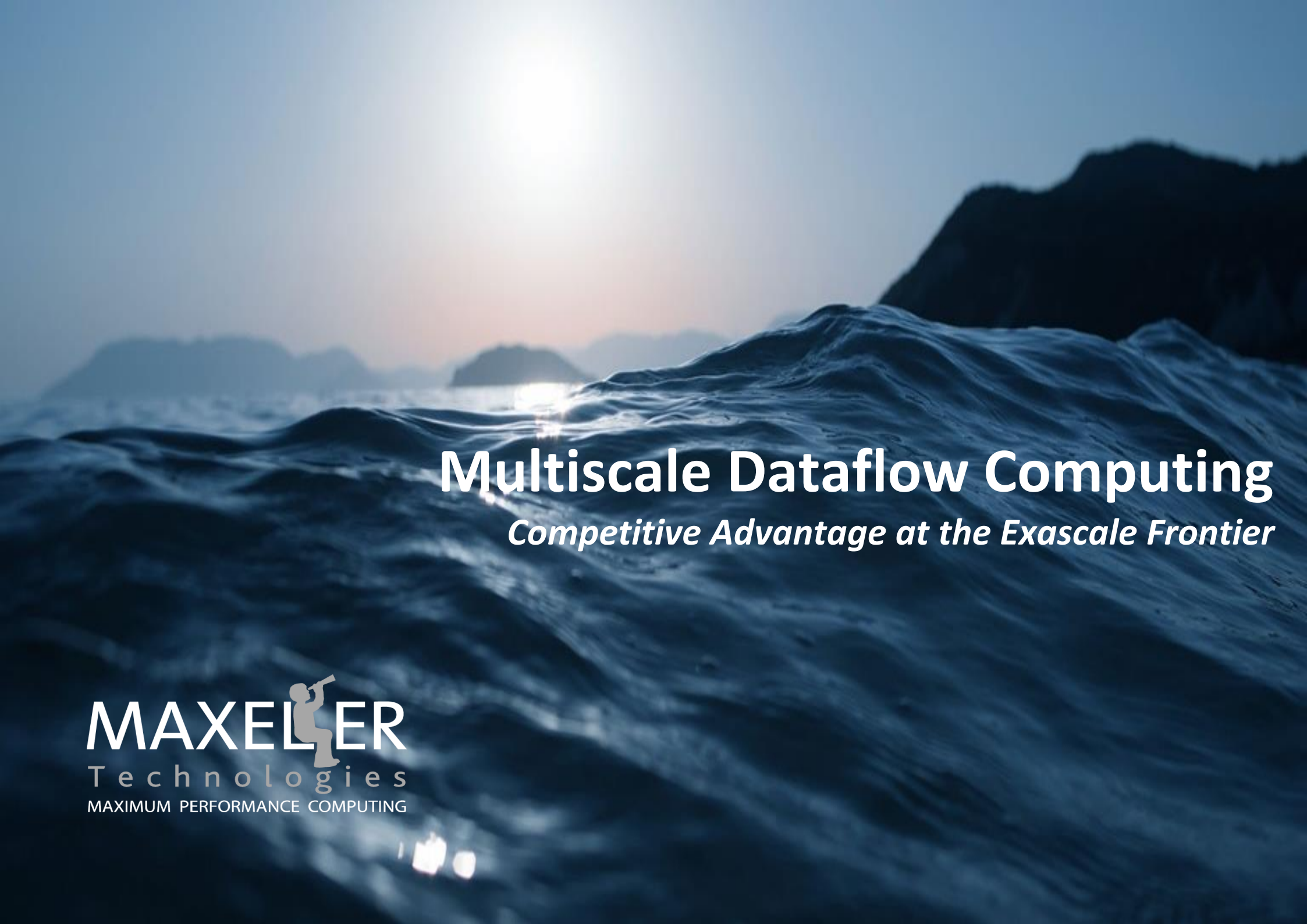




Multiscale Dataflow Computing

Competitive Advantage at the Exascale Frontier

MAXELER
Technologies
MAXIMUM PERFORMANCE COMPUTING

What Makes Computers Inefficient?

A metaphor



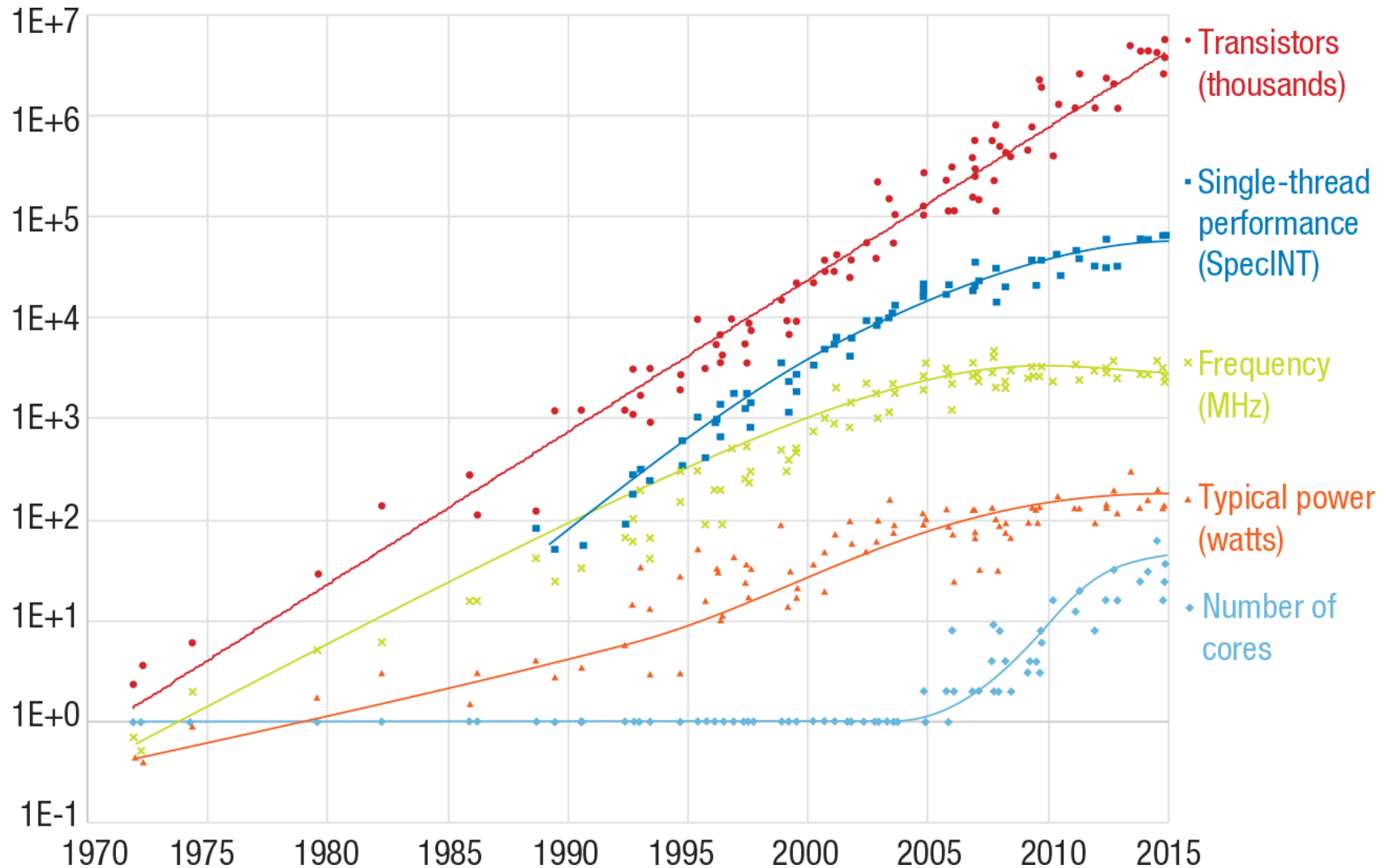
What Makes Computers Inefficient?

A metaphor



The End of Free Performance

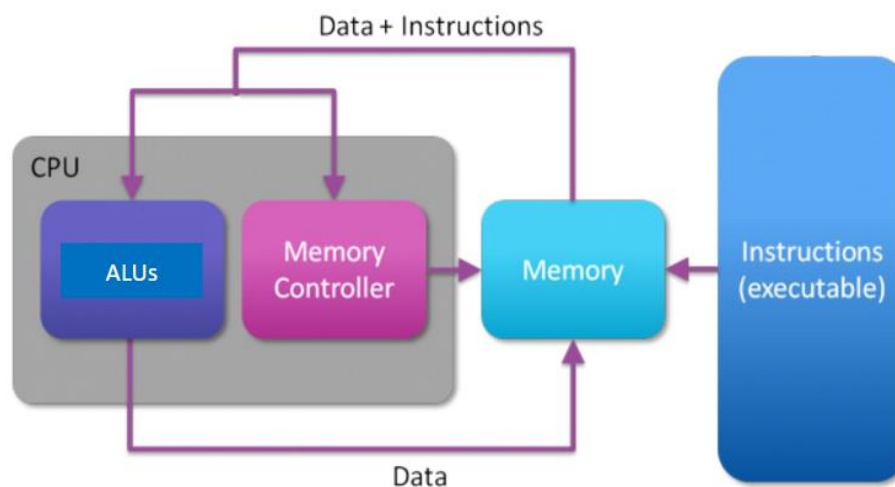
Frequency levels off, cores fill in the gap



The Control Flow Model

General but suboptimal

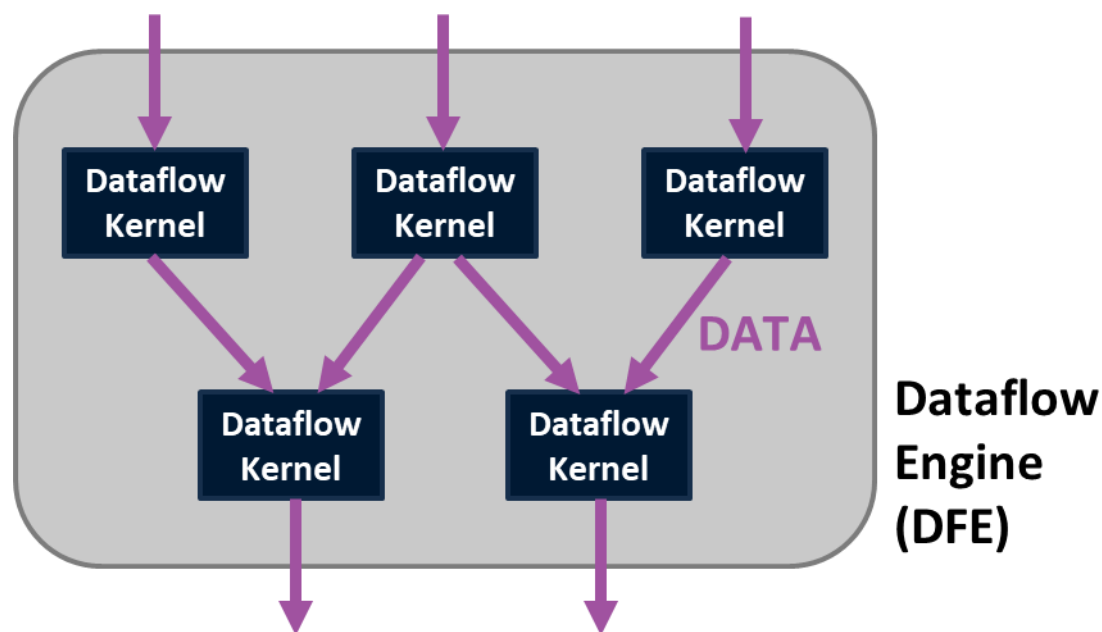
- ◆ Data is static, must be loaded/stored
- ◆ Instructions are data too – compute in *time*
- ◆ *Inefficient* way to solve any problem
 - ◆ Most silicon used to move data, decode instructions etc
- ◆ Inefficient way to solve *any* problem
 - ◆ Software development is fast and easy
 - ◆ Hardware development is difficult and specialized



The Dataflow Model

Build the computer around the problem

- ◆ Data moves continuously
- ◆ Compute in *space* – arrange operations in 2D
- ◆ Optimal solution for a *specific* problem
 - ◆ No wasted silicon – maximum performance density
 - ◆ No wasted clock cycles – predictable speed



The Story of Maxeler Dataflow Computing

Research to real world



- ◆ Researched at **Stanford** pre 2000
 - ◆ Mencer, O. (2000) *Rational Arithmetic in Computer Systems*, (Ph. D. Thesis). Stanford University, California, USA.
- ◆ Refined at **Bell Labs** from 2000 - 2003
 - ◆ Computing Sciences Center, Unit 1127
 - ◆ Birthplace of the transistor, Unix, C, C++ ...
- ◆ Realized via **Maxeler**, founded in 2003
 - ◆ Oil and Gas with **Chevron, ENI, Schlumberger**
 - ◆ Finance with **J.P. Morgan, CME, Citi**
 - ◆ Defense and Cyber Security
 - ◆ Strategic Technology Partnerships
 - ◆ **Juniper, Hitachi, AWS**

Maxeler Success Stories

Dataflow computing provides competitive advantage in multiple industries

◆ Chevron

- ◆ Seismic shoot data must be processed for imaging
- ◆ Maxeler developed dataflow computing to address performance density



◆ JP Morgan

- ◆ Complex credit derivatives
- ◆ Unable to run risk calculations in 2008 crisis
- ◆ Maxeler DFEs reduced run time from 8 hours to 2 minutes

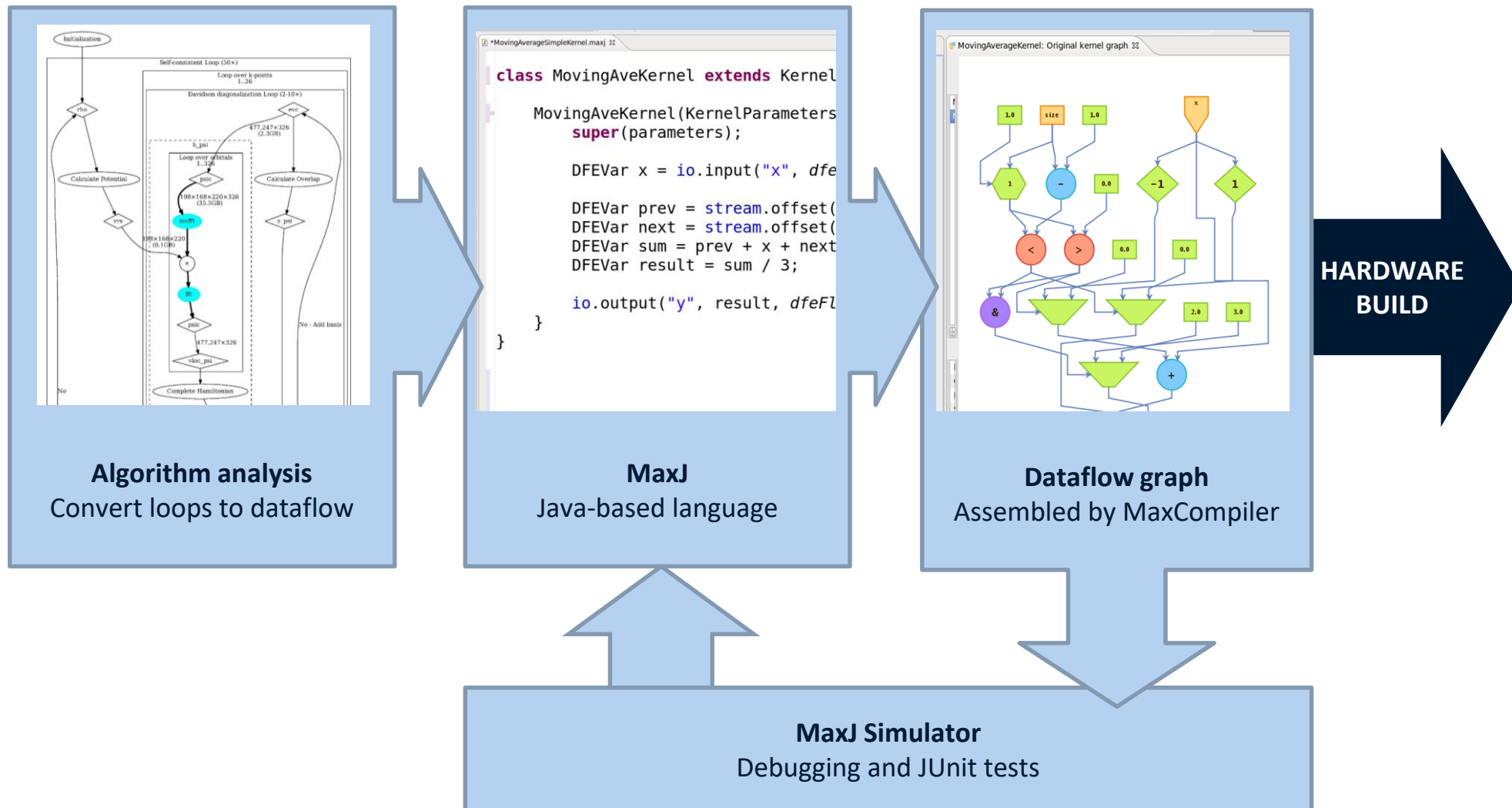
◆ Juniper Networks

- ◆ Added dataflow acceleration to top-of-rack QFX5100 switch
- ◆ Maxeler delivers in-line processing of network data



Building a Dataflow Computer

First, convert the problem to MaxJ



MaxJ

Dataflow computing in a language you know

The screenshot displays the MaxCompiler IDE interface. The main editor shows a MaxJ class definition for `MovingAveKernel`, which extends `Kernel`. The code implements a moving average kernel by offsetting the input stream by -1 and 1, summing the current value with its neighbors, and then dividing the result by 3.

```
class MovingAveKernel extends Kernel {
    MovingAveKernel(KernelParameters parameters) {
        super(parameters);

        DFEVar x = io.input("x", dfeFloat(8, 24));

        DFEVar prev = stream.offset(x, -1);
        DFEVar next = stream.offset(x, 1);
        DFEVar sum = prev + x + next;
        DFEVar result = sum / 3;

        io.output("y", result, dfeFloat(8, 24));
    }
}
```

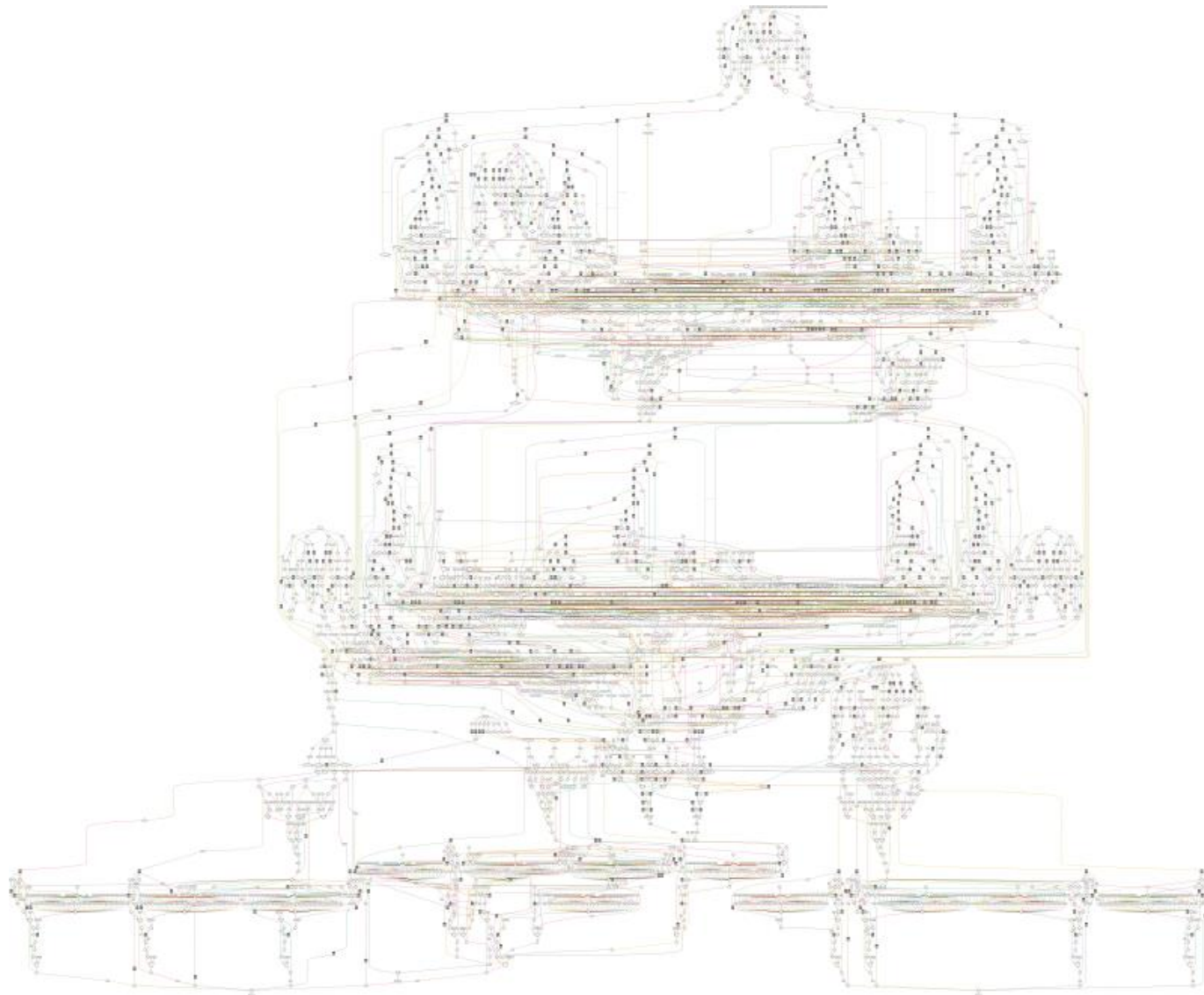
The right-hand pane shows the 'Original kernel graph' for this code. The graph is a dataflow graph with the following components:

- Input:** A yellow trapezoid labeled 'x'.
- Offset Operations:** Two green diamond-shaped nodes labeled '-1' and '1'. The '1' node is highlighted with a red dashed border.
- Addition:** A blue circular node labeled '+' that receives inputs from the 'x' node and the '-1' offset node.
- Division:** A blue circular node labeled '/' that receives inputs from the result of the first addition and a green rectangular node labeled '3.0'.
- Output:** A yellow trapezoid labeled 'y'.
- Control:** A yellow rectangular node labeled 'run_cycle_count' is present in the top right corner of the graph area.

The bottom status bar indicates the simulation is running: 'Simulation (tutorial-chap03-example1-movingaveragesimple)'.

MaxJ

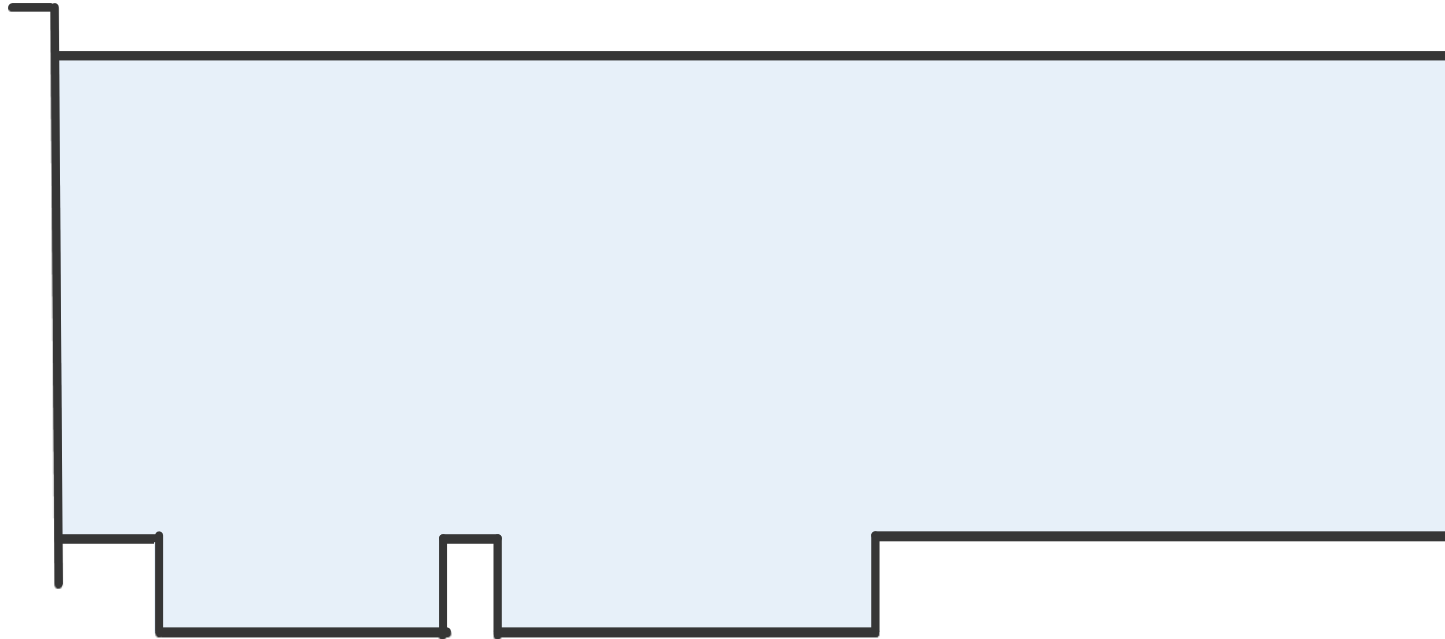
Complex graphs from simple code



3D finite difference time step

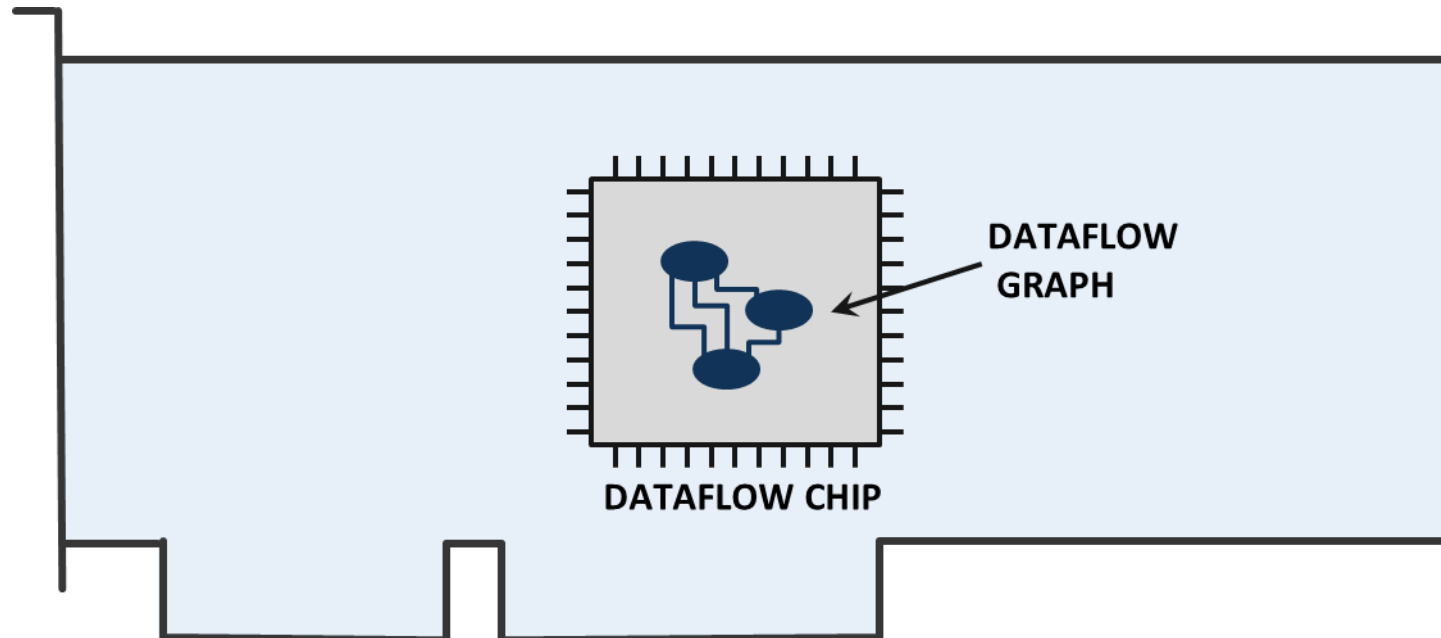
Building a Dataflow Computer

Then build a physical machine



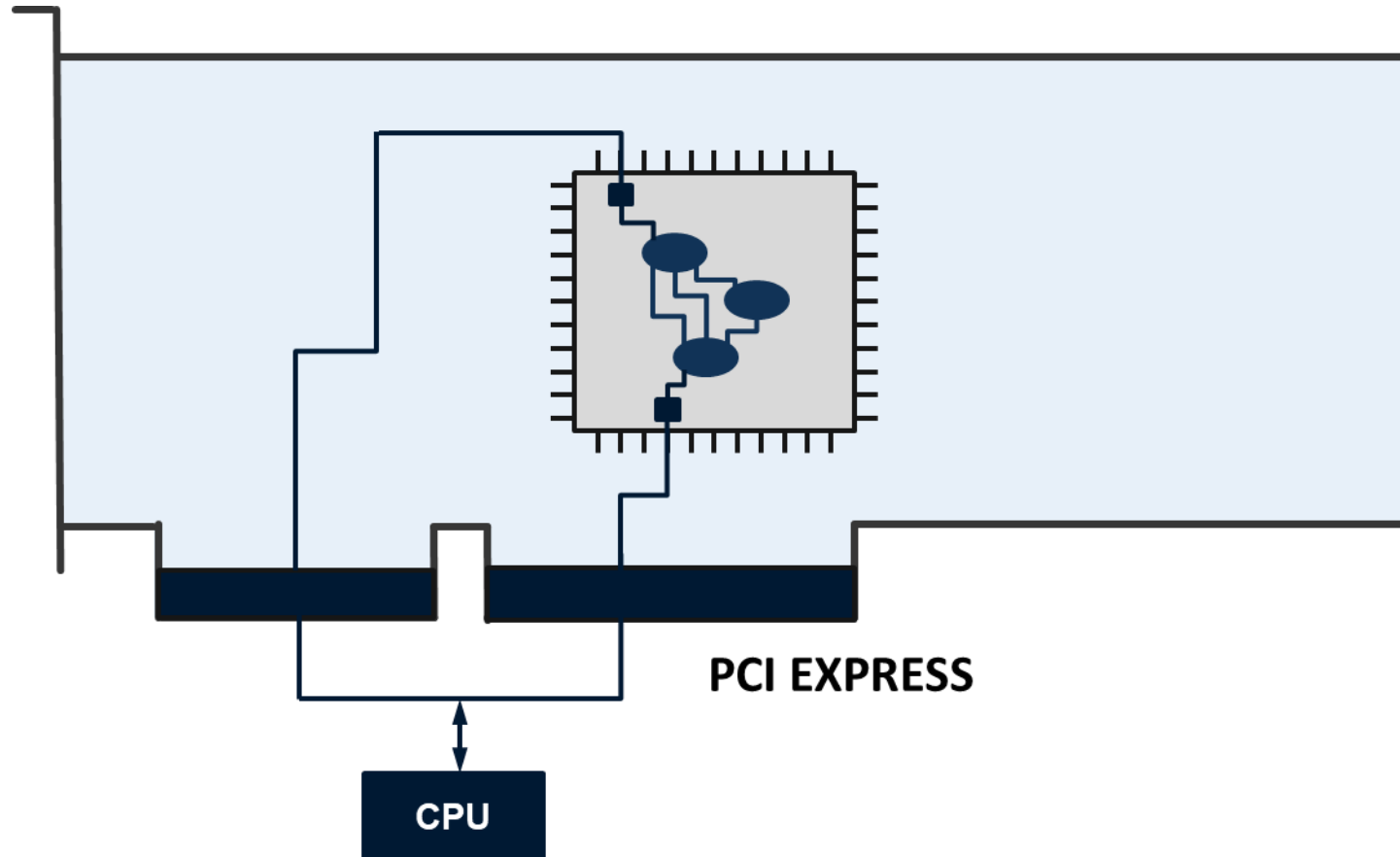
The Dataflow Engine

The dataflow graph as hardware



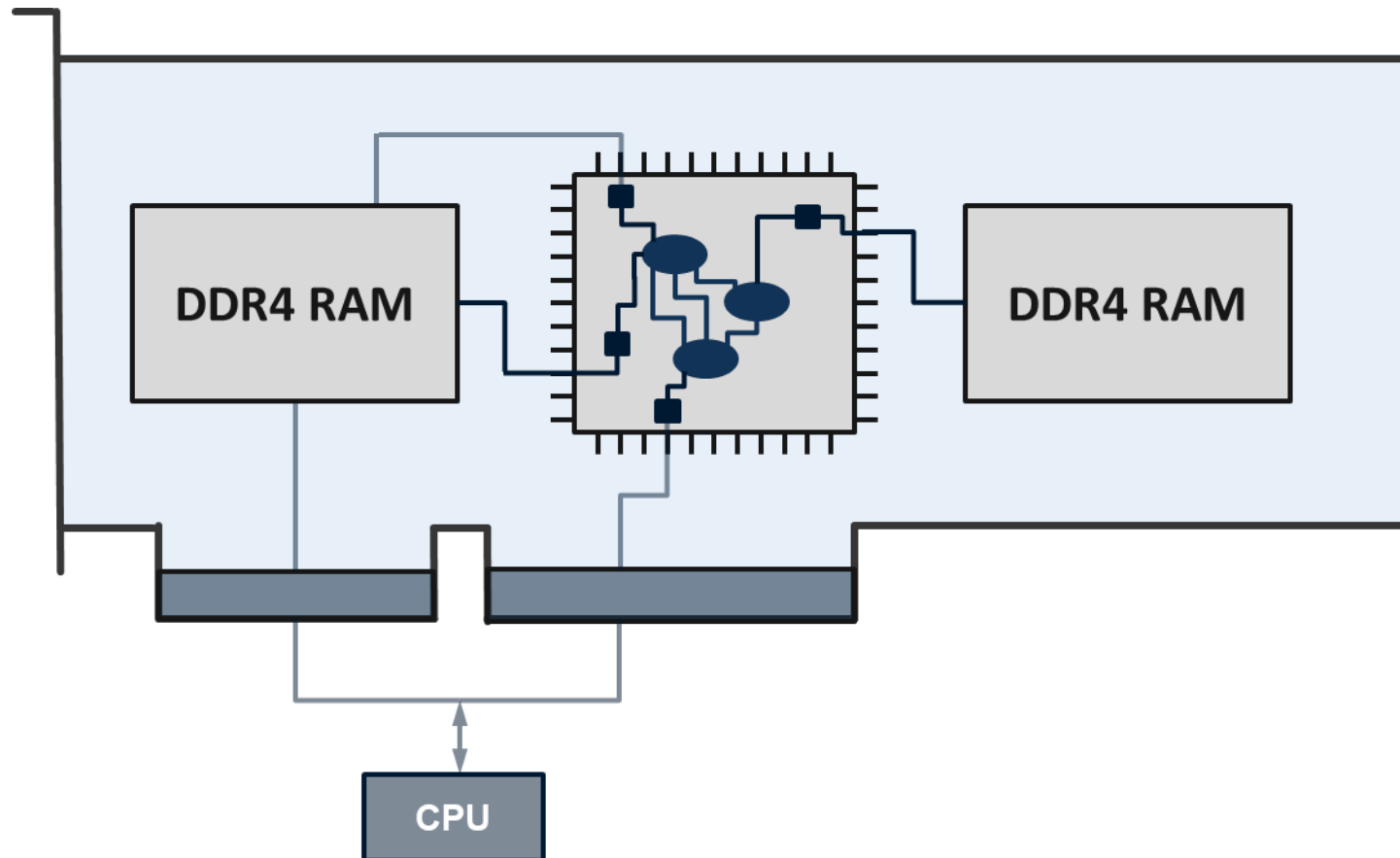
The Dataflow Engine

Communicate with a CPU through PCIe and the MaxelerOS API



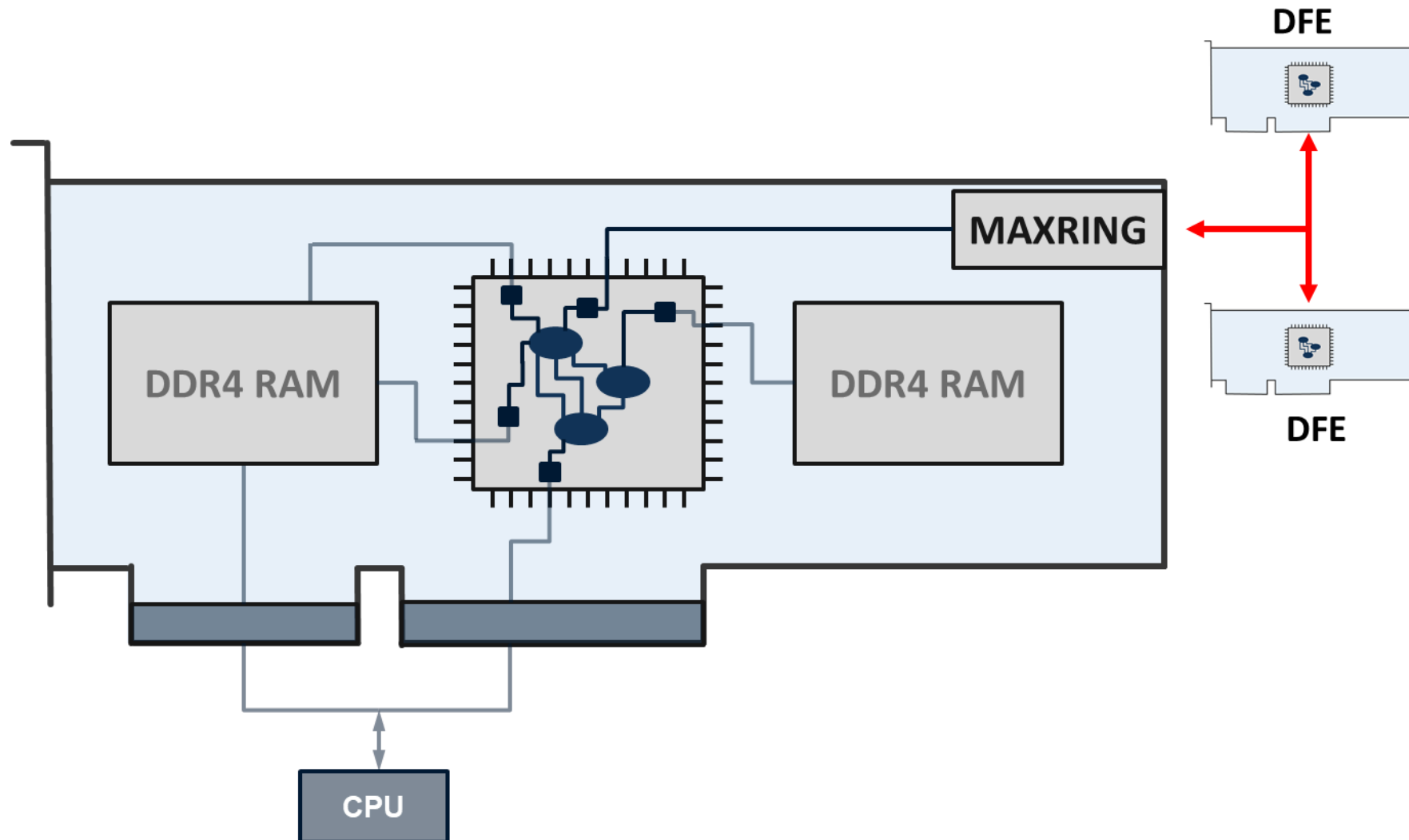
The Dataflow Engine

High-bandwidth connections to large on-card memory



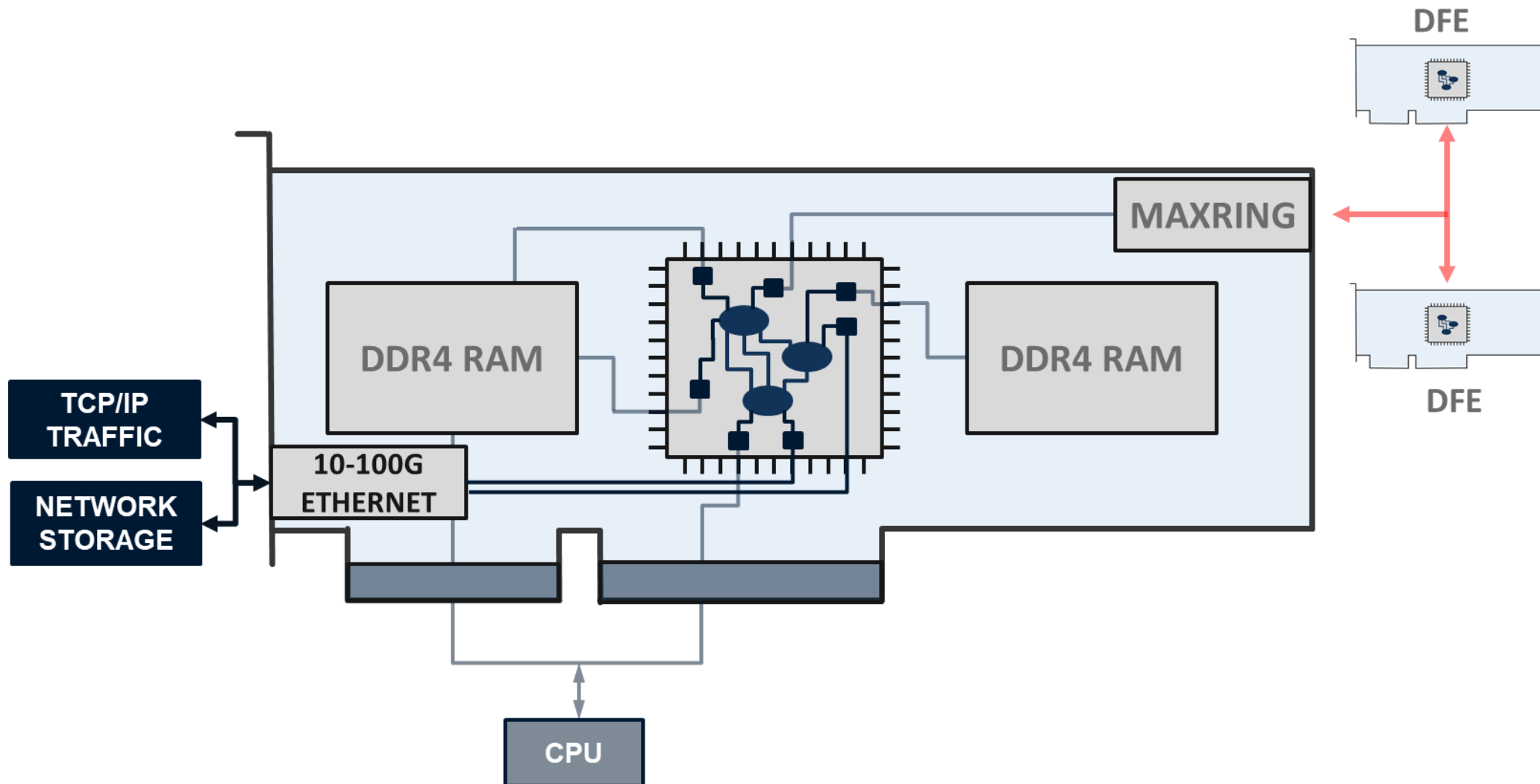
The Dataflow Engine

Two high-speed duplex interconnects to other DFEs through MaxRing



The Dataflow Engine

Optional networking hardware using MaxCompilerNet for frame decoding



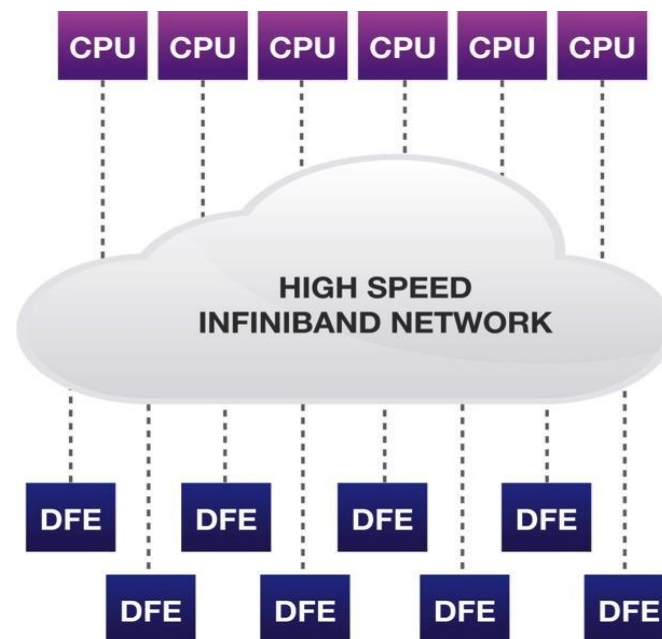
The Maxeler DFE

Dataflow appliance



MPC-X1000

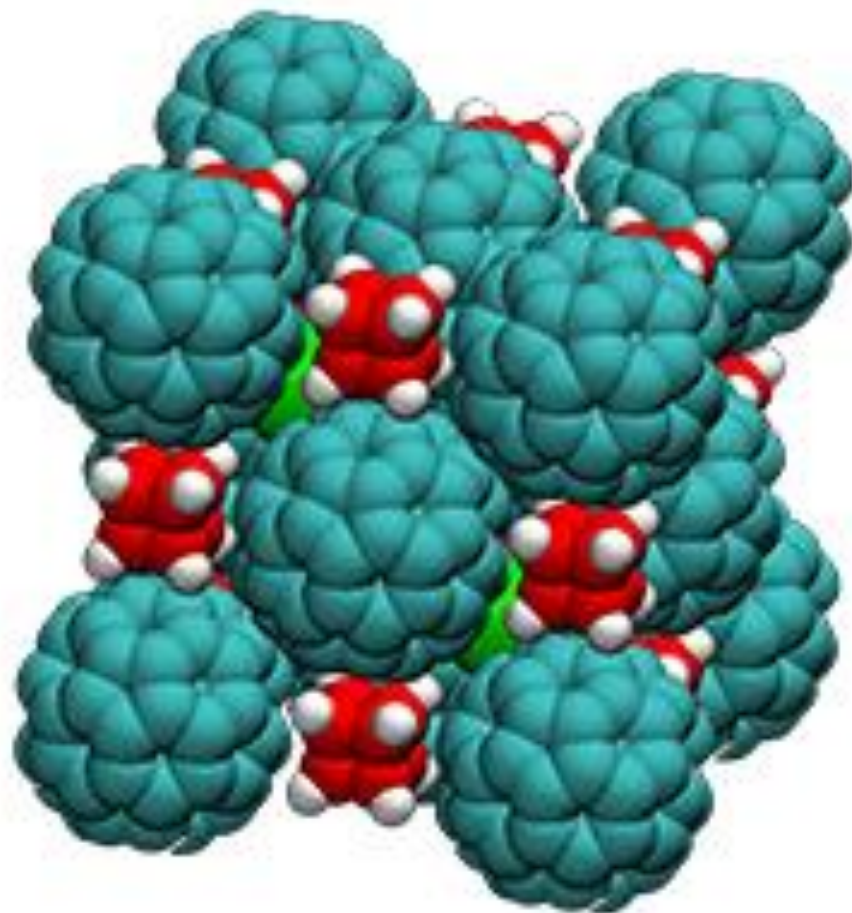
- 8 Dataflow Engines in 1U
- Up to 1 TB of DFE RAM
- Dynamic allocation of DFEs to conventional CPU servers through Infiniband
- Equivalent performance to 20-50 x86 servers



Dataflow Case Study

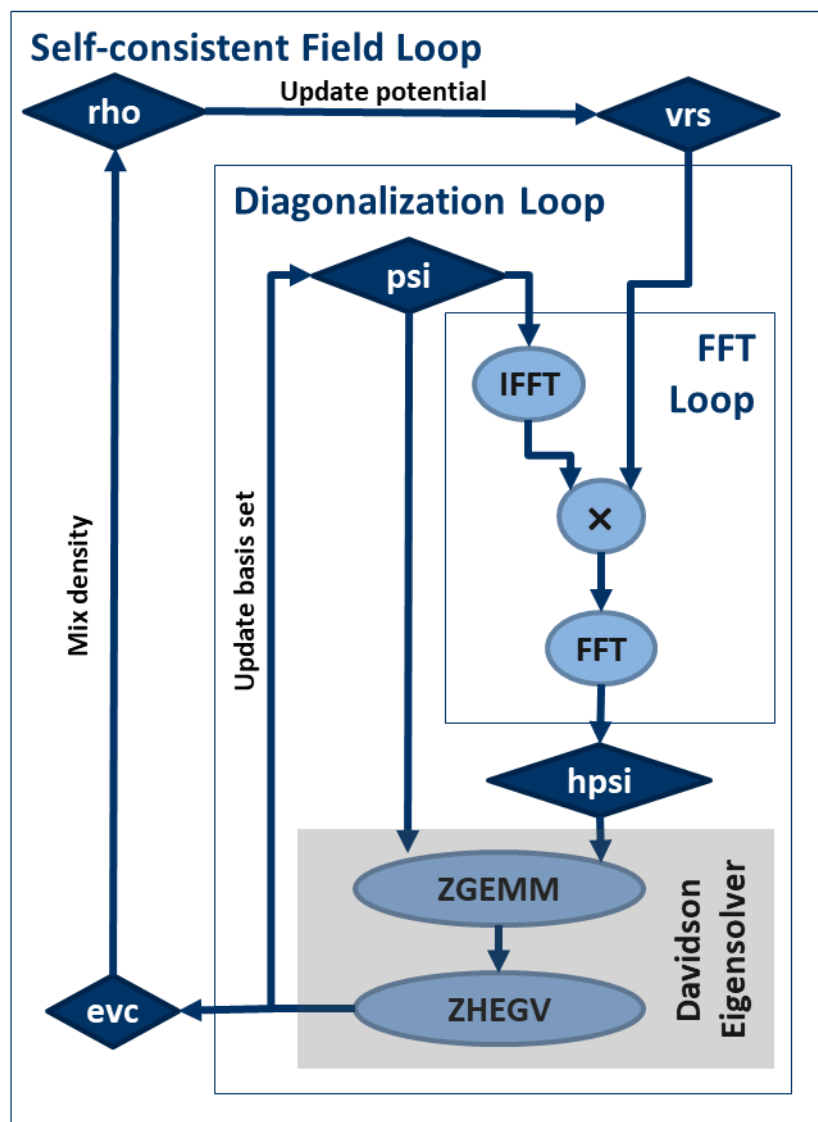
Quantum ESPRESSO

- ◆ FORTRAN software package for
 - ◆ Ab initio quantum chemistry
 - ◆ Materials modeling
- ◆ Iterative solve with FFTs and linear algebra (BLAS etc)
- ◆ Reference system – Ta_2O_5
 - ◆ Two racks of BlueGene/Q
 - ◆ 6.7 m³ of space
 - ◆ 32,768 cores
 - ◆ 53m wall time
 - ◆ 384 kW (25% cooling)



Loopflow Graph

Focus profiling on loop structure, not function calls

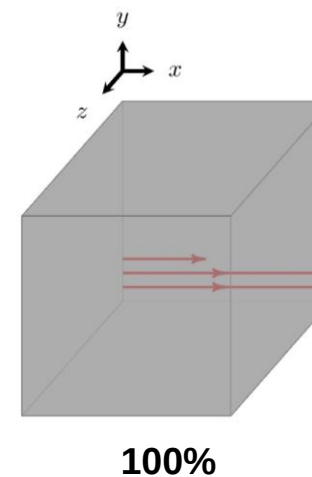
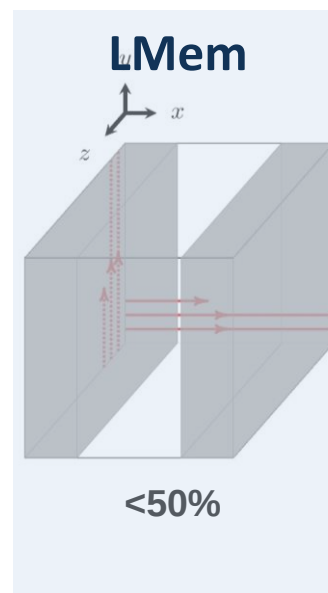
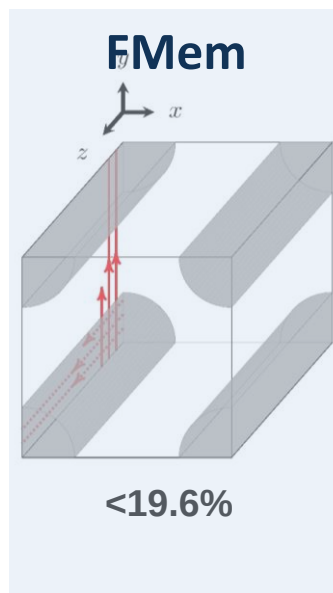
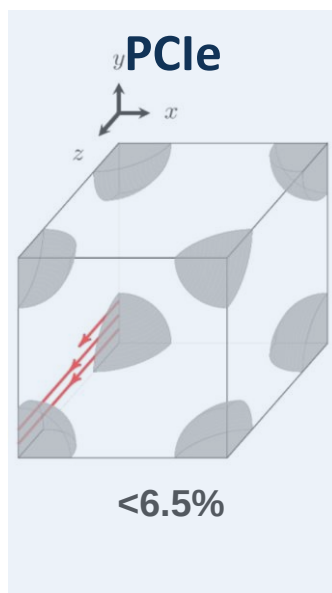


- ◆ Function calls are control flow concept
 - ◆ Jump to another point in instruction data
 - ◆ Reusable logic, independent of calling order
 - ◆ Most profiling tools focus on function calls
- ◆ For dataflow, map out major **loops**
 - ◆ Dataflow engines have an *implicit* outer loop
 - ◆ Measure rates of data flowing in and out
 - ◆ Compare to volume of transient data generated internally
- ◆ QE case study
 - ◆ Typical FFT loops over 5GB **psi** input data
 - ◆ Input **vrs** is 128MB, changes rarely
 - ◆ Equivalent internal memory is 250GB
 - ◆ Control flow – break into small batches
 - ◆ Dataflow – run single streaming action

Optimize Memory

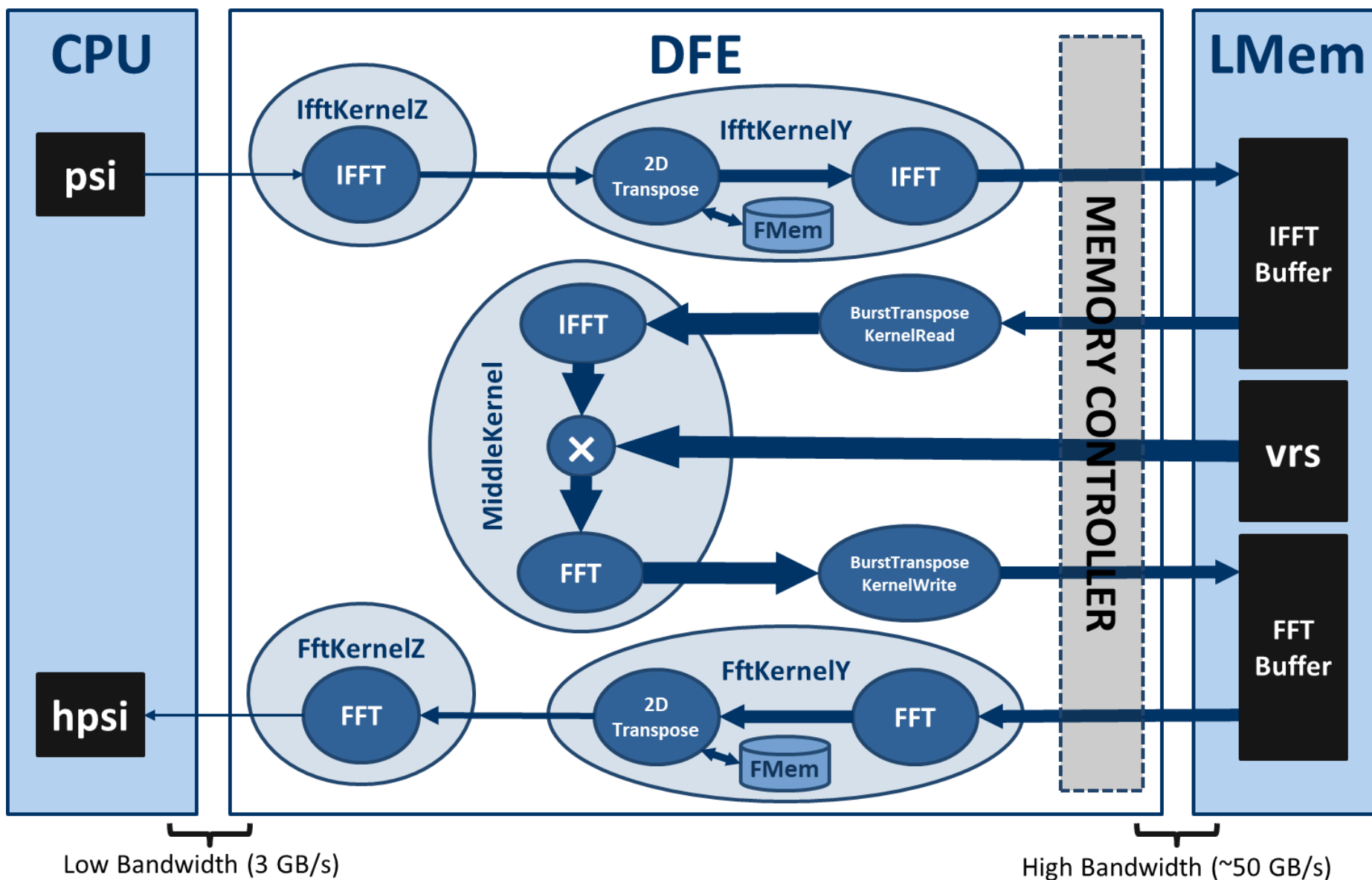
Identify data sizes to layout dataflow architecture

- ◆ Two types of memory:
 - ◆ **FMem** is *fast* and local to the chip – up to 40MB accessed every clock cycle
 - ◆ **LMem** is *large* on-board memory up to 96GB
- ◆ QE case study
 - ◆ Use FMem for 2D transposes (one plane is 0.5MB)
 - ◆ Use LMem for 3D transposes (one cube is 128MB)
 - ◆ Need to move 10x more data over LMem bandwidth than PCIe bandwidth



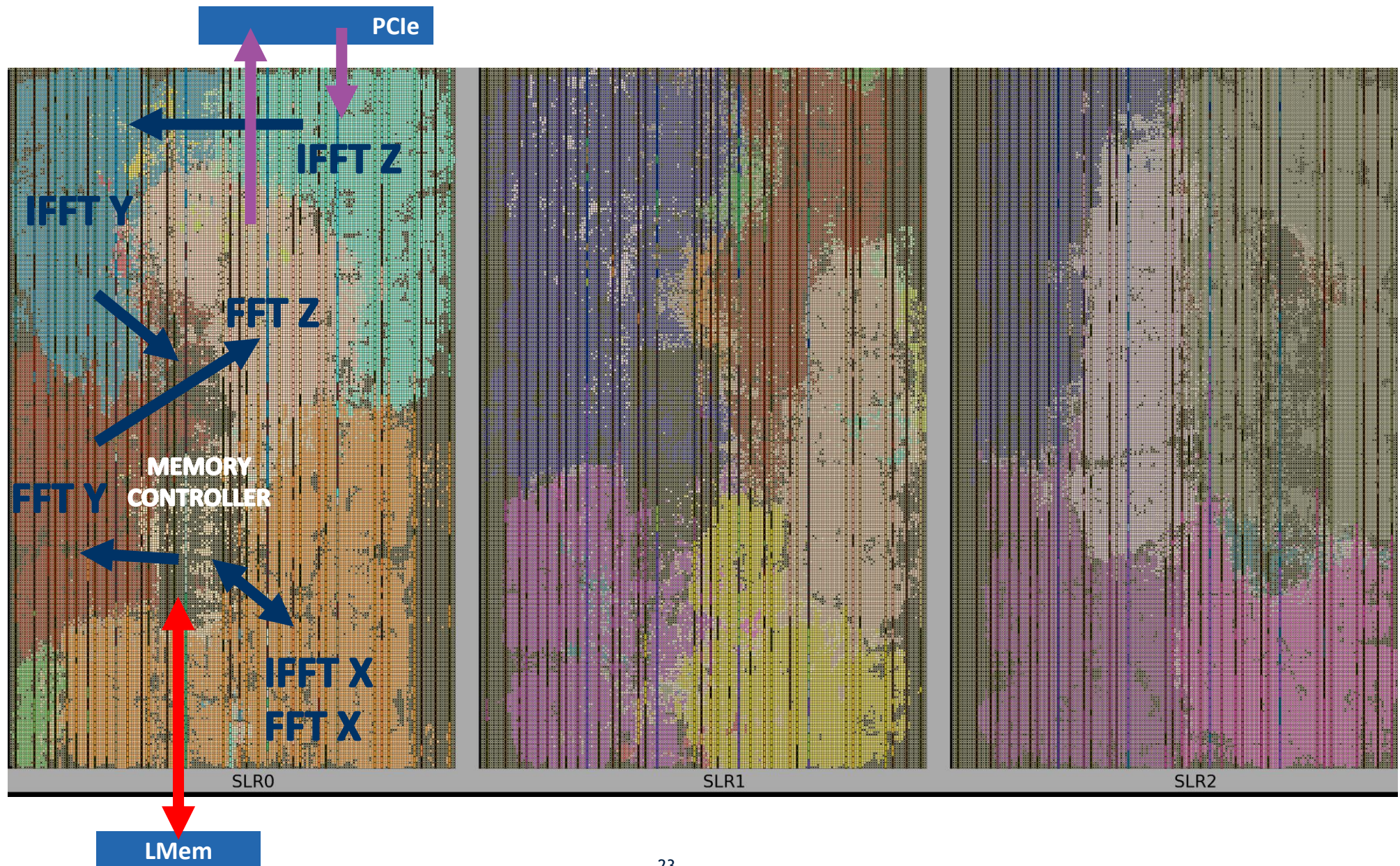
Dataflow Architecture

Match dataflows to available capacities and bandwidths



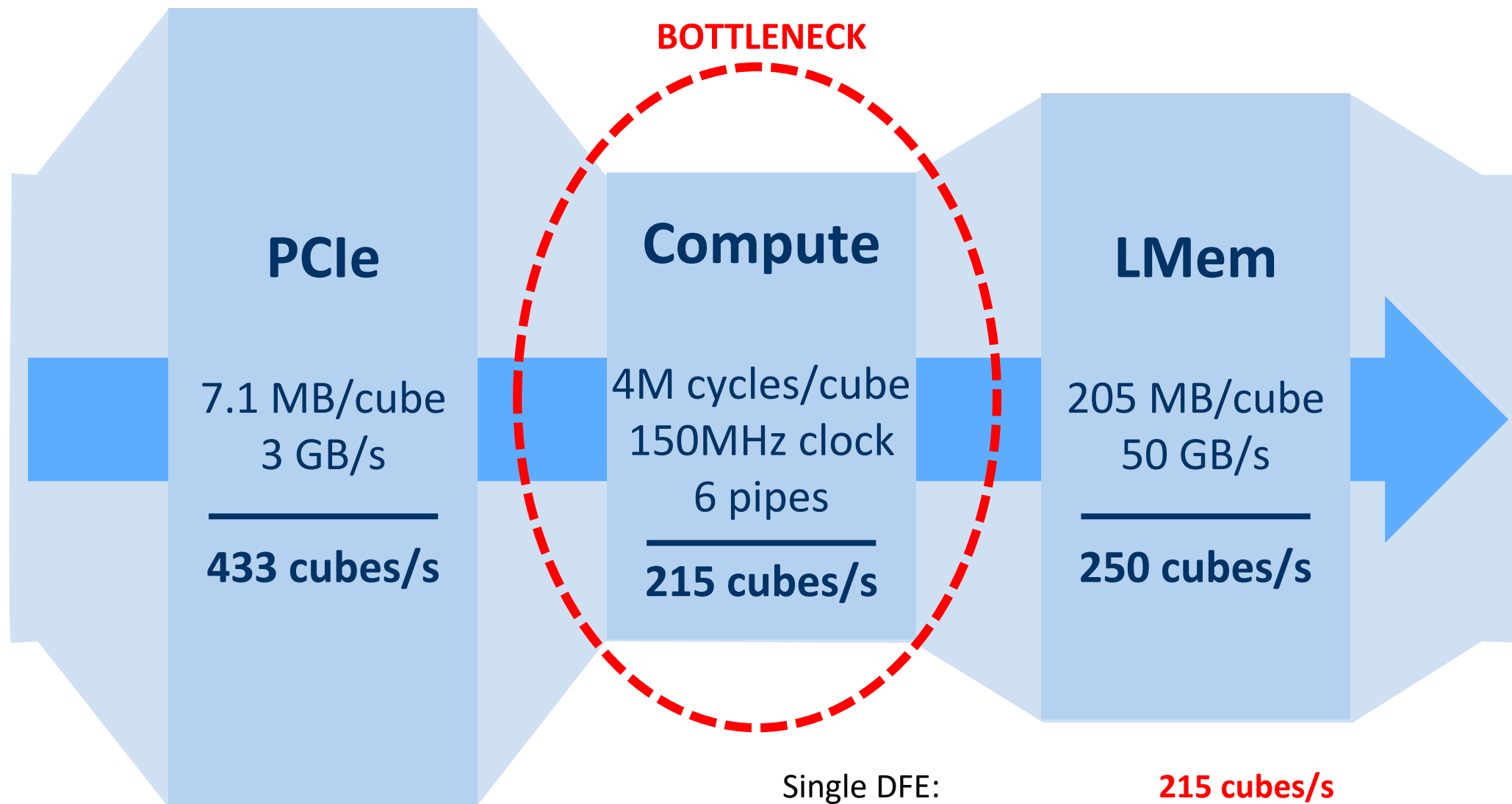
Computing in Space

Fill up the chip for maximum performance



Performance Modeling

Simple arithmetic without guess work of cache, OS, etc



Single DFE: **215 cubes/s**
One rack of BlueGene/Q: 337 cubes/s

Performance Modeling

Comparison to reference system



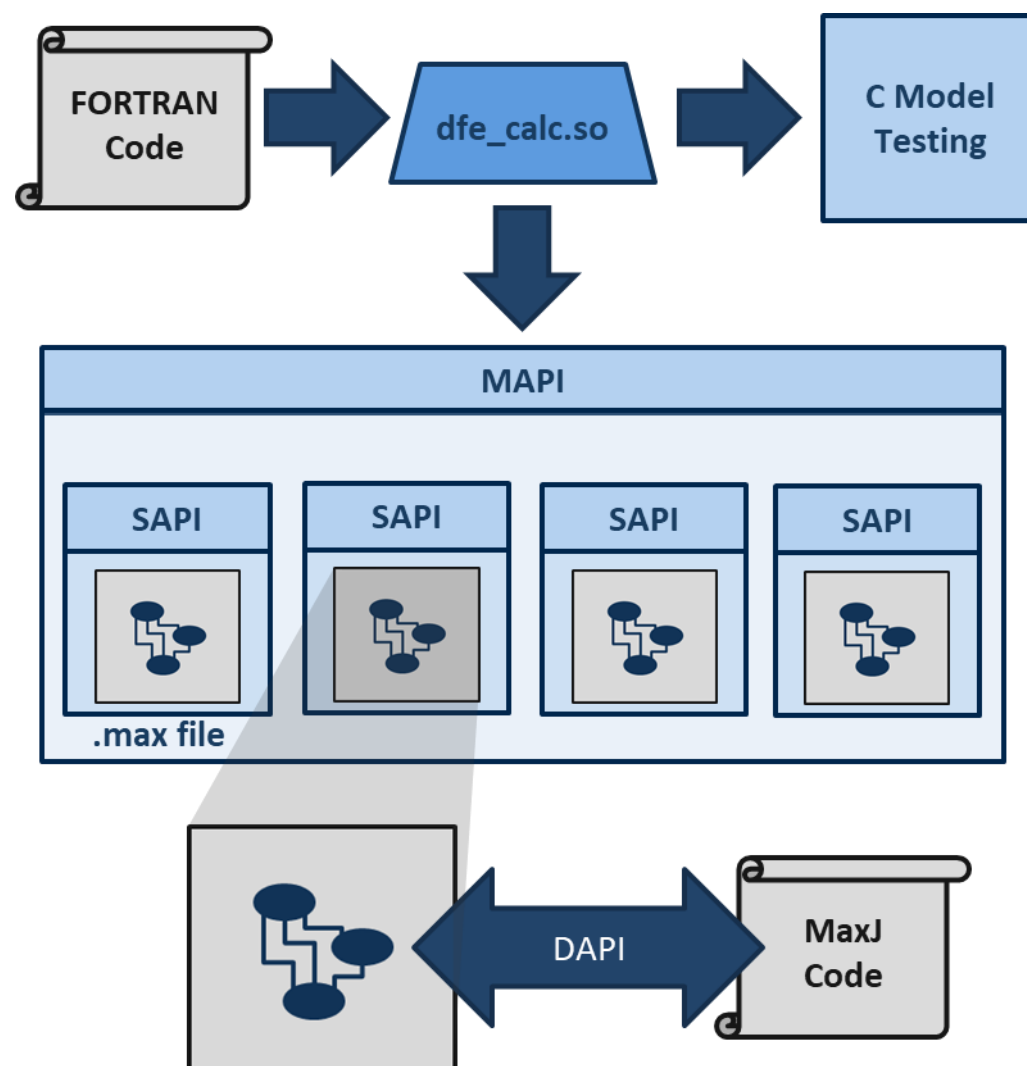
System	1 rack of BlueGene/Q	Maxeler MPC-X 1U with 8 MAX5 DFEs	Comparison
Space	3.374 m ³	0.025 m ³	135x
Power	192 kW	1 kW	192x
Performance	338 cubes/s	1716 cubes/s	5.1x

- ◆ BlueGene/Q contains significant water cooling and communication – FFT divided to 256 nodes
- ◆ Maxeler MPC-X is air-cooled, optically connected internally – FFT in a single node
- ◆ Overall **700x** improvement in compute/space and **1000x** improvement in compute/power
 - ◆ These are for the FFT task *only* – but a proper phase 2 architecture should scale them up to the full model

Code Integration

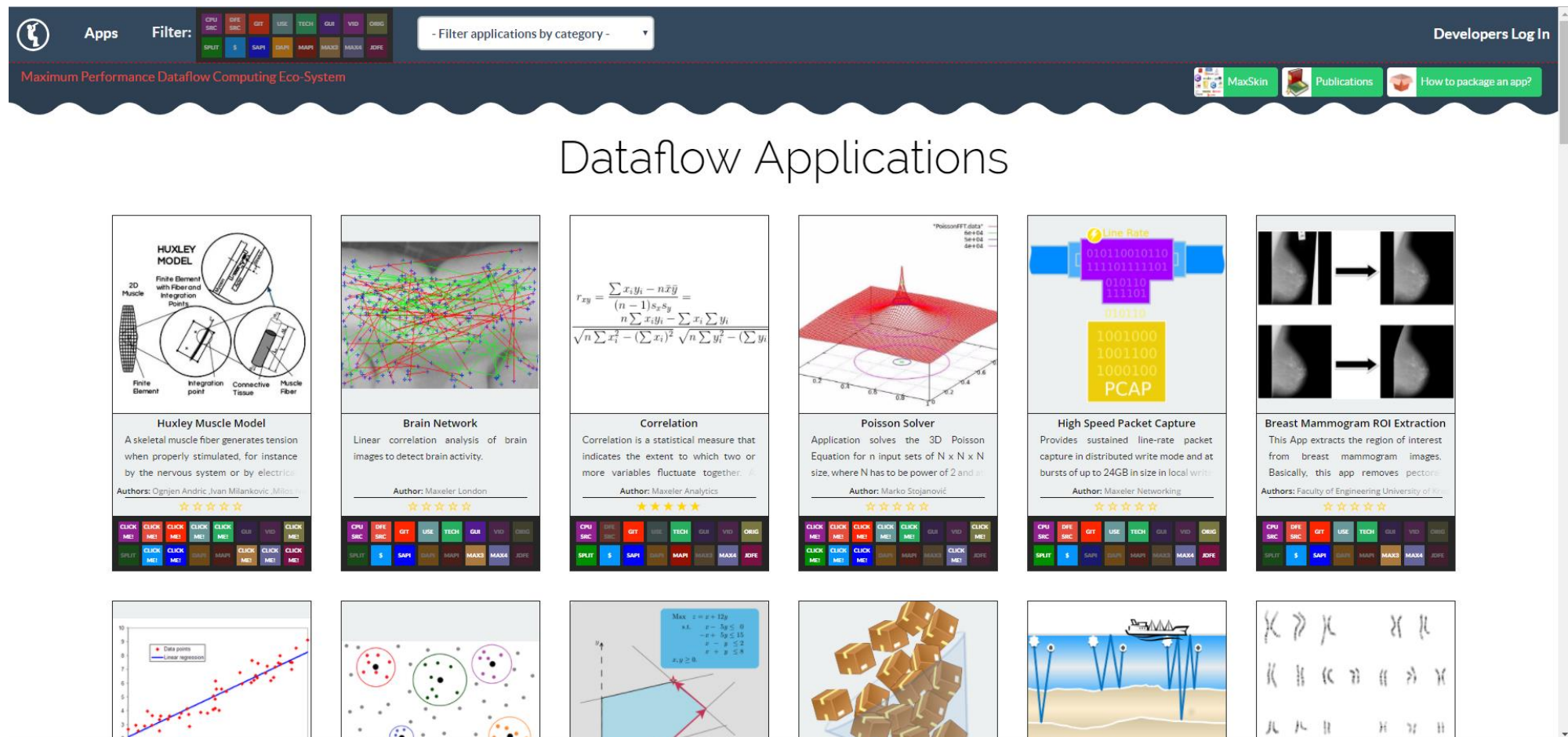
APIs at multiple levels

- ◆ SAPI – Single DFE
 - ◆ Simple Live CPU (SLiC) interface
 - ◆ Non-blocking actions
 - ◆ Portable shared-object file
- ◆ MAPI – Multiple DFEs
 - ◆ Partition problem space
 - ◆ Allocate engines dynamically
- ◆ DAPI – Device API
 - ◆ Interact with pre-built MaxJ logic
 - ◆ Reconfigure an existing dataflow solution for a new problem



AppGallery

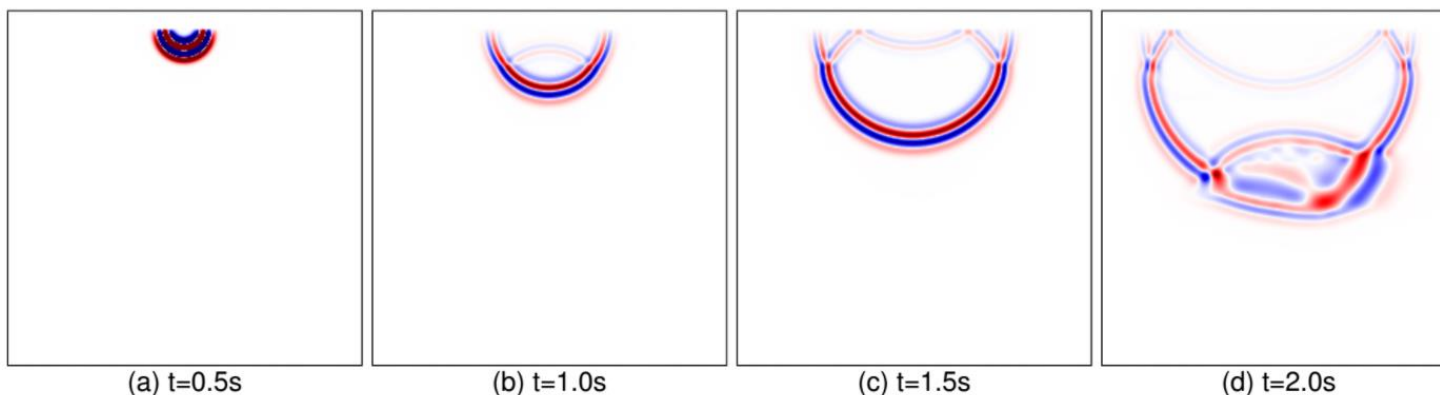
Largest collection of dataflow applications



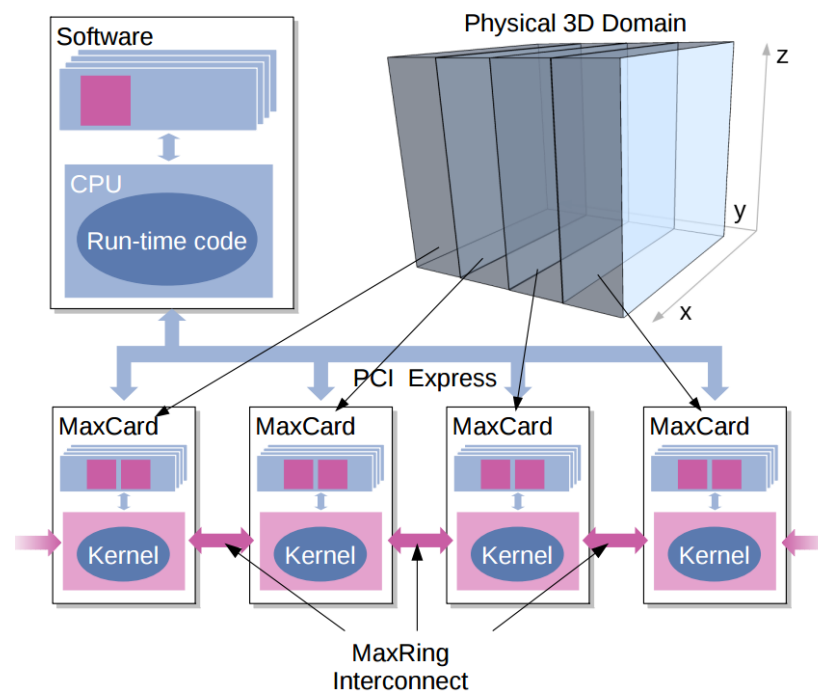
<http://appgallery.maxeler.com/#/>

MaxGenFD

Purpose-built finite difference suite for dataflow computing



- ◆ Developed to serve energy industry
 - ◆ Finite-difference in 3D
 - ◆ Seismic study modeling
- ◆ Layer over MaxJ/MaxCompiler
 - ◆ Science user codes FD equations in Java
 - ◆ Domain decomposition
 - ◆ Sharing of halo through MaxRing
 - ◆ Minimal dataflow knowledge required



Proven Performance

An order of magnitude improvement over a leading supercomputer

- ◆ Gan, L., Fu, H., Luk, W., Yang, C., Xue, W., Huang, X., et al. (2015, April). Solving the Global Atmospheric Equations through Heterogeneous Reconfigurable Platforms. *ACM Transactions on Reconfigurable Technology and Systems*, 8(2)
- ◆ Joint research with Imperial College and Tsinghua University
- ◆ Simulating the atmosphere using the shallow water equation

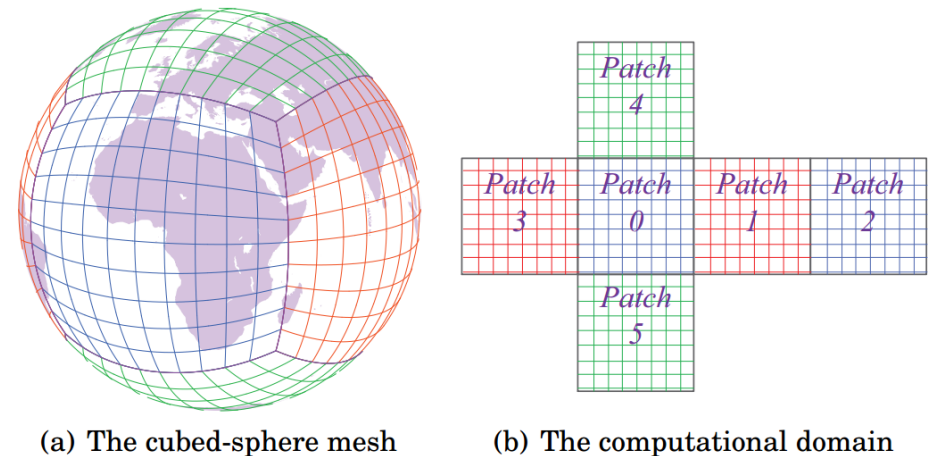


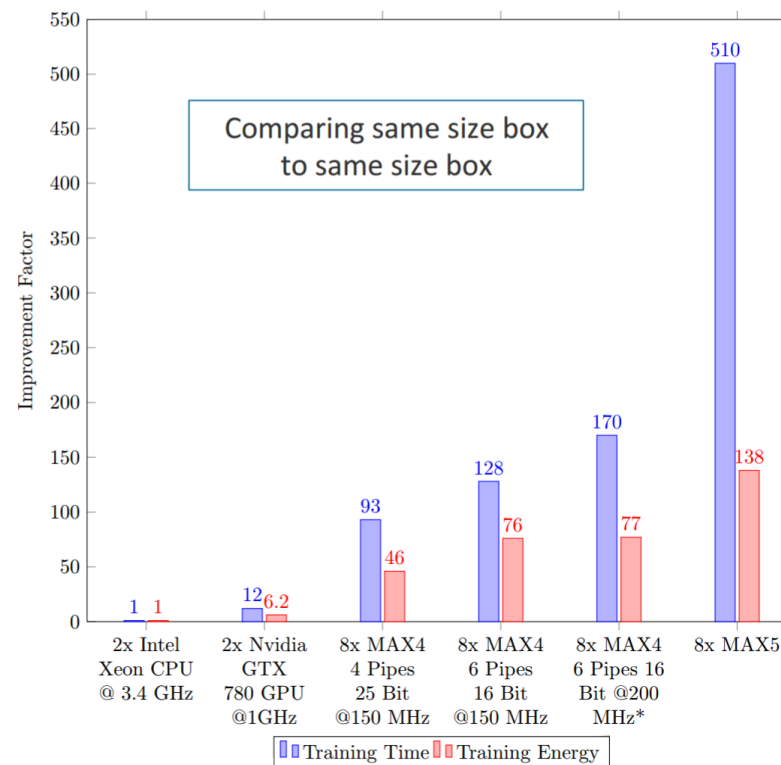
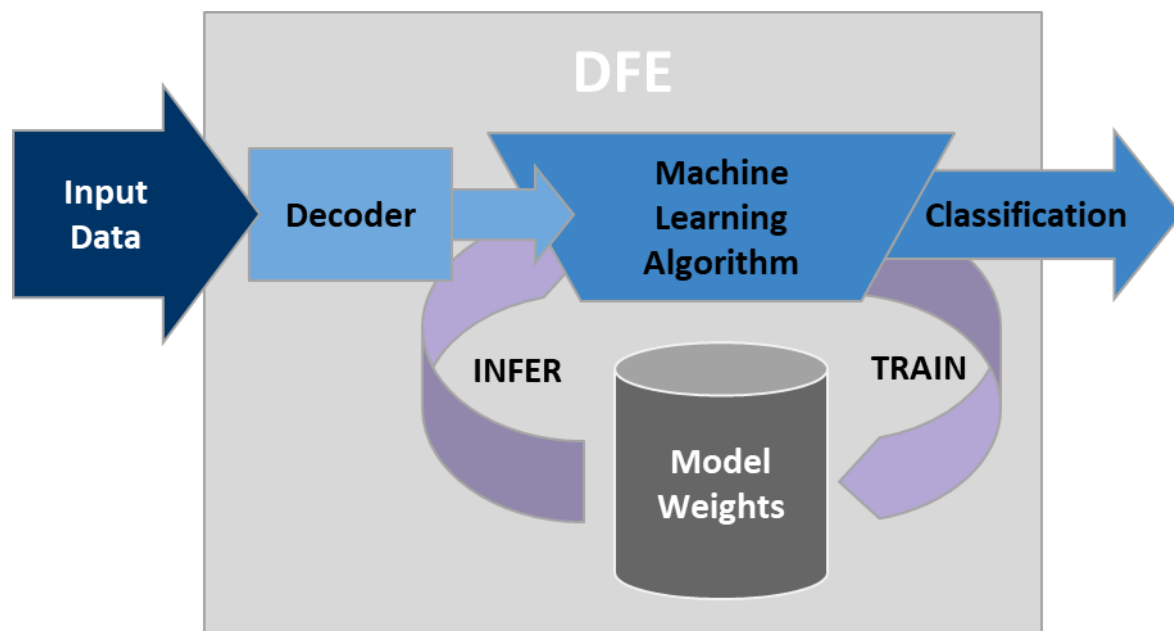
Fig. 1. Mesh and computational domain.

Platform	Processor	Points/s	Speedup	Power (W)	Efficiency
CPU Rack	2xCPU	82K	1x	377	1x
Tianhe-1A Node	2xCPU + Fermi GPU	110.4K	1.4x	360	1.4x
Kepler K20x	2xCPU + Kepler GPU	468.1K	2.6x	365	2.6x
Maxeler MPC-X	4xDFE	1.54M	19.4x	514	14.2x

MaxML for Machine Learning

Order of magnitude improvements in training and inference

- ◆ Machine learning on DFEs uses large-capacity memory and in-line training updates
- ◆ Support for convolutional and fully connected layers
- ◆ Choose the exact precision you need for maximum performance



Questions?

What can dataflow programming accelerate for you?