Accelerating computation with FPGAs

Michael J. Flynn
Maxeler Technologies and Stanford University

Based on work done by my colleagues at Maxeler, especially Oskar Mencer, Oliver Pell, Rob Dimond and Jacob Bower

and for the Seismic speedup study

Tamas Nemeth of Chevron

The seismic background slides are courtesy of

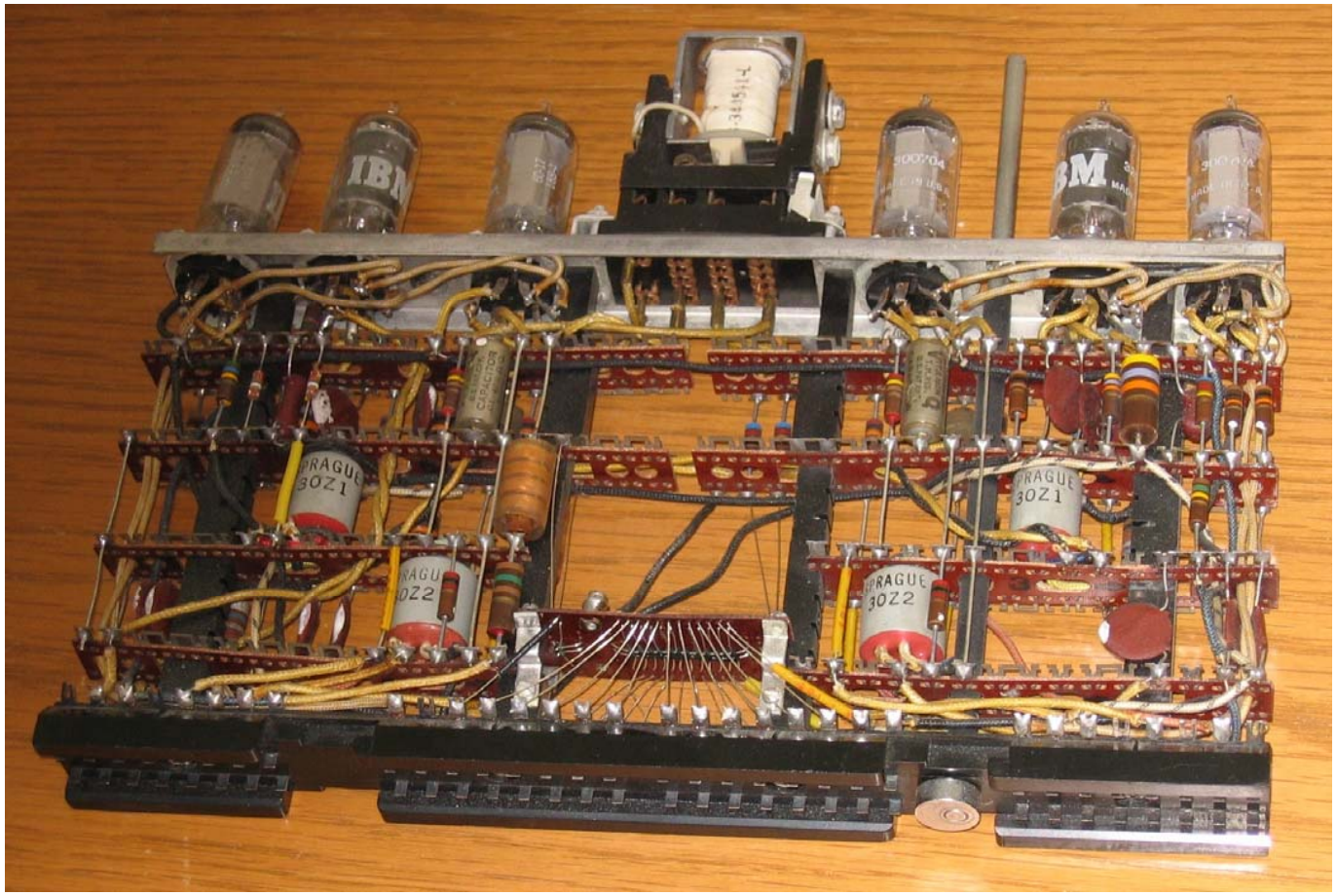
Olav Lindtjorn of Schlumberger and Bob Clapp of Stanford Geophysics

Outline

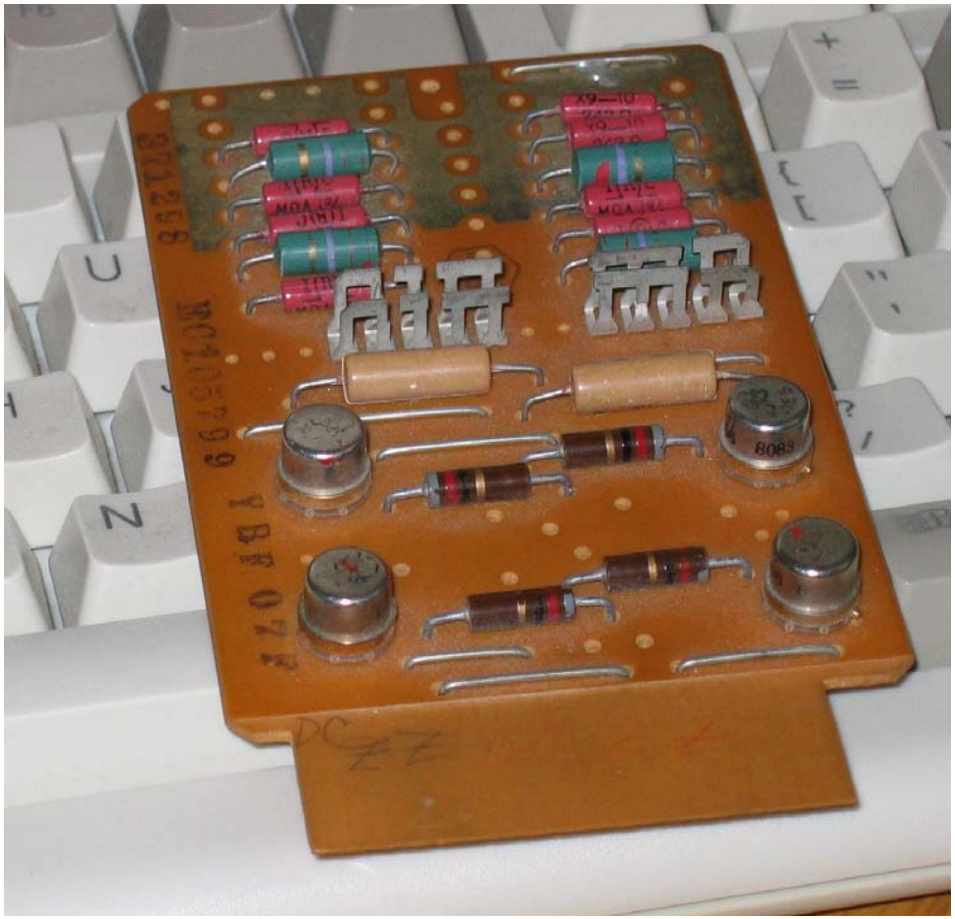
- A walk back to the beginning of hardware “Design Automation”
- An alternative parallel model for hardware and software.
- Maxeler tools and hardware.
- Seismic computation, and an example
- Some results and comments

A walk back to the beginning of hardware “Design Automation”

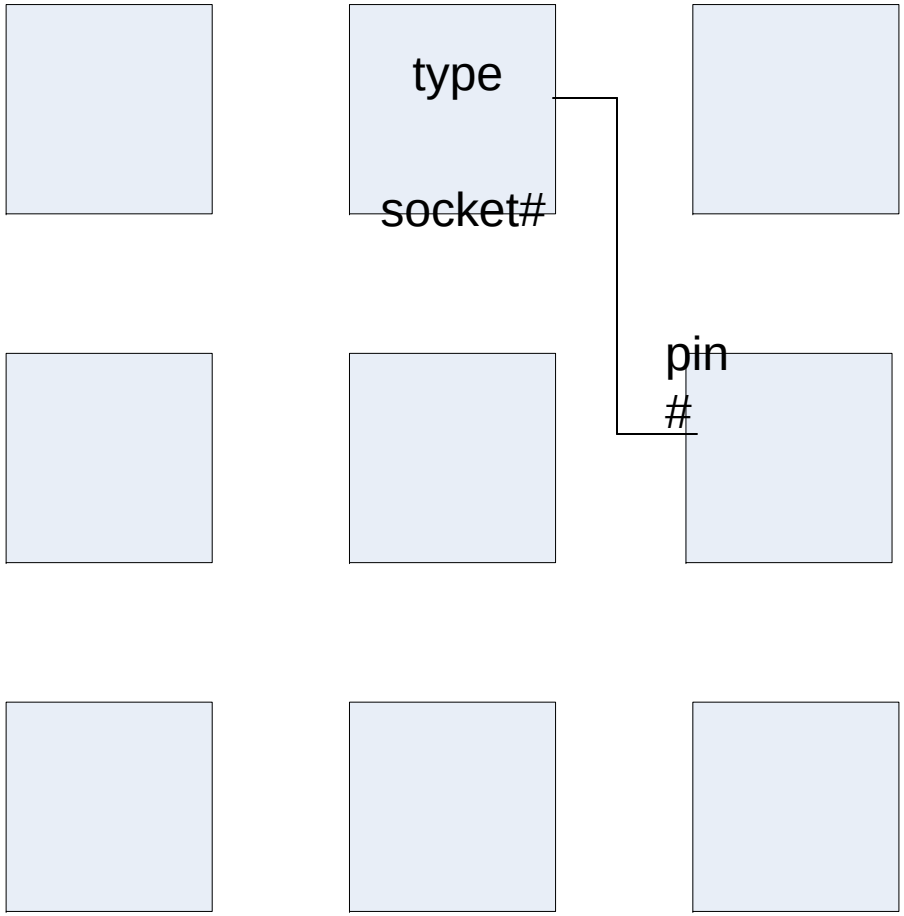
704 / 709 (c.1955)
hand assembly



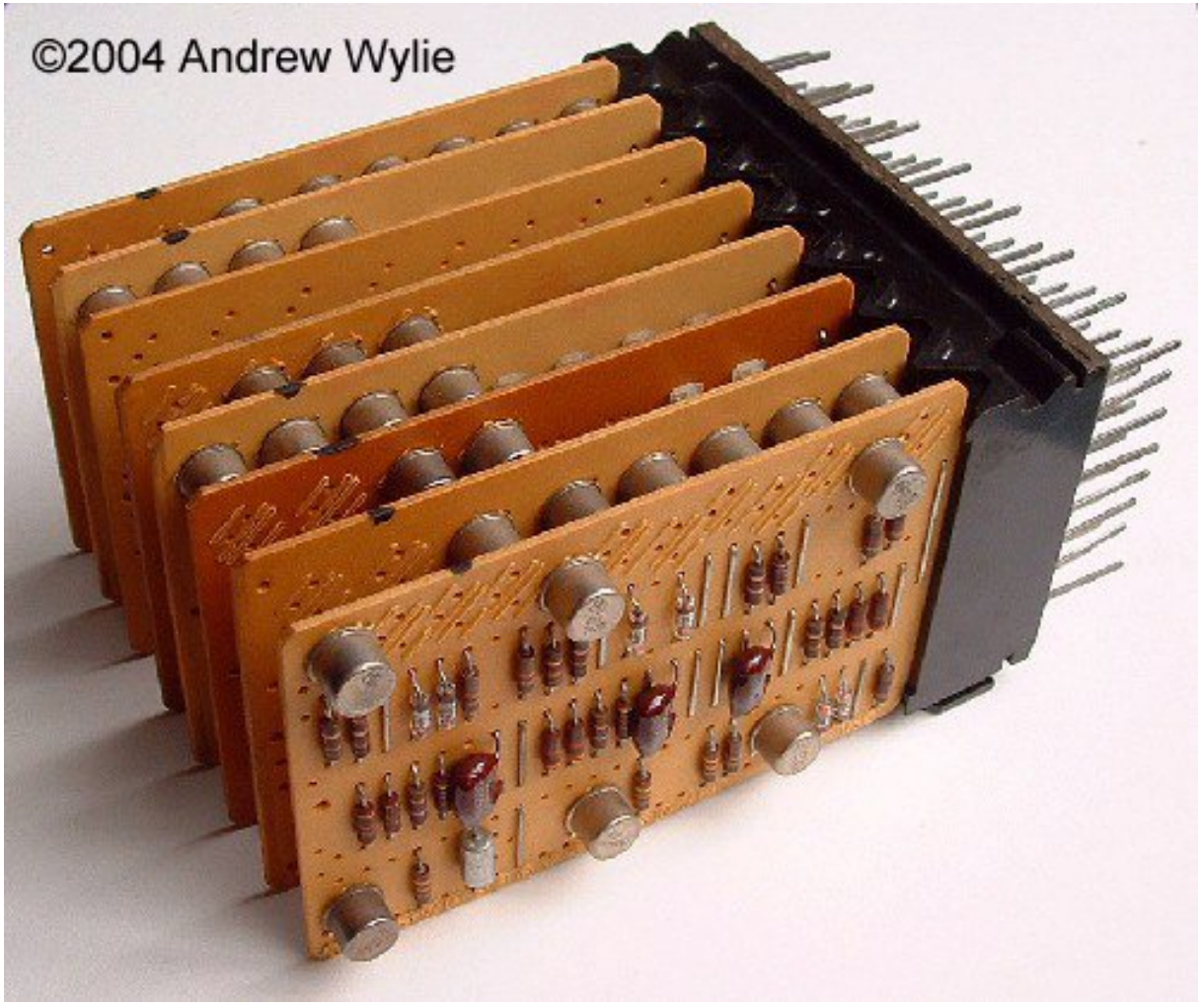
The Standard SMS Card



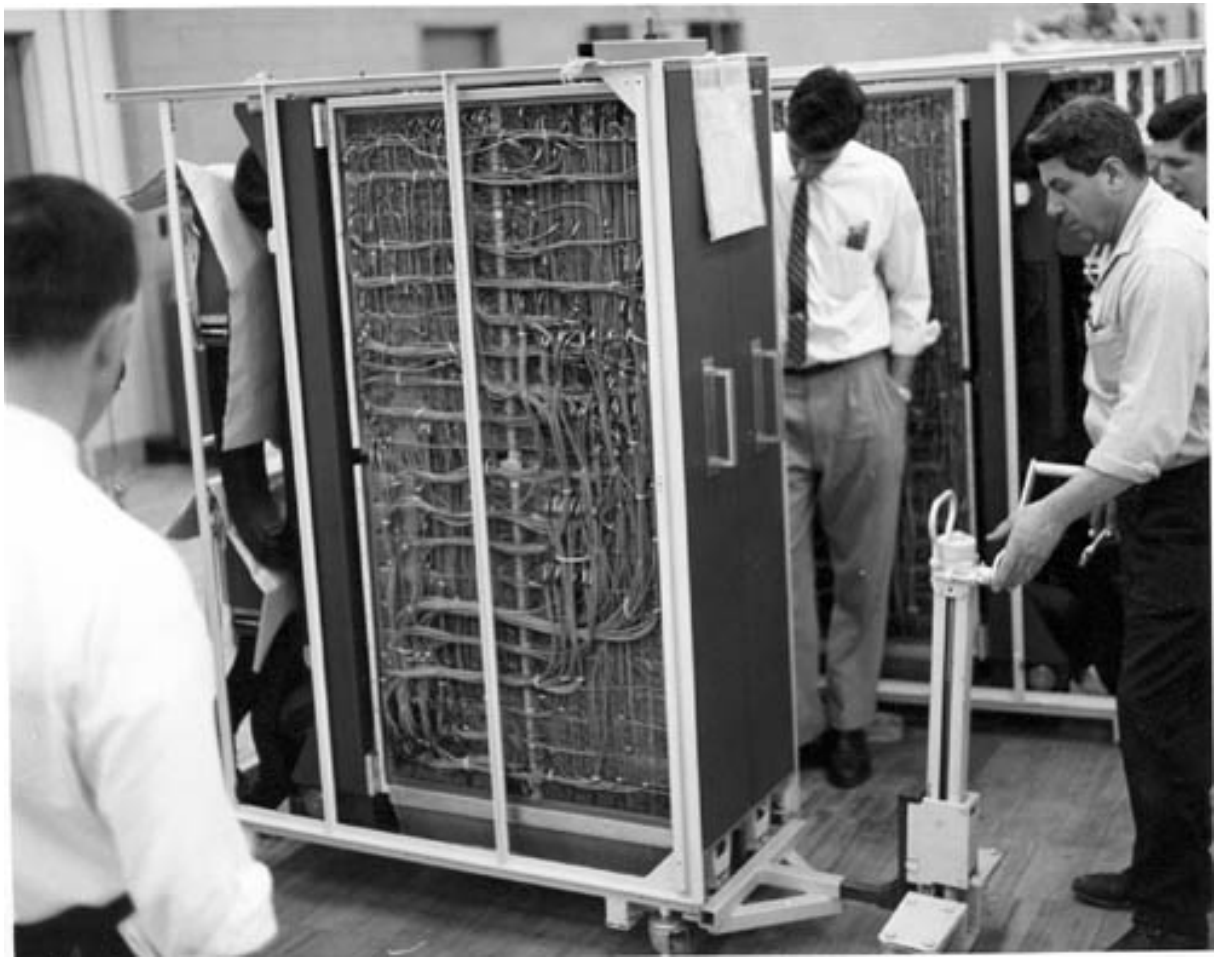
ALDs; each block was a logic gate and represented on a punched card. Machines did the routing and wiring



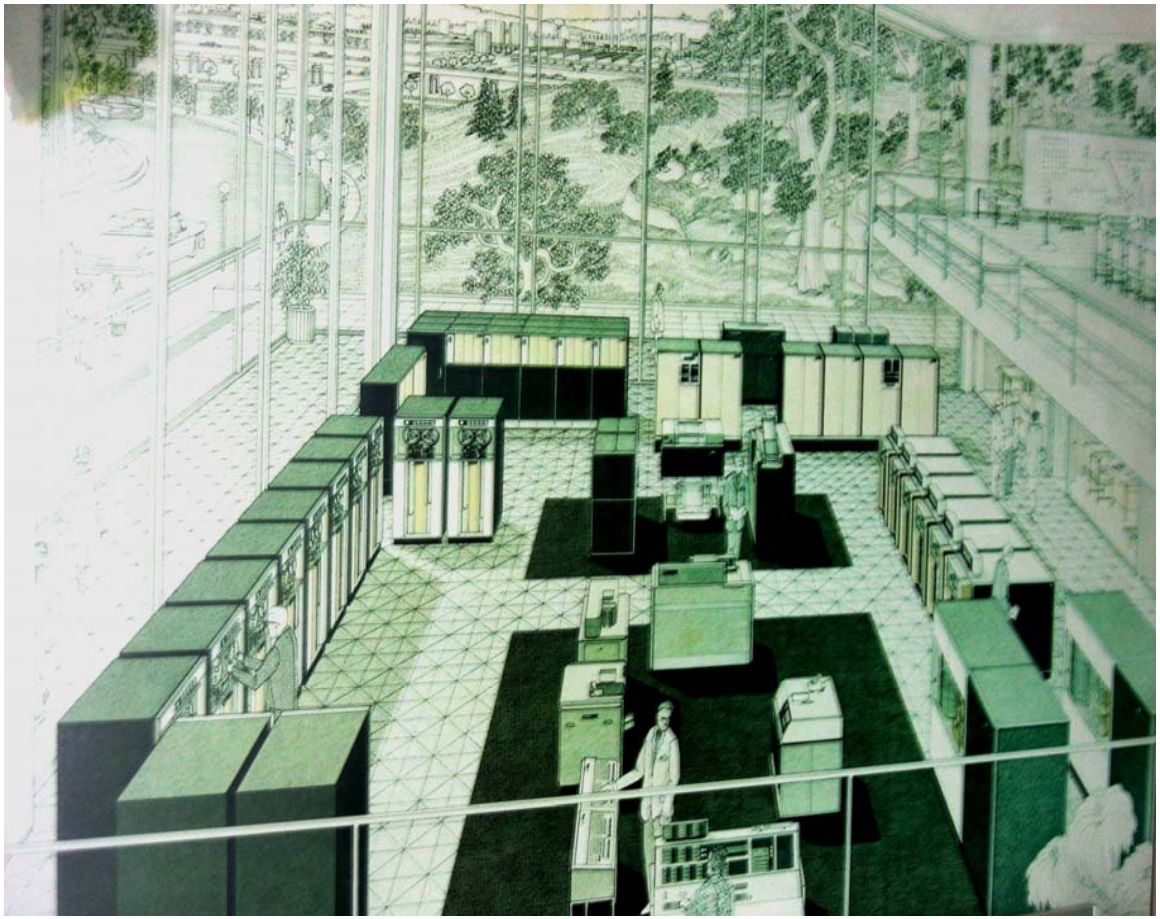
SMS Cards and Sockets



Large SMS frame (7030 and 7090)

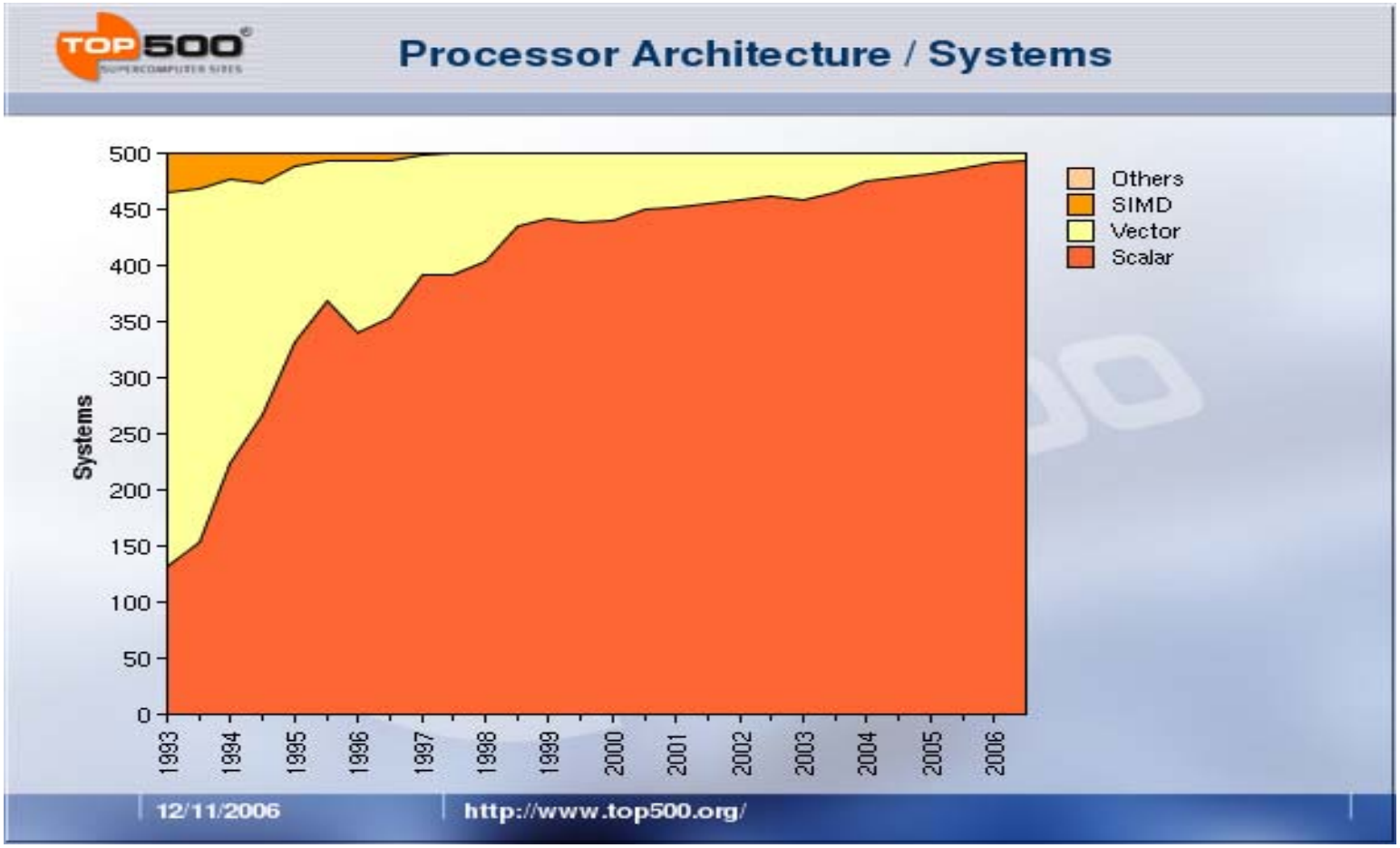


7094 (c.1961); more than 350 made *enabled by design automation*



An alternative parallel model for hardware and software.

Today's HPC architecture



Hardware and software alternatives

- **Hardware:**
  A more generalized (and reconfigurable) parallel array model.

- **Software:**
  A cylindrical rather than
  a layered model suits many applications.

Design space for large applications

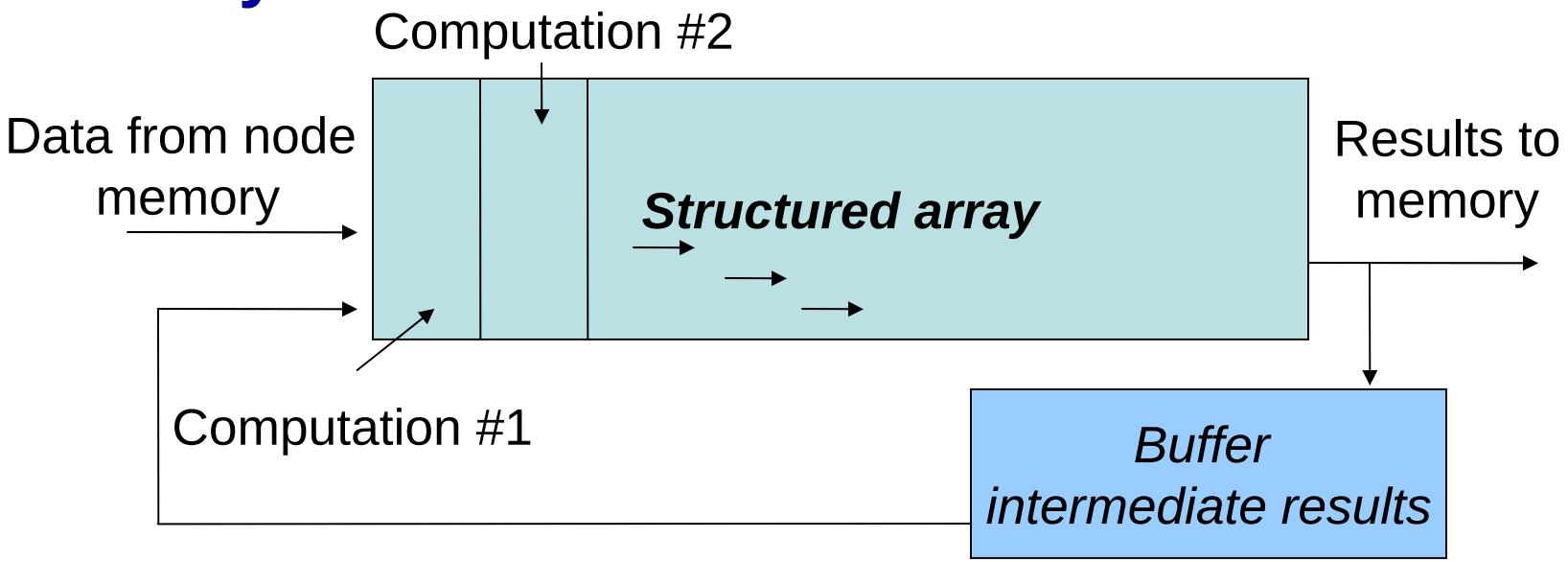
- **Some distinctions**
  - Memory limited vs. compute limited
  - Static vs. dynamic control structure
  - Homogeneous (scalar) vs. core + accelerator at the node

HPC: the case for reconfigurable heterogeneous nodes

- Traditional “scalar” multicore processors are designed to optimize latency, but not throughput.
- **Structured (systolic) arrays**
  FPGAs, GPUs, hyper “core” or “cell” dies, offer complementary throughput acceleration.
- Properly configured, an array can provide 10x-100x improvement in arithmetic (SIMD) or memory bandwidth (by streaming or MISD).

Accelerate tasks by streaming

- MISD structured computation:
  streaming computations across a long array before storing results in memory.
  Can achieve >100x in improved use of memory.



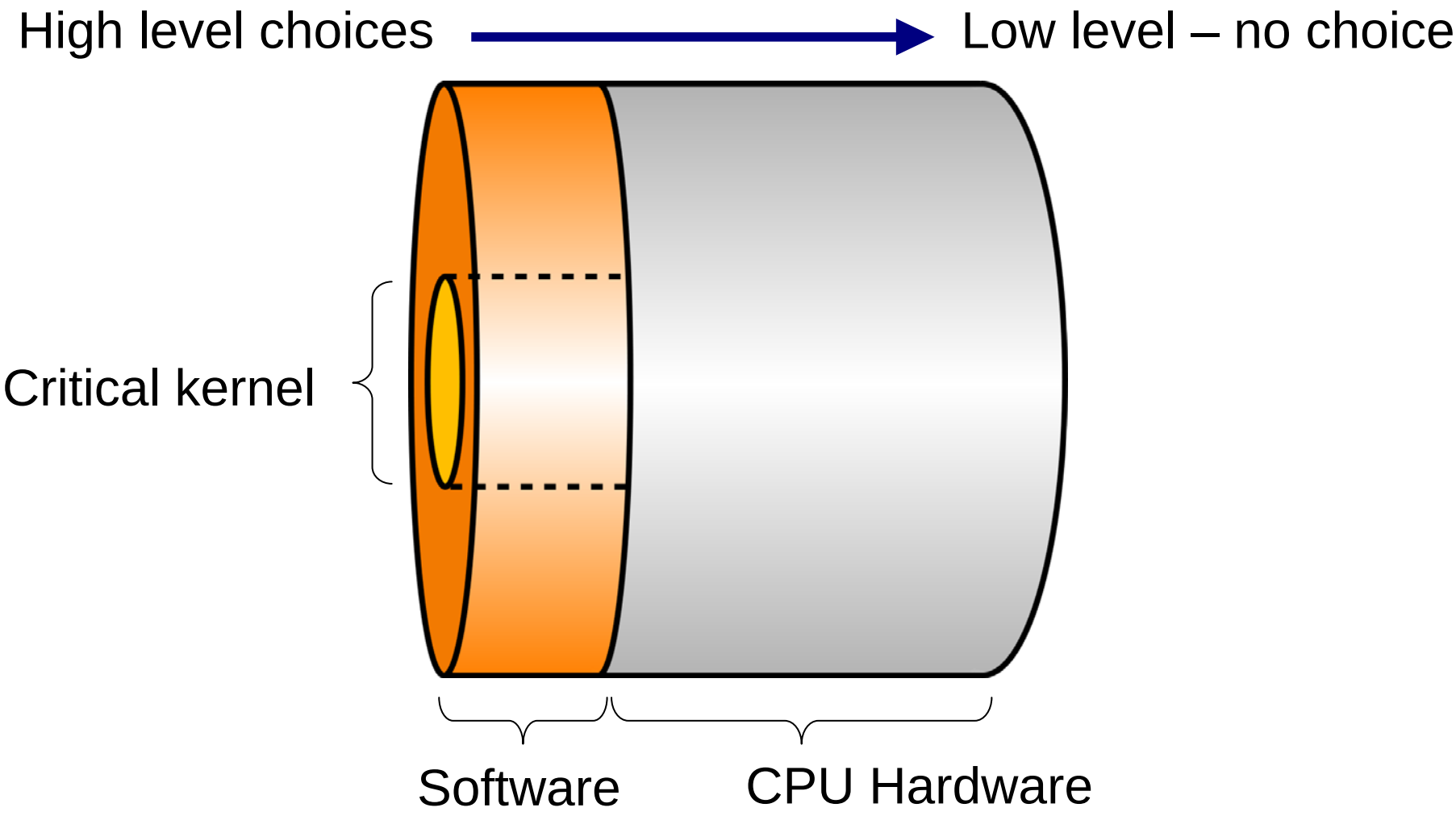
FPGA acceleration

- One tenth the frequency with 10^5 - 10^6 cells per die.
- Magnitude of parallelism overcomes frequency limitations.
- Stream data across large cell array, minimizing memory bandwidth; create a systolic array without nearest neighbour constraints.
- Customized data structures e.g.17 bit floating point; always just enough precision.
- A software (re) configurable technology
- Need an in-depth application study to realize acceleration; acceleration is *not* automatic; acceleration requires more programming effort.

A different programming model

- A cylindrical rather than a layered model suits static applications
- A streaming (MISD) computational model suits memory and (often) compute limited applications.

Cylindrical Model: Non-Accelerated System

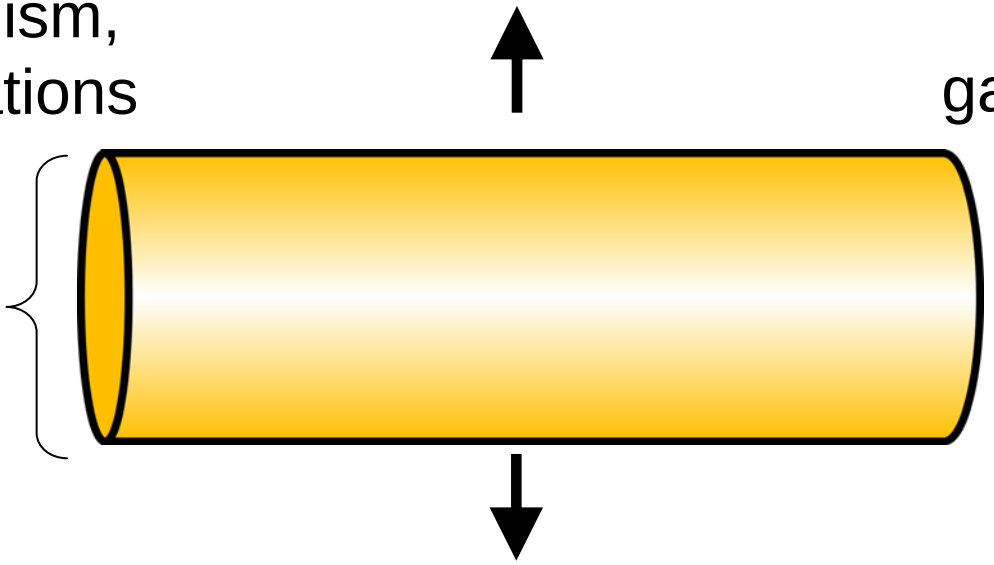


Cylindrical Model: Accelerated System

High level choices
application choices,
algorithms,
types of parallelism,
data representations

optimize

Low level choices
communications,
routing,
timing,
gate delays



- First accelerate the (initially small) kernel
- Later extend kernel size

Speedup with the Cylindrical Model

- Transform application to execute multiple simultaneous strips using DRAM “pipes”
- Stream computations through each pipe
- Strip size limited by FPGA area and DRAM bandwidth
  - Application specific data precision
    - multiplies FPGA area
    - multiplies DRAM bandwidth

Too much effort?

"The parallel approach to computing does require that some original thinking be done about numerical analysis and data management in order to secure efficient use.

In an environment which has represented the absence of the need to think as the highest virtue this is a decided disadvantage."

-Daniel Slotnick, 1967

Maxeler approach and tools.

MAXware: acceleration support

- Acceleration methodology
- Application tool set
- Acceleration hardware from Maxeler

Acceleration methodology

- Four stages:
  - Analysis of the program (static and dynamic)
  - Transformation: create dataflow graph, data layout and representation.
  - Partitioning: optimize dataflow, data access and data representation options.
  - Implementation: uses Maxeler compiler to configure application for FPGAs, and set run time interfaces.

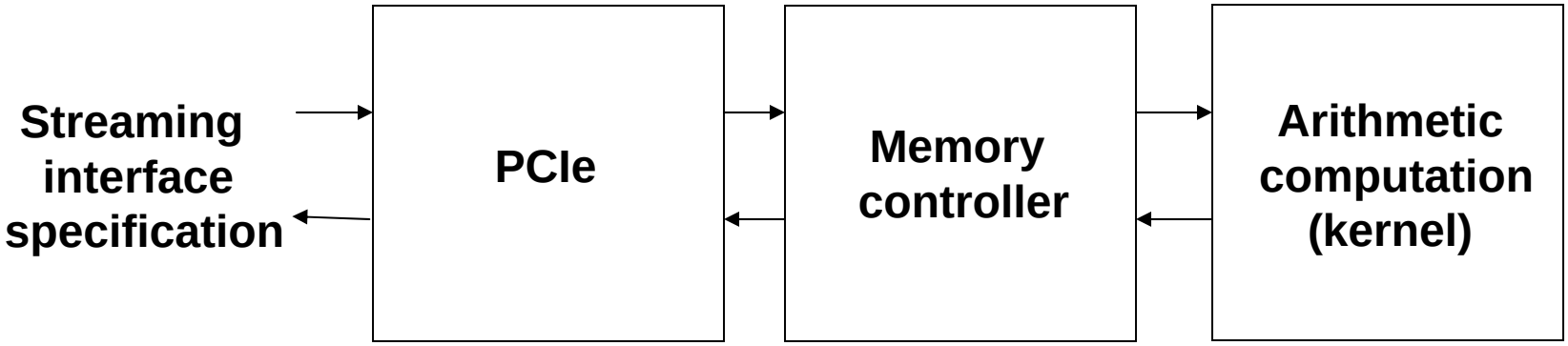
Maxeler tool set

- Parton profiles running application, finds hot spots and suitability for FPGA
  key metric: computations per memory access
- Performance modeller estimates execution time of alternative designs for FPGA; accurate to 10%
- MaxelerOS support for MAX2 board
- MaxQ: FPGA bitstream manufacturing

Maxeler Compiler

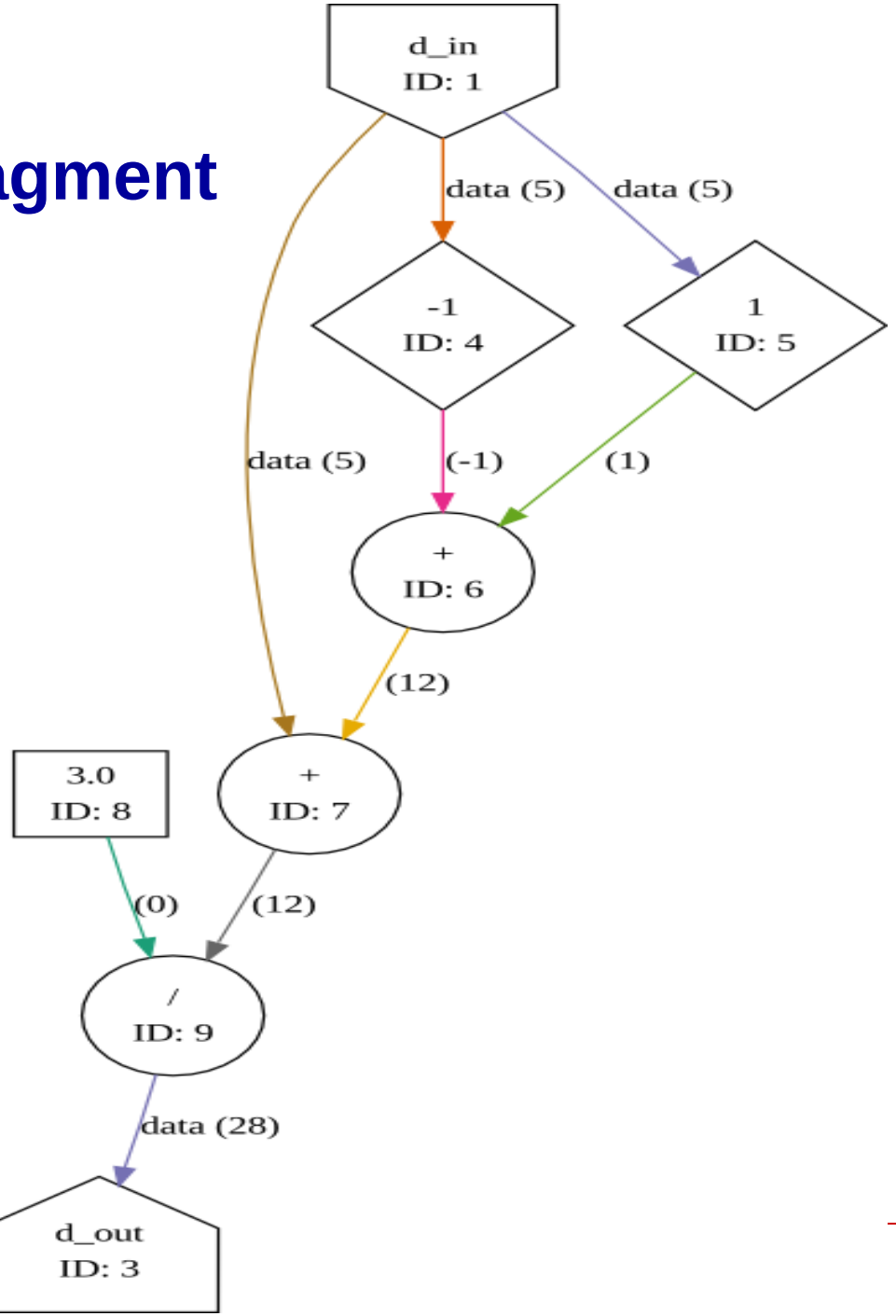
1) User application written in (implementable) Java
2) Code is analyzed and transformed to match data bandwidth
3) Maxeler manager compiler
   generates structural hardware design.
4) Maxeler kernel compiler
   generates data flow graph.

Maxeler Manager Compiler

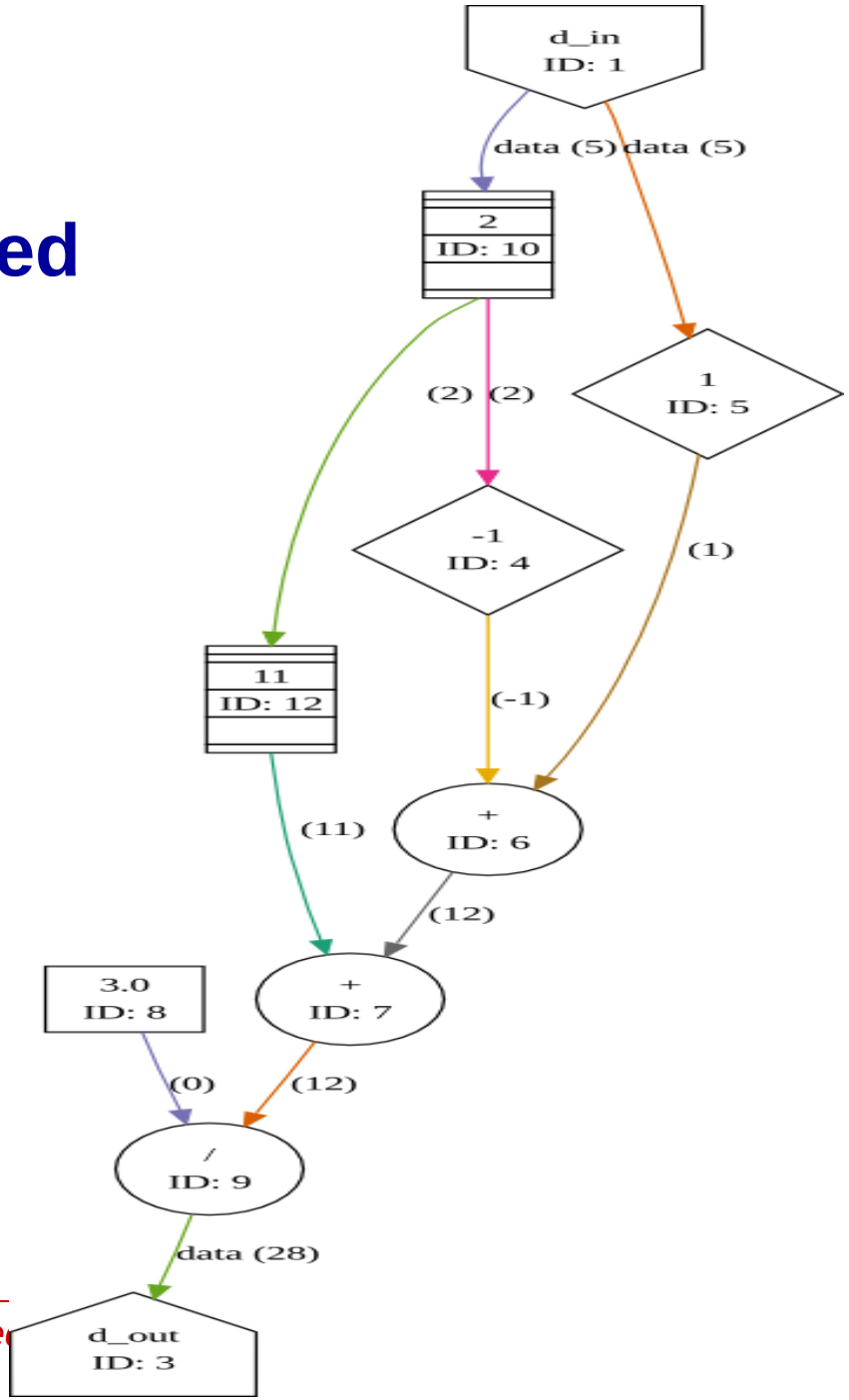


- connects blocks in different clock-domains
- bridges streaming blocks w/ different configurations
- performs aspect-changes on data flow

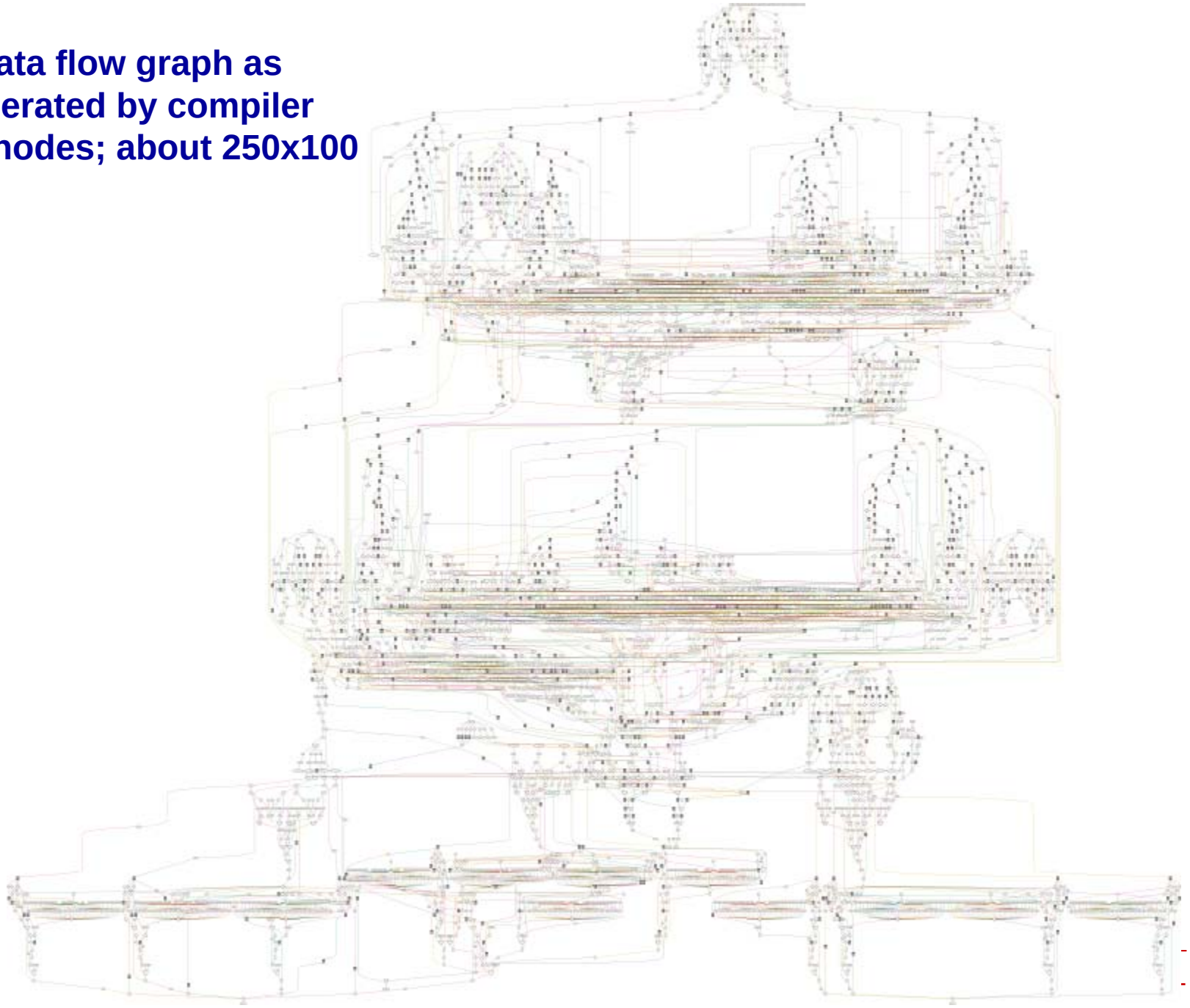
Data flow graph fragment



Same fragment
now buffer synchronized



Data flow graph as generated by compiler
4866 nodes; about 250x100

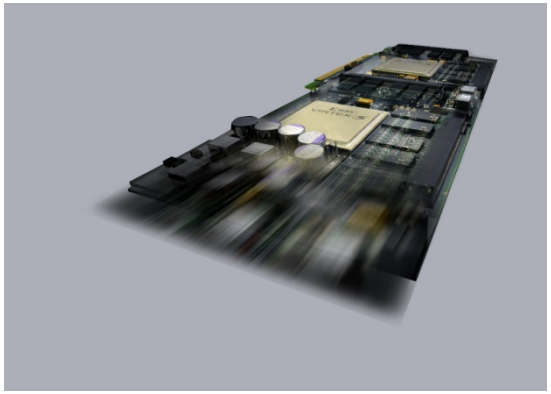


Maxeler Kernel Compiler

- Takes data flow graph as input
- Performs a number of transformations
  - Pre-scheduling optimizations
    e.g. constant-folding
  - Adds scheduling buffers
  - Buffer optimization
- Generates a structural hardware design

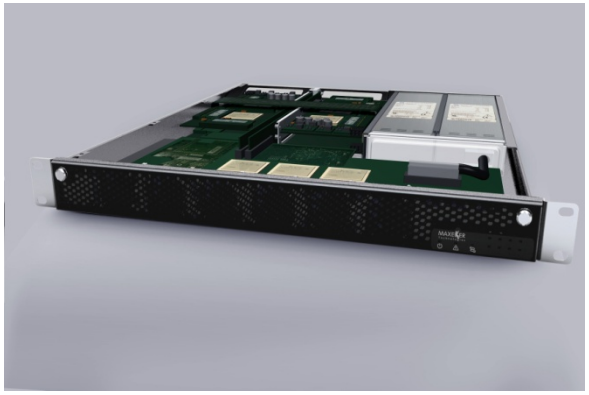
Streaming hardware examples

MaxCard
PCI Express x16 Card



max 75W, typical 50W

MaxBox
1U server with 1 MaxCard
or 4 MaxCards in a 1U box



MaxRack
10U, 20U or 40U

from Maxeler Technologies

The MAX2 board

- 2 x Virtex 5 LX330T each with
  24 GB memory with PCI express x16 link.
- 10 GBPS inter FPGA link;
  total off-chip bandwidth of 48 GBPS.
- MaxelerOS has Linux device drivers,
  manages all PCI & DRAM “plumbing”.

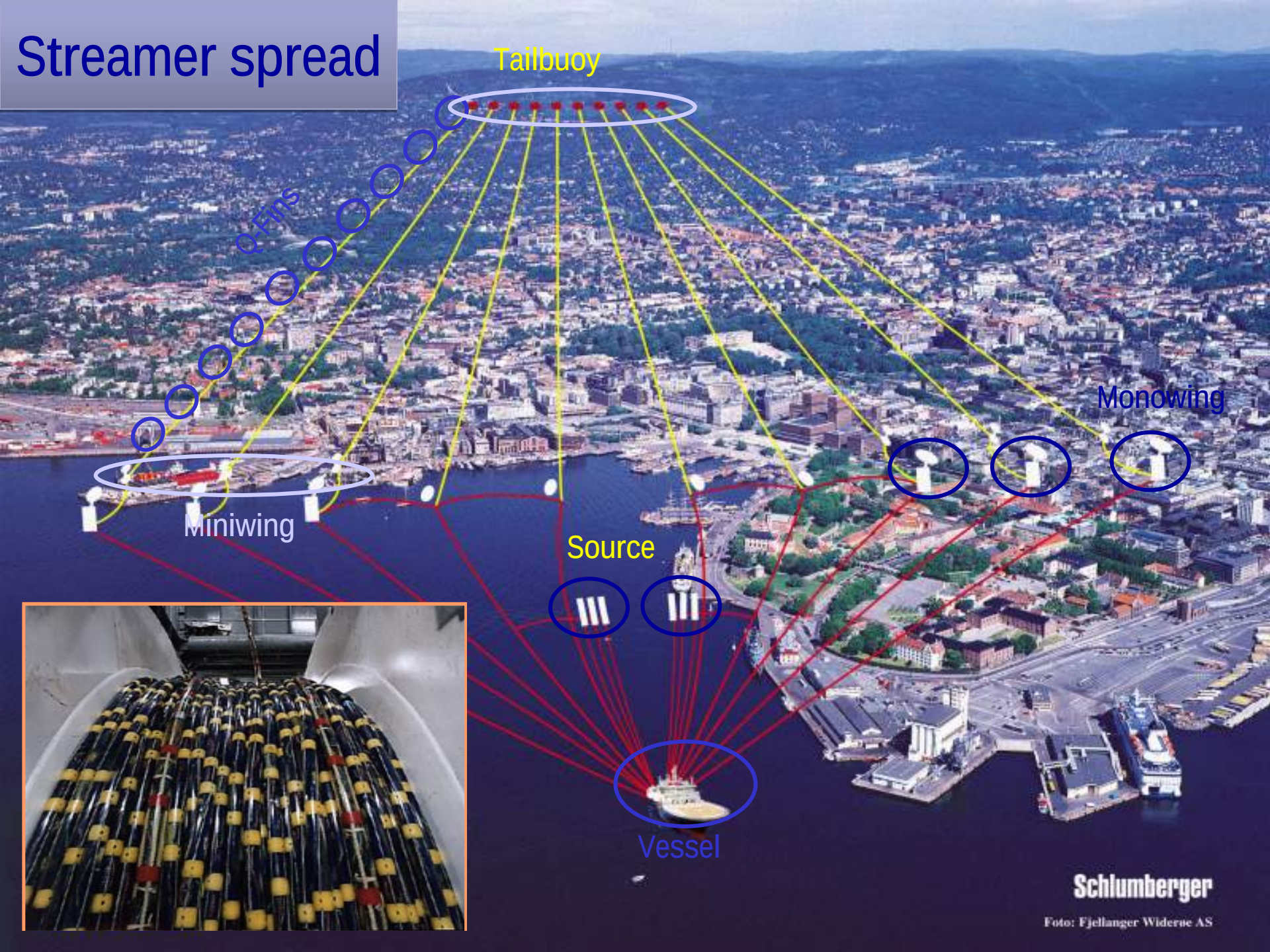
Seismic computation

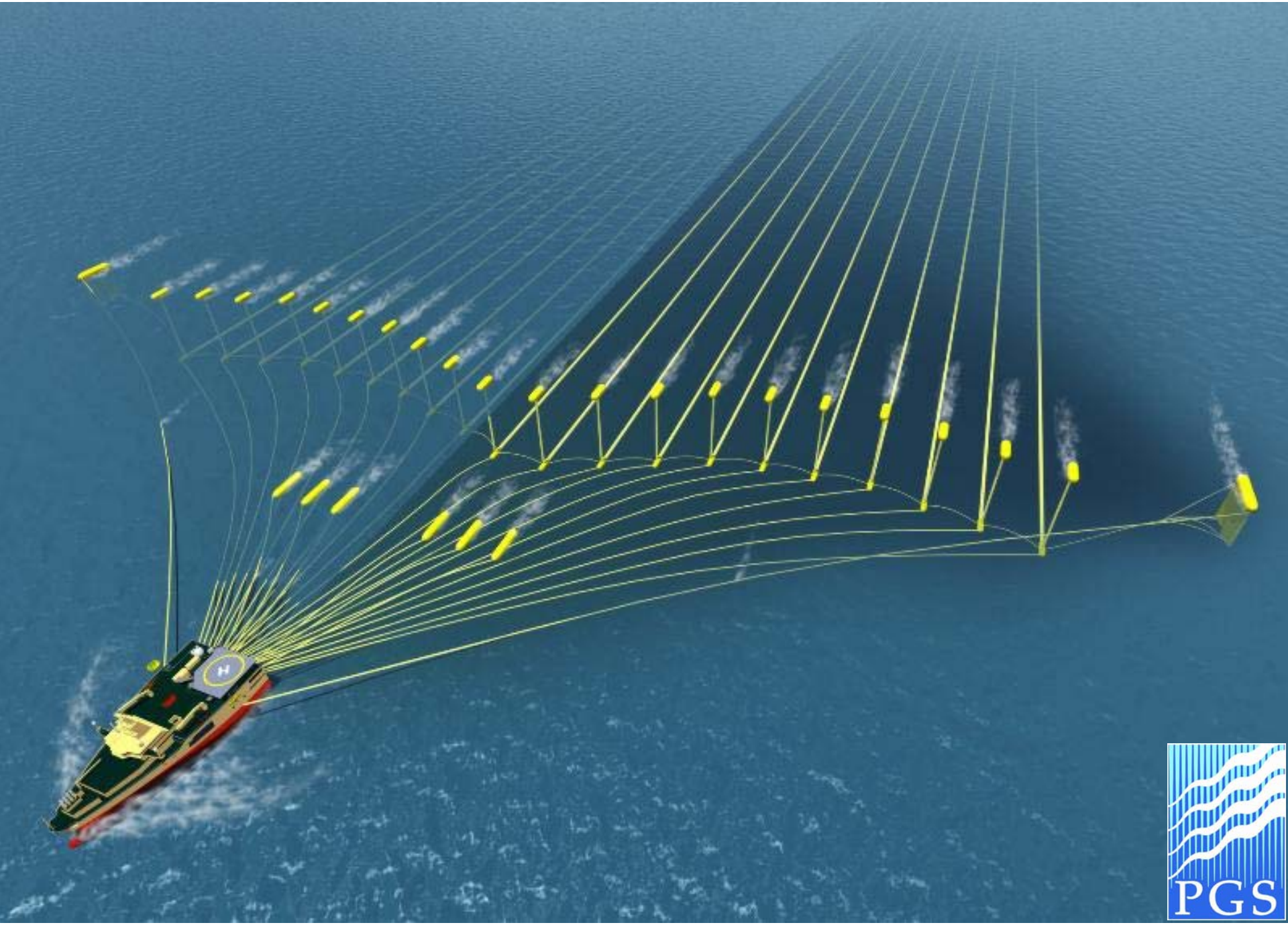
Example: seismic data processing

- For O&G exploration:
  distribute grid of sensors over large area
- Sonic impulse the area and record reflections:
  frequency, amplitude, delay at each sensor
- Sea based surveys use 30,000 sensors to record data (120 db range) each sampled at more than 2kbps with new sonic impulse every 10 sec.

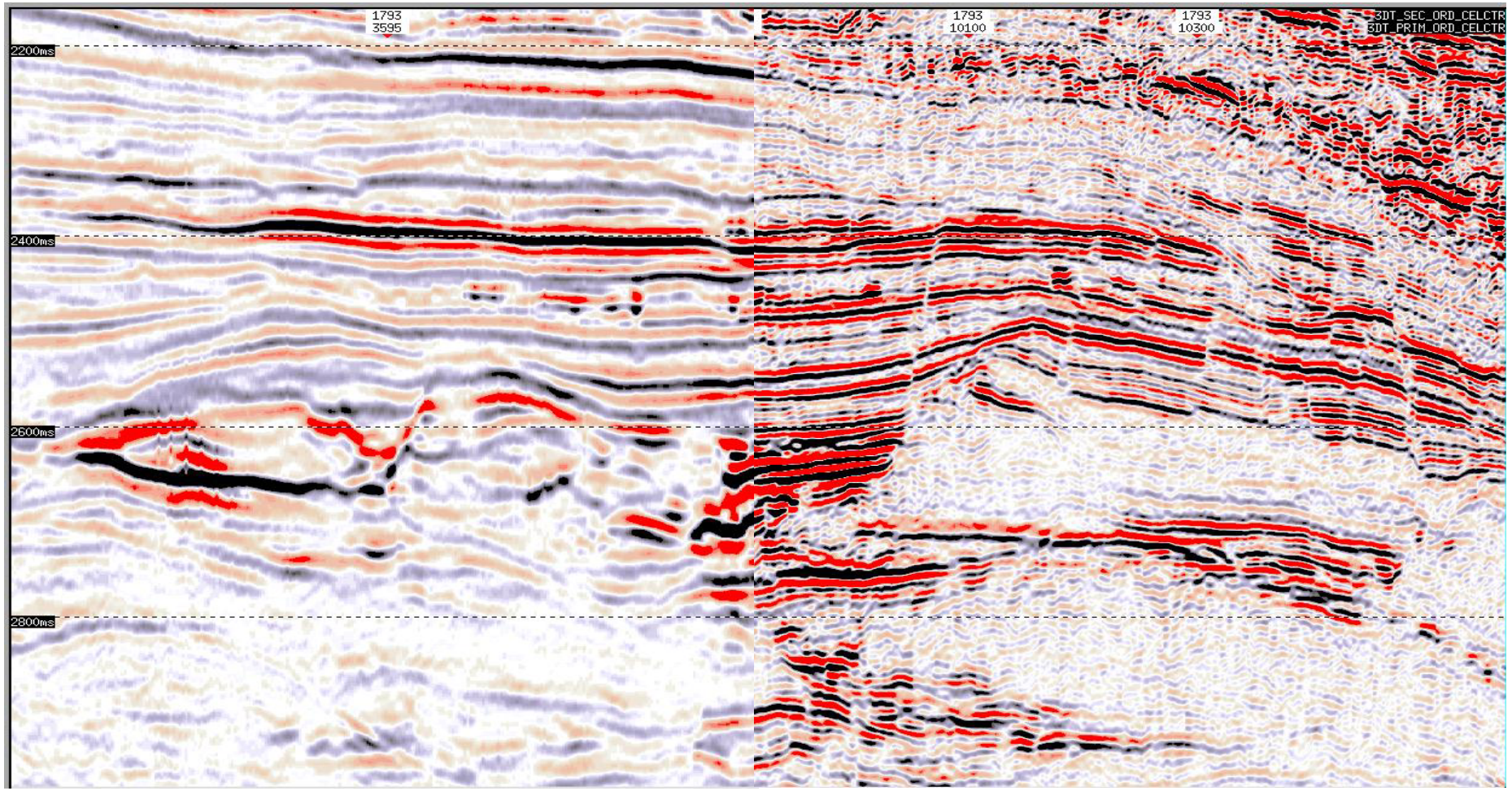
  Order of terabytes of data each day.

Streamer spread



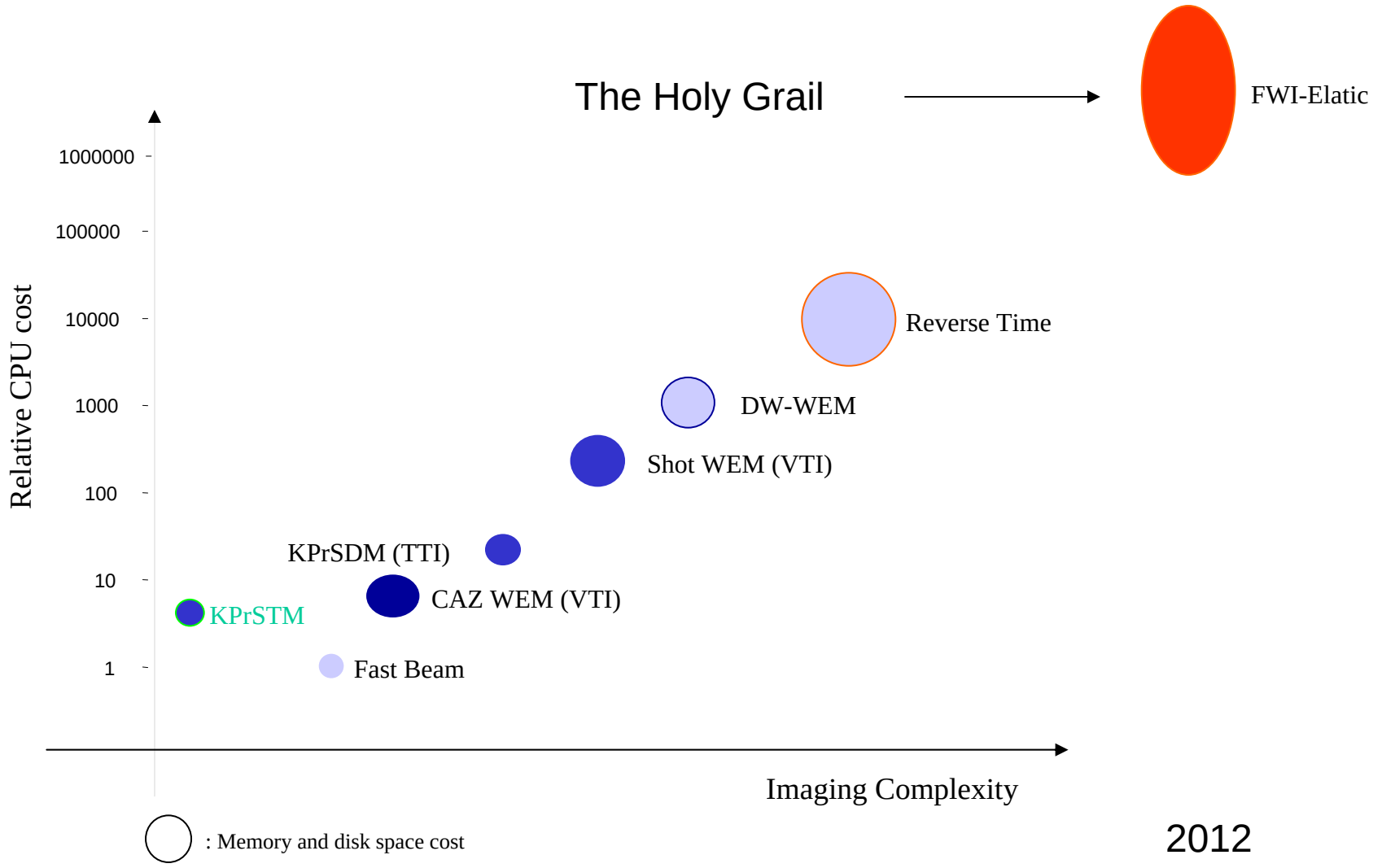

PGS

Q-Technology: Higher Resolution & Fidelity



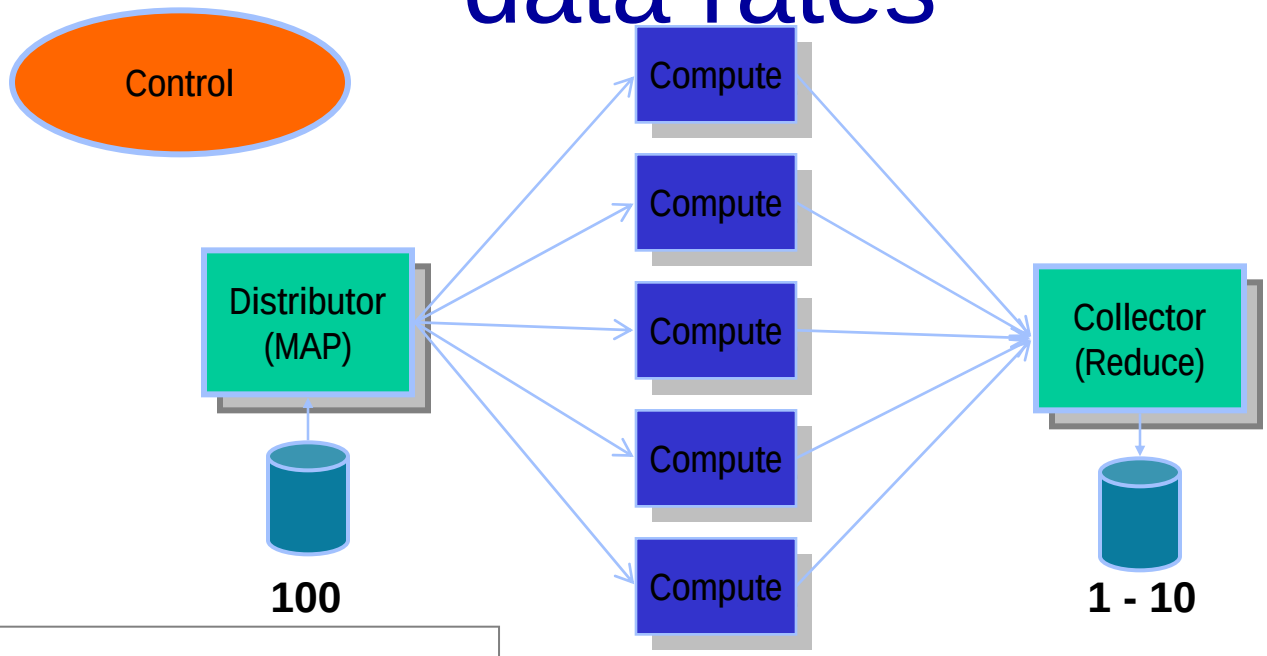
Slide courtesy of Olav Lindtjorn

Relative Cost of Imaging Algorithms



Slide courtesy of Olav Lindtjorn

Application Architecture and data rates



Data:

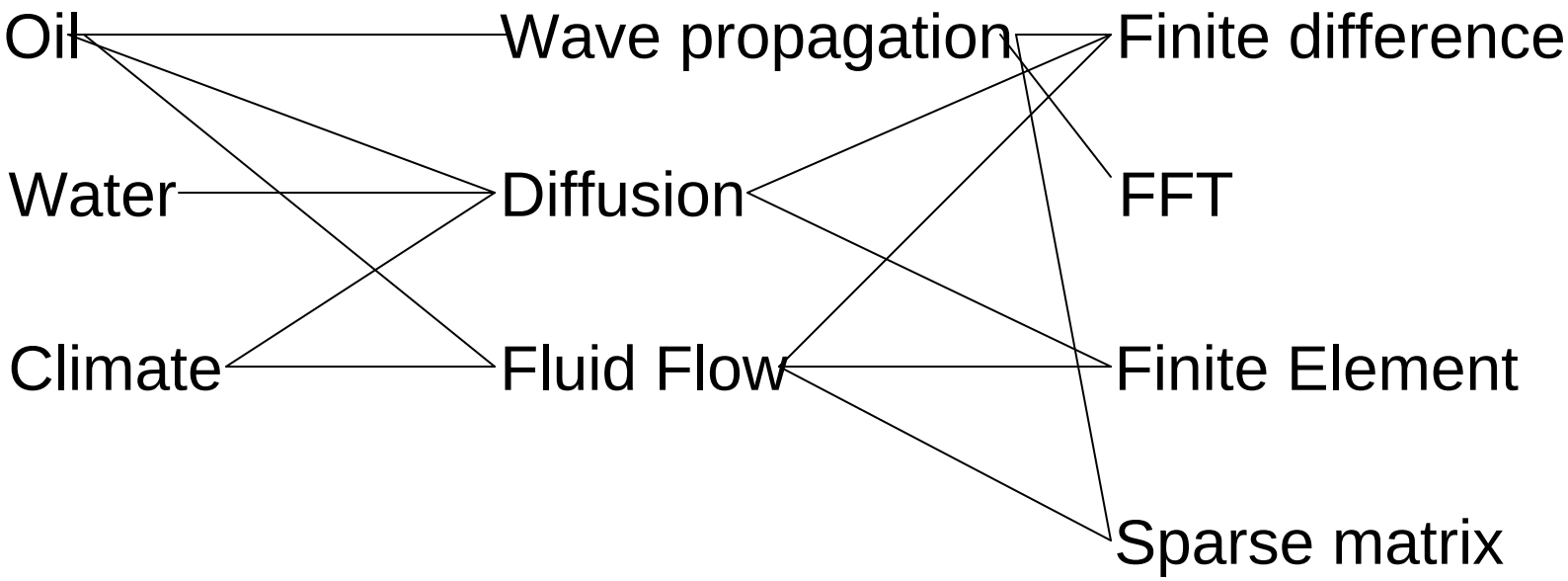
- Meta Data
- Trace: 8 – 20 kB
- Shot: 500 MB – 2 GB
- Survey: 2 – 500 TB

100s – 10,000+ nodes

Run time: days – weeks

Slide courtesy of Olav Lindtjorn

Computational problems in geophysics



Slide courtesy of Bob Clapp

Some results and comments

Seismic forward modelling example

- Data can be interpreted with frequency and amplitude indicates structure, delay indicates depth (z axis).
- Process data to determine location of structures of interest
- Many different ways to process
  - One option:
    simulate wave propagation from source to receivers and compare to recorded data

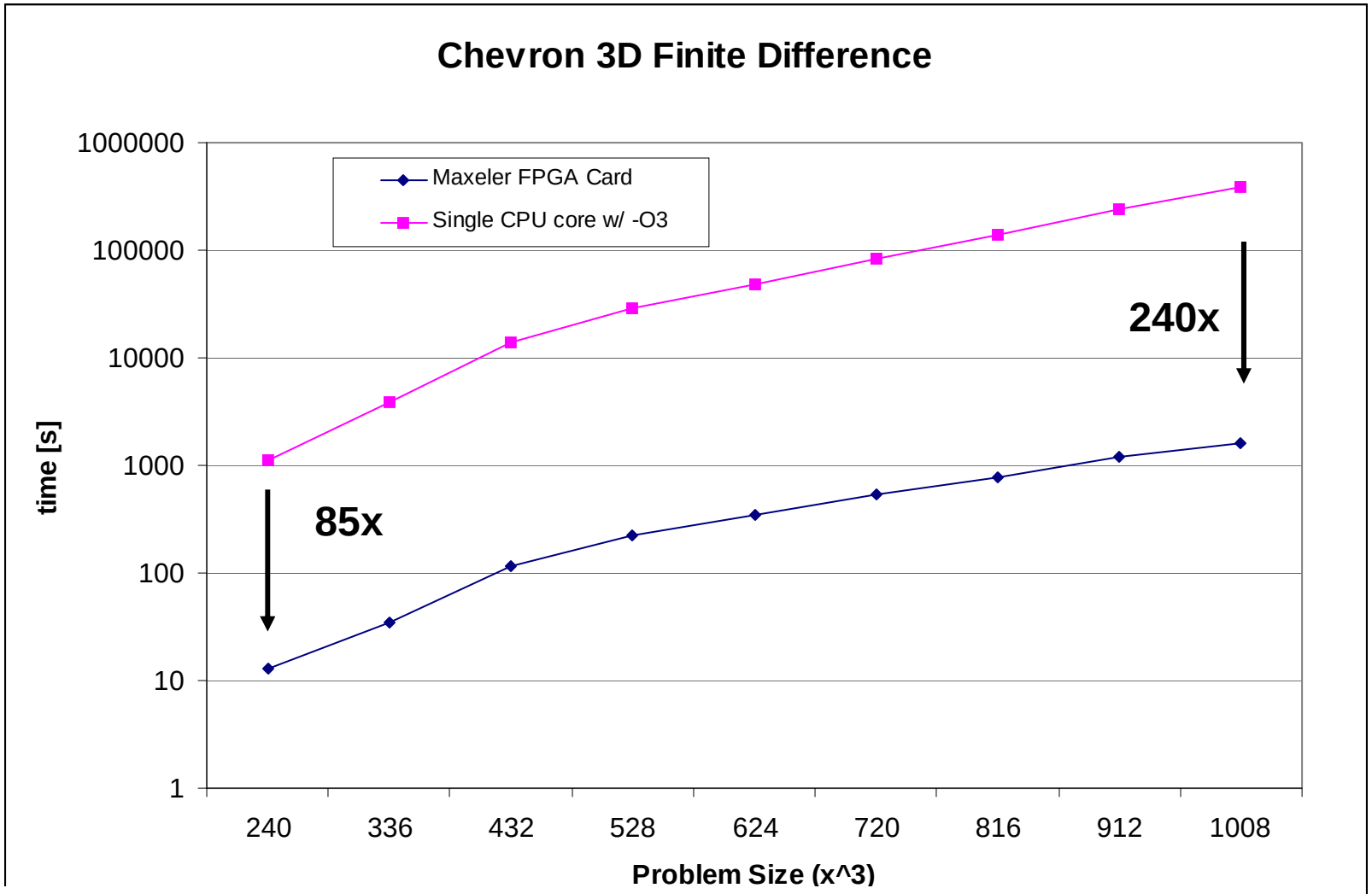
Computations per output point on Intel Xeon

	Cycles	FLOPs	Other Ops	L1 Cache Miss Rate	CPI
2nd order – X pass	11%	40.2	72.3	0.2%	0.6
2nd order – Y pass	15%	40.2	72.3	3.8%	0.8
2nd order – Z pass	21%	40.2	72.4	7.3%	1.1
Vector add	1%	1.0	9.0	1.3%	0.9
4th order – X pass	10%	40.2	64.7	0.2%	0.6
4th order – Y pass	15%	40.2	64.7	4.0%	0.9
4th order – Z pass	21%	39.6	65.3	7.8%	1.2
Update pressure	1%	1.9	9.9	1.0%	0.7
Boundary sponge	5%	0.8	4.0	5.6%	5.8

Computations

- On average about a data cache miss per 10 floating point ops.
- Xeon achieves about 1.0 CPI
- So, Xeon has a 20x frequency starting advantage over an FPGA based computation
- BUT FPGA uses lots of parallelism to significant advantage

Speedup using the Maxeler FPGA System at Chevron

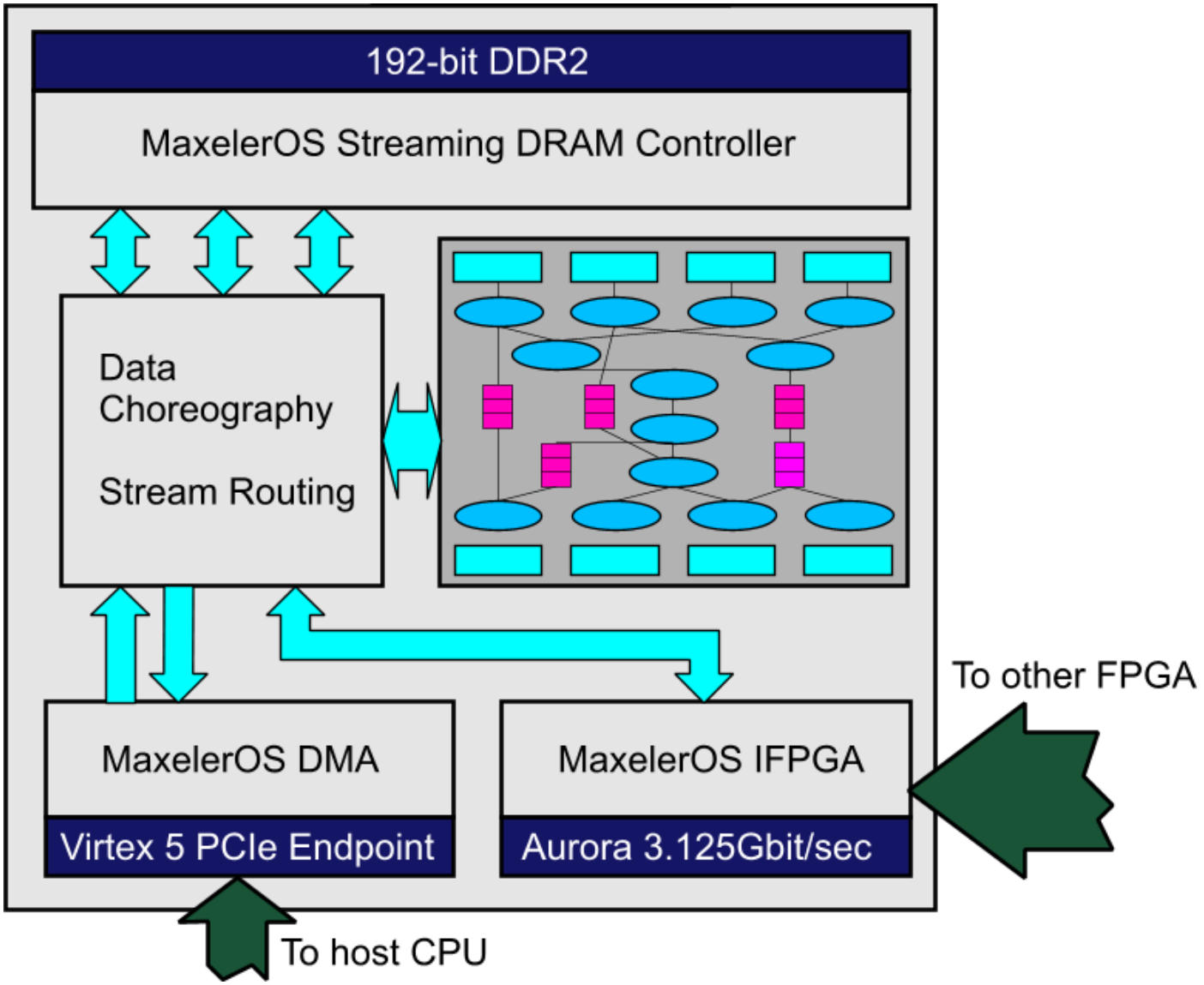


Data from Tamas Nemeth (Chevron), SEG conference, Nov 08, Las Vegas.

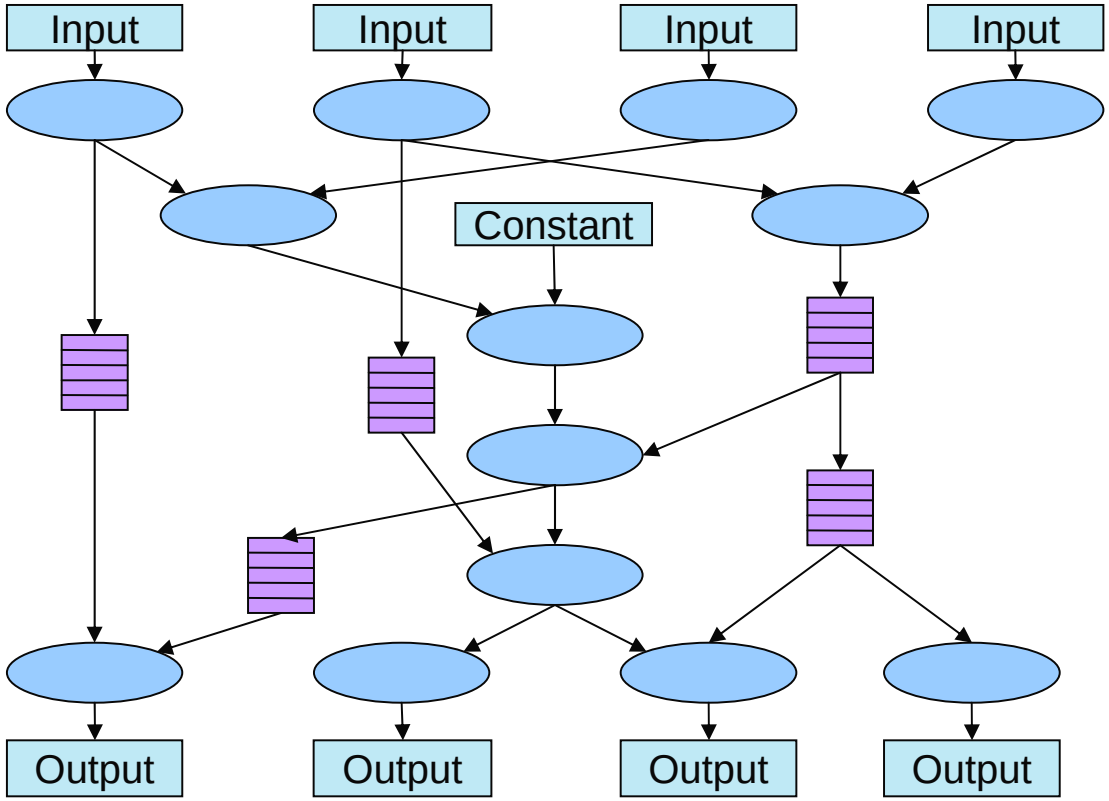
Streaming solution (FPGA)

- Convolve 4 input points (strips) simultaneously (per FPGA); buffer intermediate results; forward 4 outputs to next pipeline stage: SIMD
- Continue (streaming) pipelining until the silicon runs out (468 stages): MISD
- Size the Floating Point so that there is *just* enough range & precision
- One PCIe board provides 8 x 468 FLOPS every 4 ns; almost 1 teraflop.

O. Pell, T. Nemeth, J. Stefani and R. Ergas. *Design Space Analysis for the Acoustic Wave Equation Implementation on FPGA Circuits.* European Association of Geoscientists and Engineers (EAGE) Conference, Rome, June 2008.


192-bit DDR2
MaxelerOS Streaming DRAM Controller
Data
Choreography
Stream Routing
MaxelerOS DMA
MaxelerOS IFPGA
Virtex 5 PCIe Endpoint
Aurora 3.125Gbit/sec
To other FPGA
To host CPU

4 input x 4 output points
no cache misses; no memory accesses



Implements dataflow graph in 468 pipeline stages

Achieving Speedup > 100x

- Stream the computation in 468 stage pipeline
- Execute 8 points simultaneously
- Eliminate cache misses, eliminate overhead operations (load, store, branch..)
- So 8x (points processed) x 468 stages x 2 (overhead ops)/ 20 = 374 (max Sp possible)
- Operate at one-twentieth the frequency; reduce power and space.

So how typical is this?

- Other application areas (ongoing studies)
  - Financial computations, Monte Carlo based derivative price and risk
  - PDE solvers: Finite Difference, Finite Elements
  - Image processing
  - CFD, Lattice Boltzmann
- Kernels, referenced to a Xeon
  - Gaussian Random Number Generation 300x
  - 3D finite difference 160x
  - Seismic offset gathers 170x

So how can emulation technology (FPGA) be better than the x86 processor(s)?

- Lack of robustness in x86 streaming hardware (spanning area, time, power)
- Lack of robust parallel software methodology and tools
- Effort and support tools enable large systolic dataflow with surprisingly good A x T x P
- Effort and support tools provide significant application speedup

So, are we there yet?

- Do we have a design automation technology for parallelizing software?
- Well, no; but Slotnick promised it wouldn't be easy and we're making progress.

there's no free lunch in parallel programming, yet

Summary

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

- Acceleration based speedup using systolic arrays.
- More attention to memory and/or arithmetic: data representation, streaming, and RAM.
- Lower power with non aggressive frequency use.
- Programming uses cylindrical model; but speedup requires lots of low level program optimization. Good tools are golden.