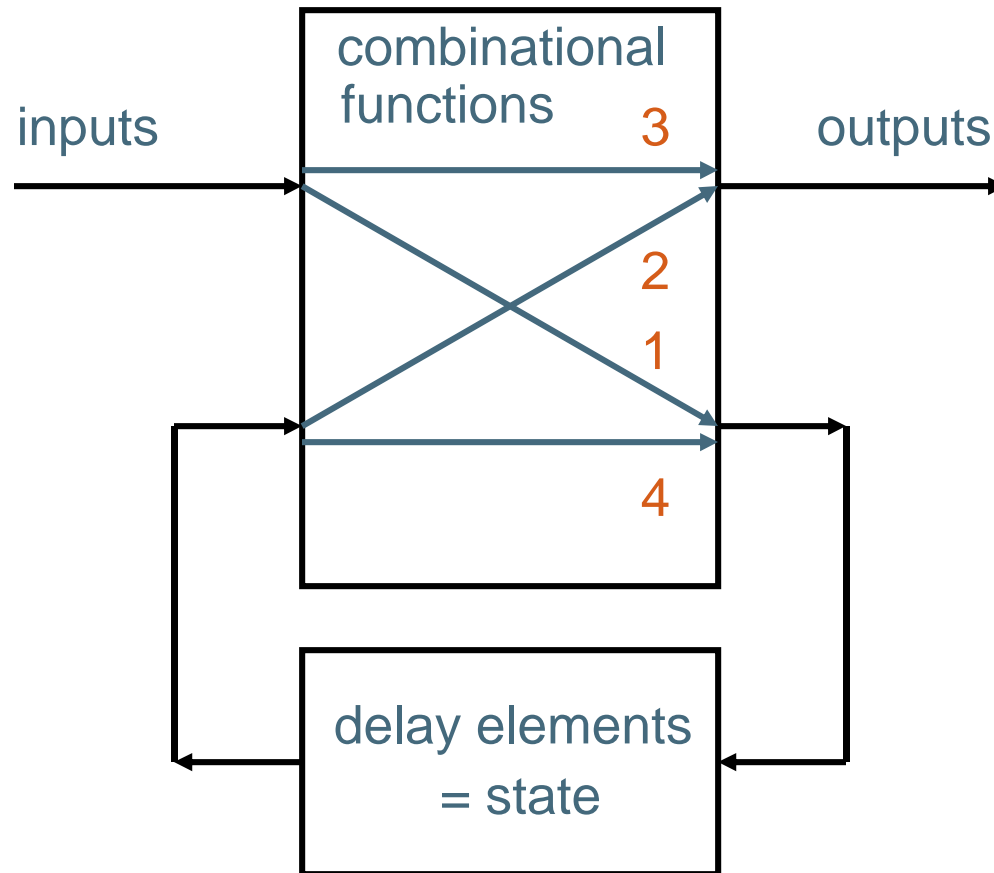
- If no delay element in the loop, then $T_{\infty} = T_L / 0 = \infty$
 - Delay-free loops are non-computable, see the example:
- Non-causal systems cannot be implemented



$$\left\{ \begin{array}{ll} B = A \cdot Z & \text{non-causal} \\ A = B \cdot Z^{-1} & \text{causal} \end{array} \right\}$$

- Speed of the DSP system: depends on the “critical path comp. time”
 - Paths: do not contain delay elements (4 possible path locations)
 - (1) input node \rightarrow delay element
 - (2) delay element's output \rightarrow output node
 - (3) input node \rightarrow output node
 - (4) delay element \rightarrow delay element
 - Critical path of a DFG: the path with the longest computation time among all paths that contain zero delays
 - Clock period is lower bounded by the critical path computation time

4 types of delay paths

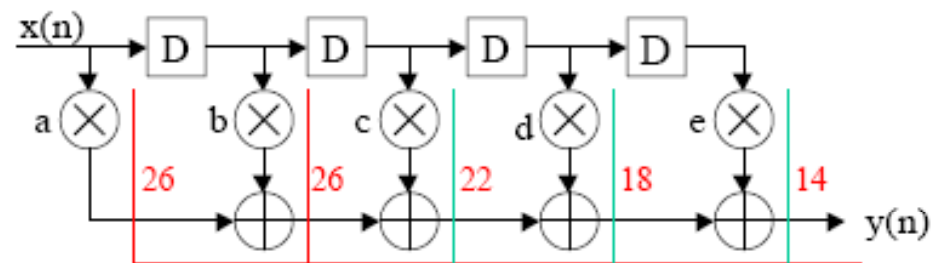


	from	to
1	inputs	state
2	state	outputs
3	inputs	outputs
4	state	state

Finite state machine (FSM) representation of a DSP system

Iteration Bound (cont'd)

- **Example:** Assume $T_m = 10\text{ns}$, $T_a = 4\text{ns}$, then the length of the critical path is 26ns (see the red lines in the following figure)



- Critical path: the lower bound on clock period
- To achieve high-speed, the length of the critical path can be reduced by *pipelining and parallel processing (Chapter 3)*.

DSP references

- Keshab K. Parhi. *VLSI Digital Signal Processing Systems, Design and Implementation*. Wiley Inter–Science 1999.
- Richard G. Lyons. *Understanding Digital Signal Processing* (2nd edition). Prentice Hall 2004.
- John G. Proakis and Dimitris K Manolakis. *Digital Signal Processing* (4th edition), Prentice Hall, 2006.
- Simon Haykin. *Neural Networks, a Comprehensive Foundation* (2nd edition). Prentice Hall 1999.

Computer Architecture and DSP references

- Hennessy and Patterson, Computer Architecture, a Quantitative Approach. 3rd edition. Morgan Kaufmann, 2002.
- Phil Lapsley, Jeff Bier, Amit Sholam, Edward Lee. *DSP Processor Fundamentals*, Berkeley Design Technology, Inc, 1994–1999
- Jennifer Eyre, Jeff Bier, *The Evolution of DSP Processors*, IEEE Signal Processing Magazine, 2000.
- Kees van Berkel et al. *Vector Processing as an Enabler for Software-Defined Radio in Handheld Devices*, EURASIP Journal on Applied Signal Processing 2005:16, 2613–2625.

VLSI Programming: next lecture

- Parhi, Chapters 2, 3
- Representations of DSP algorithms
- Data flow graphs
- Loop bounds and iteration bounds
- Pipelining of digital filters
- Parallel processing
- Retiming techniques

LET'S
CREATE
IT

THANK YOU

