20 Years of Static Dataflow

Oskar Mencer
Founder
Maxeler Technologies

Nov 2021

2001

Computer Systems Laboratory Colloquium

4:15PM, Wednesday, May 2nd, 2001
NEC Auditorium, Gates Computer Science Building B03

Computing with FPGAs

Oskar Mencer
Lucent / Bell Labs, and Imperial College, London

About the talk:

Field-Programmable Gate Arrays (FPGAs) can outperform microprocessors on certain tasks by many orders of magnitude. The open research problems of computing with FPGAs are: (1) understanding the limitations of FPGAs when competing with microprocessors, and (2) providing a useful programming methodology.

First, I will show how FPGAs can be utilized to accelerate certain algorithms by up to three orders of magnitude. Examples for methods achieving the speedups are:

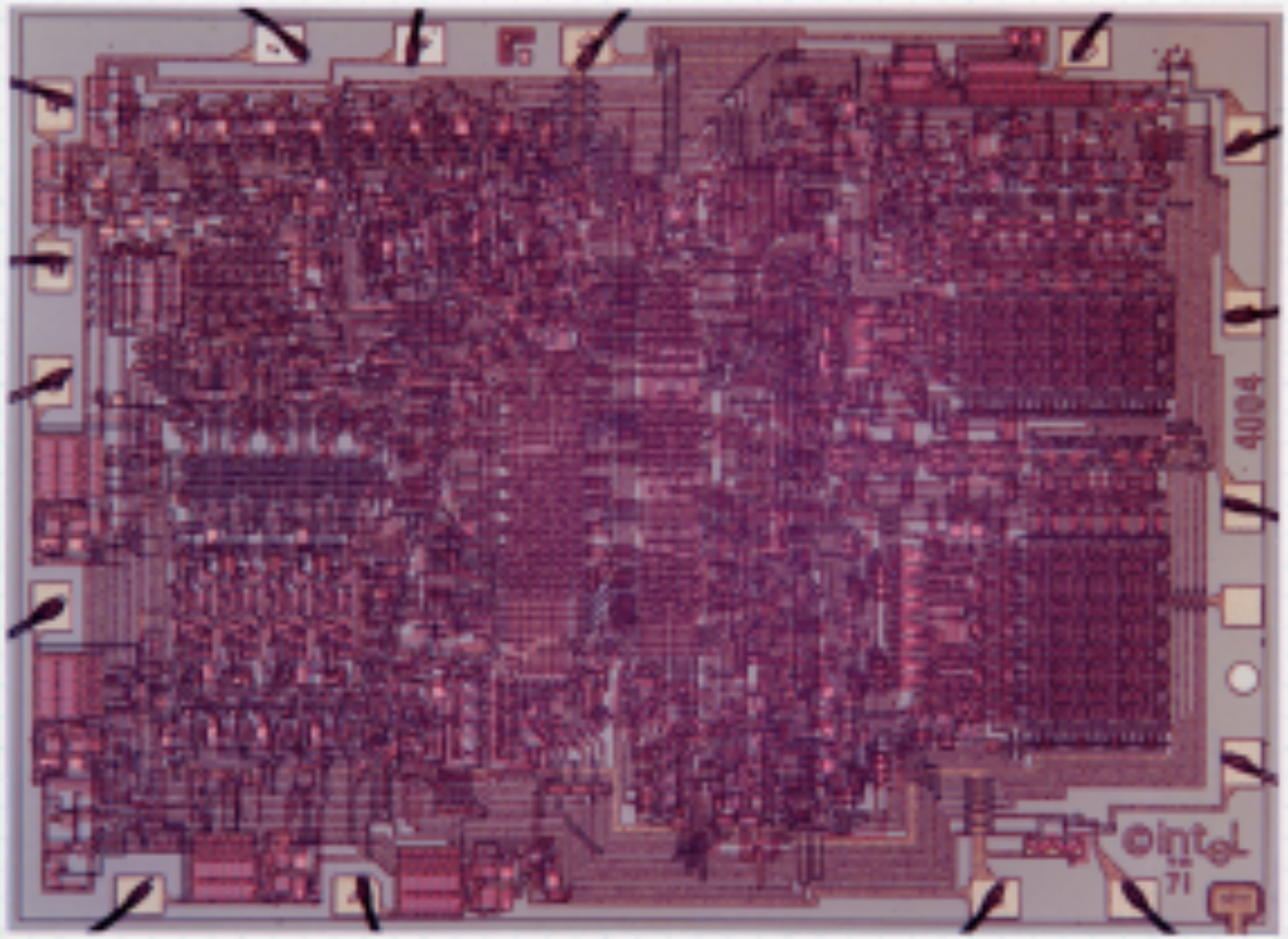
a. exploring parallelism and pipelining on the bit-level,
b. optimizing the encoding of data values (number representation), and/or
c. reducing the required memory bandwidth by implementing data-structures and algorithms directly on the FPGA.

In addition, the speedup could be translated into savings in power consumption.

Second, I suggest a programming methodology for FPGAs based on Domain Specific Compilers. Domain specific compilers implement a divide-and-conquer, bottom-up approach to programming FPGAs. The vast space of possible architectures fragments into architecture families, which indirectly defines application domains. A domain specific compiler targets one architecture family and thus focuses on a single application domain. The StReAm compiler, under development at Bell Labs and Imperial College targets pipelined data-flow graphs mapped directly from object-oriented C++ to hardware. The goal is to provide a simple abstraction for programming FPGAs analogous to the abstraction of a microprocessor provided by the C programming language.

Euclids Elements, Representing $a^2+b^2=c^2$

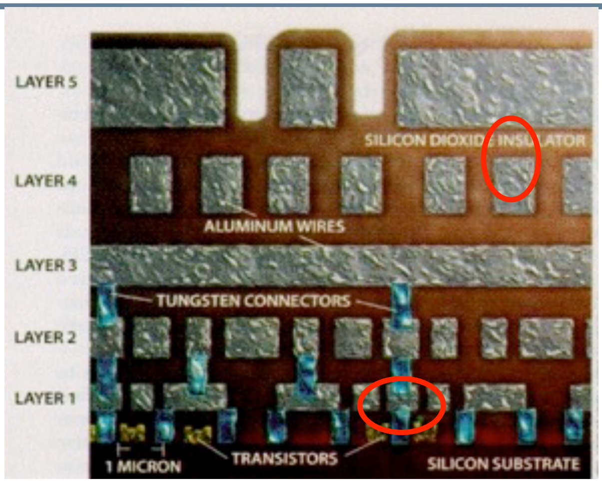
1971



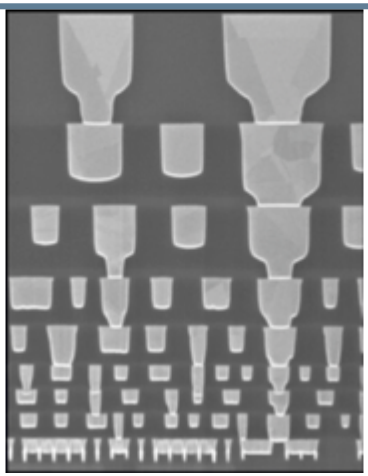
The Intel 4004 microprocessor, which was introduced in 1971. The 4004 contained 2300 transistors and performed 60,000 calculations per second. Courtesy: Intel.

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

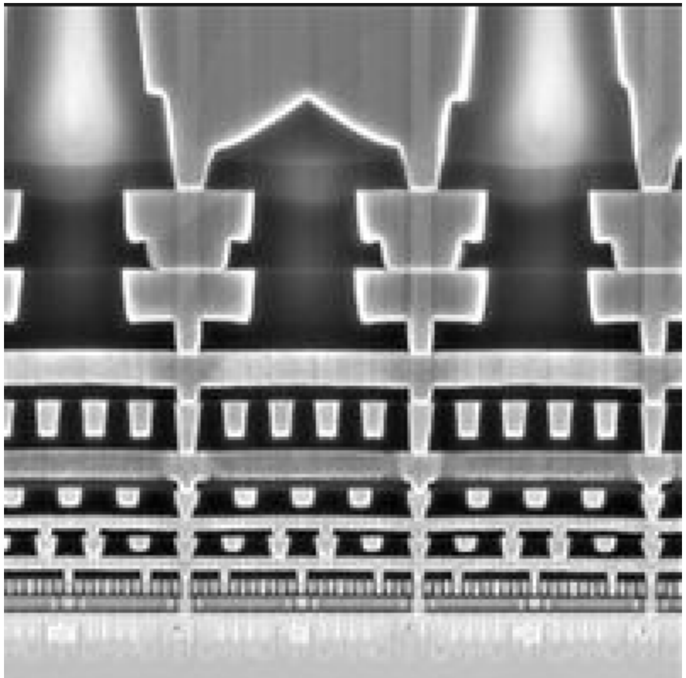
Wires getting much bigger in size than transistors



1 µm CMOS
10^{-6}



32nm CMOS
10^{-9}

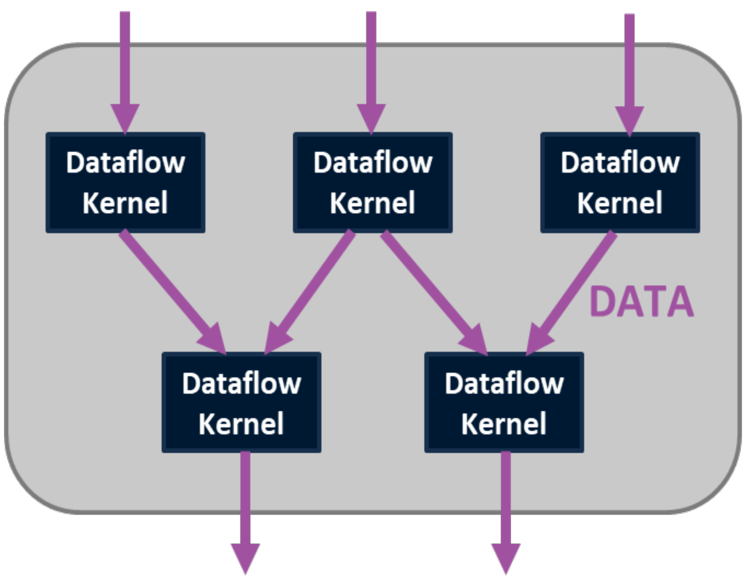


10nm CMOS
10^{-9}

2003: The Maxeler Static Dataflow Model

let's create a computing structure to fit the problem

- Data moves continuously (flow) and drive computation
- Compute in ***Space*** – arrange operations in 2D
- Find optimal solution for any *specific* flow problem
  - No wasted silicon – maximum performance density
  - No wasted clock cycles – data rate = clock rate
  - Predictable throughput & latency, MIN ENERGY for moving data



a different way to compute?

a warning from history...I did not listen...

LECTURE 45 26 AUGUST 1946

A PARALLEL CHANNEL COMPUTING MACHINE

Lecture by
J. P. Eckert, Jr.
Electronic Control Company

. . . Again I wish to reiterate the point that all the arguments for parallel operation are only valid provided one applies them to the steps which the built in or wired in programming of the machine operates. Any steps which are programmed by the operator, who sets up the machine, should be set up only in a serial fashion. It has been shown over and over again that any departure from this procedure results in a system which is much too complicated to use.

See also http://www.digital60.org/birth/themooreschool/lectures.html#45

Credit: Prof. Paul H.J. Kelly

- J. P, Eckert, Jr (Co-Inventor of ENIAC)

Fast and Slow

John von Neumann, 1946:

“We are forced to recognize the possibility of constructing a ***hierarchy of memories***, each of which has greater capacity than the preceding, but which is less quickly accessible.”

So, clearly what matters is the location of data!!

MAXELER Technologies

As a result of the von Neumann hierarchy:

latency.txt **Assembly Instruction: LD A, (B) at 2 GHz**

```
Latency Comparison Numbers
--------------------------
L1 cache reference                            0.5 ns
Branch mispredict                             5   ns
L2 cache reference                            7   ns
Mutex lock/unlock                            25   ns
Main memory reference                       100   ns
Compress 1K bytes with Zippy              3,000   ns        3 us
Send 1K bytes over 1 Gbps network        10,000   ns       10 us
Read 4K randomly from SSD*              150,000   ns      150 us
Read 1 MB sequentially from memory      250,000   ns      250 us
Round trip within same datacenter       500,000   ns      500 us
Read 1 MB sequentially from SSD*      1,000,000   ns    1,000 us    1 ms
Disk seek                            10,000,000   ns   10,000 us   10 ms
Read 1 MB sequentially from disk     20,000,000   ns   20,000 us   20 ms
Send packet CA->Netherlands->CA     150,000,000   ns  150,000 us  150 ms
```

SQL Database Transactions > 150,000 us > 150ms

MAXELER Technologies

PYTHON vs SQL: Real Customer Project

SQL

```
s = run_sql_procedure('[redacted]ys
```

Running SQL:

```
Duration: 114.239367 seconds
```

Python

```
p = [redacted]qu ... ])
```

Running Python:

```
Duration: 0.952739 seconds
```

```
speedup(p, s)
```

```
Speedup: 119.9x
```

MAXELER Technologies

Kolmogorov Complexity, 1965

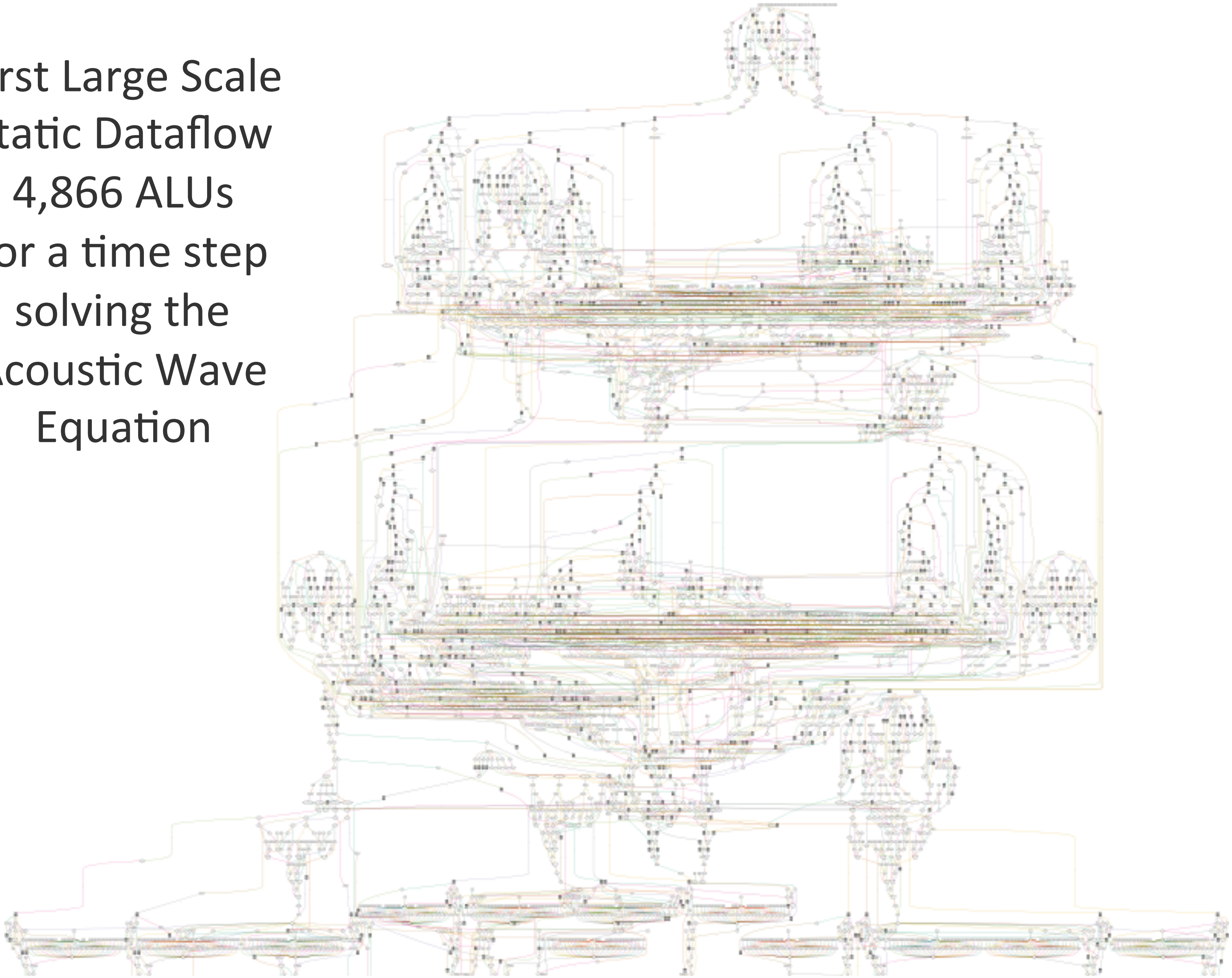
Definition (Kolmogorov):
“If a description of *string s*, *d*(*s*),
is of minimal length, […]
it is called a **minimal description** of *s*.

the length of *d*(*s*), […] is the **Kolmogorov complexity** of *s*,
written *K*(*s*), where $K(s) = |d(s)|$”

Of course K(s) depends heavily on the ***Language L***
used to describe actions in K
(e.g., Java, Esperanto, an Executable file, etc).

Kolmogorov, A.N. (1965). "Three Approaches to the Quantitative Definition of Information". *Problems Inform. Transmission* **1** (1): 1–7.

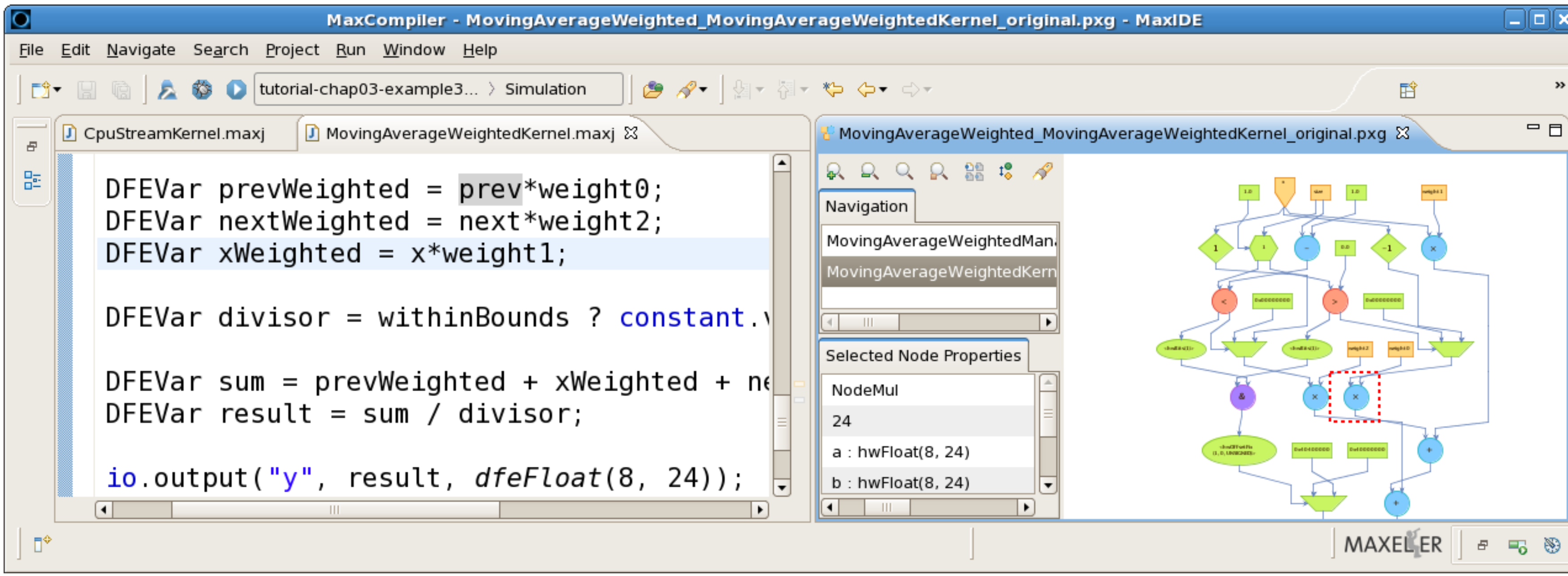
First Large Scale Static Dataflow 4,866 ALUs for a time step solving the Acoustic Wave Equation



Impossible? or merely hard?

MaxJ: A Dataflow Programming Model

Syntax based on Java, and **Semantics for** static dataflow



Making DATAFLOW programming fun,

“Easy, Desirable, and Affordable” [Terry Leahy, former CEO of TESCO]

Ludwig Wittgenstein

Born: Vienna, Austria 1889
Died: Cambridge, England, 1951

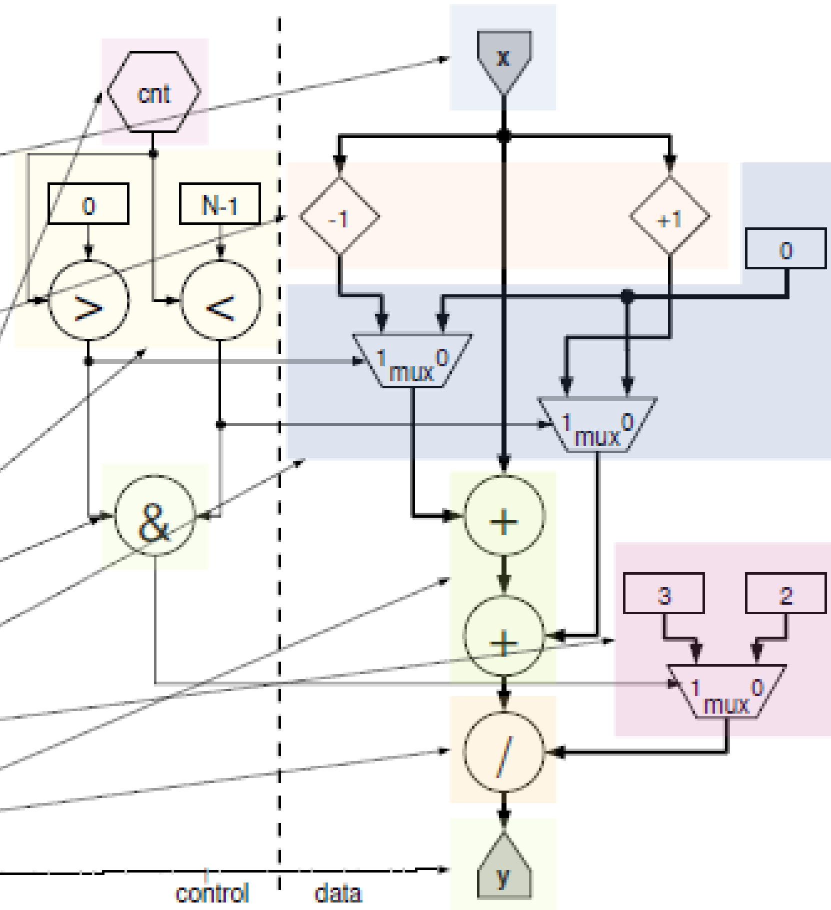
> The limits of my language mean
> the limits of my world

Dataflow Corollary:
The limits of my programming language mean the limits of what I can optimize...

A Dataflow Kernel

Every line of code corresponds to a resource

```
class MovingAverageKernel extends Kernel {

    MovingAverageKernel(KernelParameters parameters) {
        super(parameters);

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

        DFEVar size = io.scalarInput("size", dfeUInt(32));

        // Data
        DFEVar prevOriginal = stream.offset(x, -1);
        DFEVar nextOriginal = stream.offset(x, 1);

        // Control
        DFEVar count = control.count.simpleCounter(32, size);

        DFEVar aboveLowerBound = count > 0;
        DFEVar belowUpperBound = count < size - 1;

        DFEVar withinBounds = aboveLowerBound & belowUpperBound;

        DFEVar prev = aboveLowerBound ? prevOriginal : 0;
        DFEVar next = belowUpperBound ? nextOriginal : 0;

        DFEVar divisor = withinBounds ? constant.var(dfeFloat(8, 24), 3) : 2;

        DFEVar sum = prev + x + next;
        DFEVar result = sum / divisor;

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



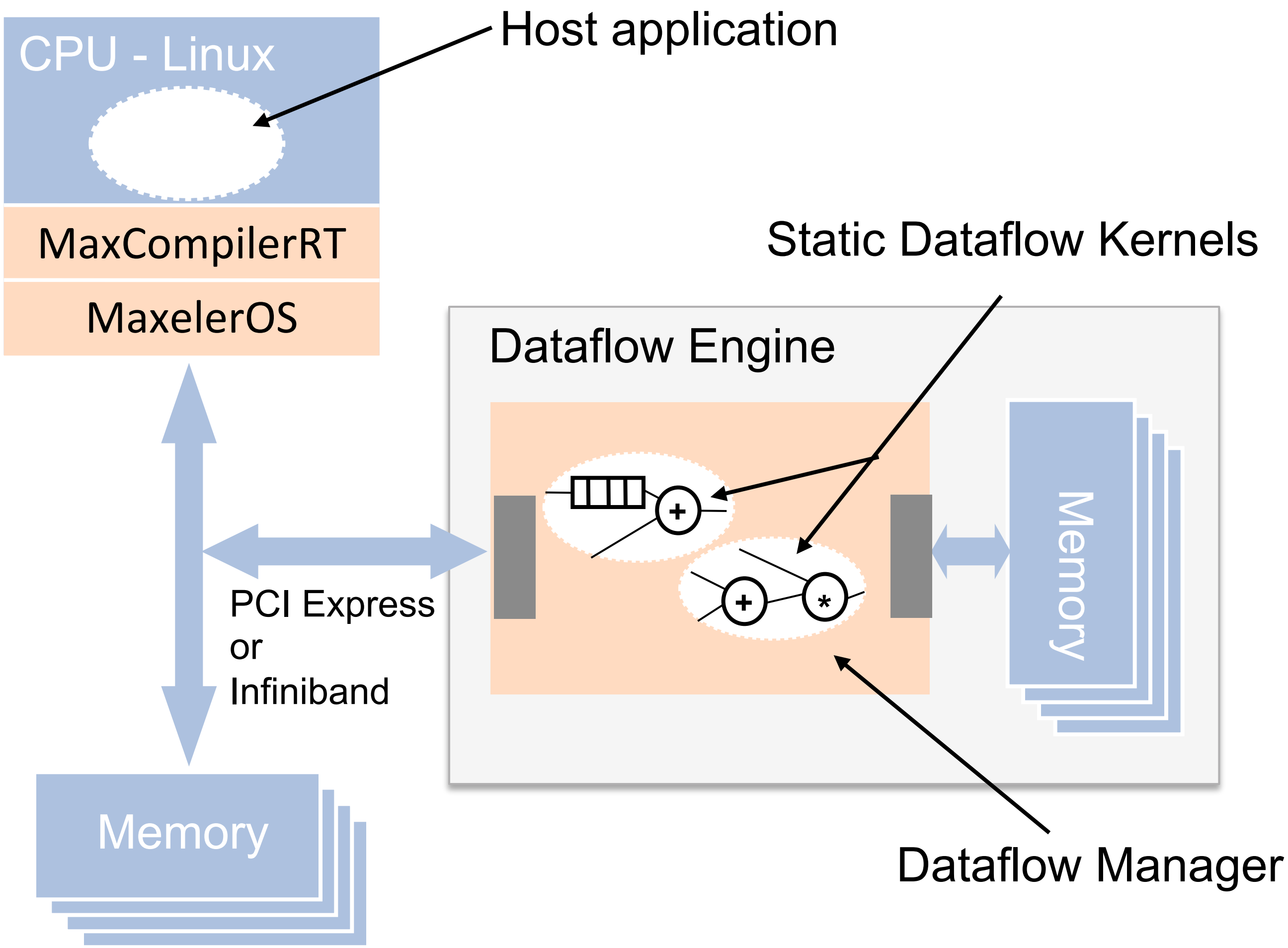
Dataflow can be annotated back into code

every line of dataflow code takes a certain space

```
  LUTs      FFs    BRAMs     DSPs : MyKernel.java
   727      871      1.0        2 : resources used by this file
 0.24%    0.15%    0.09%    0.10% : % of available
71.41%   61.82%  100.00%  100.00% : % of total used
94.29%   97.21%  100.00%  100.00% : % of user resources
                                  :
                                  : public class MyKernel extends Kernel {
                                  :   public MyKernel (KernelParameters parameters) {
                                  :      super(parameters);
     1       31      0.0        0 :      DFEVar p = io.input("p", dfeFloat(8,24));
     2        9      0.0        0 :      DFEVar q = io.input("q", dfeUInt(8));
                                  :      DFEVar offset = io.scalarInput("offset", dfeUInt(8));
     8        8      0.0        0 :      DFEVar addr = offset + q;
    18       40      1.0        0 :      DFEVar v = mem.romMapped("table", addr,
                                  :                                  dfeFloat(8,24), 256);
   139      145      0.0        2 :      p = p * p;
   401      541      0.0        0 :      p = p + v;
                                  :      io.output("r", p, dfeFloat(8,24));
                                  :   }
                                  : }
```


MAXELER Technologies

Putting it all together: A Dataflow System Architecture

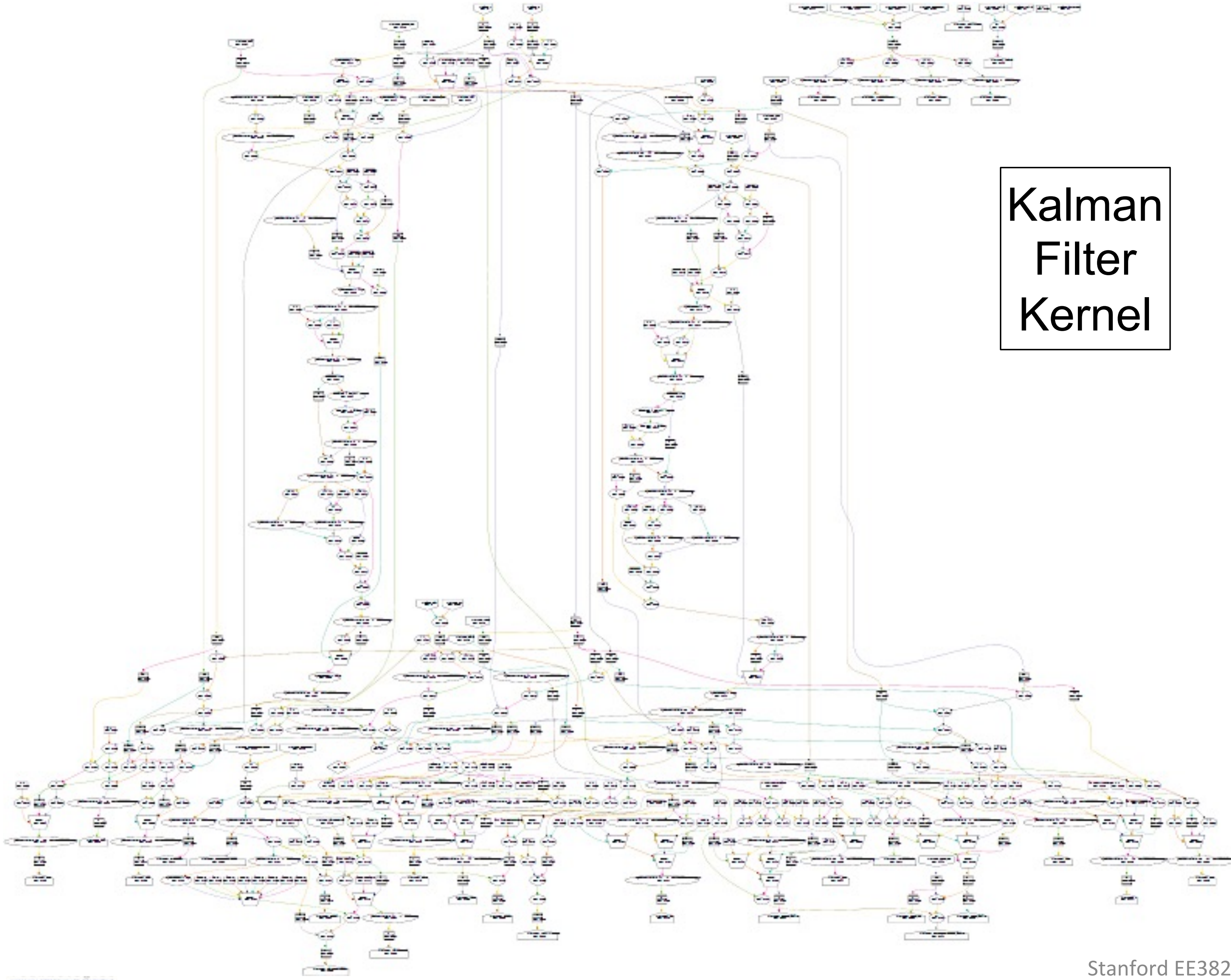


Kalman Filter as a State Machine in MaxJ

$$x_k^{k-1} = \mathbf{F}_{k-1} x_{k-1}$$
$$\mathbf{C}_k^{k-1} = \mathbf{F}_{k-1} \mathbf{C}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$$
$$r_k^{k-1} = m_k - \mathbf{H}_k x_k^{k-1}$$
$$\mathbf{R}_k^{k-1} = \mathbf{V}_k + \mathbf{H}_k \mathbf{C}_k^{k-1} \mathbf{H}_k^T$$
$$\mathbf{K}_k = \mathbf{C}_k^{k-1} \mathbf{H}_k^T \left(\mathbf{R}_k^{k-1}\right)^{-1}$$
$$x_k = x_k^{k-1} + \mathbf{K}_k r_k^{k-1}$$
$$\mathbf{C}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{C}_k^{k-1}$$
$$\chi^2_+ = r_k^{k-1T} \left(\mathbf{R}_k^{k-1}\right)^{-1} r_k^{k-1}$$
$$\chi^2_k = \chi^2_{k-1} + \chi^2_+$$

```
class TrackFitKernel extends Kernel{
 protected TrackFitKernel(KernelParameters p){
  super(p);
  Stub stubIn = Stub.input(this, “stubIn”);
  State kStateIn = State.input(this,“stateIn”);
  KF kfWorker = new KF(this,kStateIn,stubIn);
  State kStateUp = KF.update();
  kStateUp.output(“stateOut”);
 }
}

class KF extends KernelLib{
 public KF(Kernel owner,State state,Stub stub){
  Vector x = state.x();
  Matrix pxx = state.pxx();
  Matrix h = H(stub);
  Matrix pxd = pxx * H.transpose();
  …
  Vector residual = d – hx;
  Vector xUp = x + K * residual;
  return new State(xUp, pxxUp);
 }
}
```

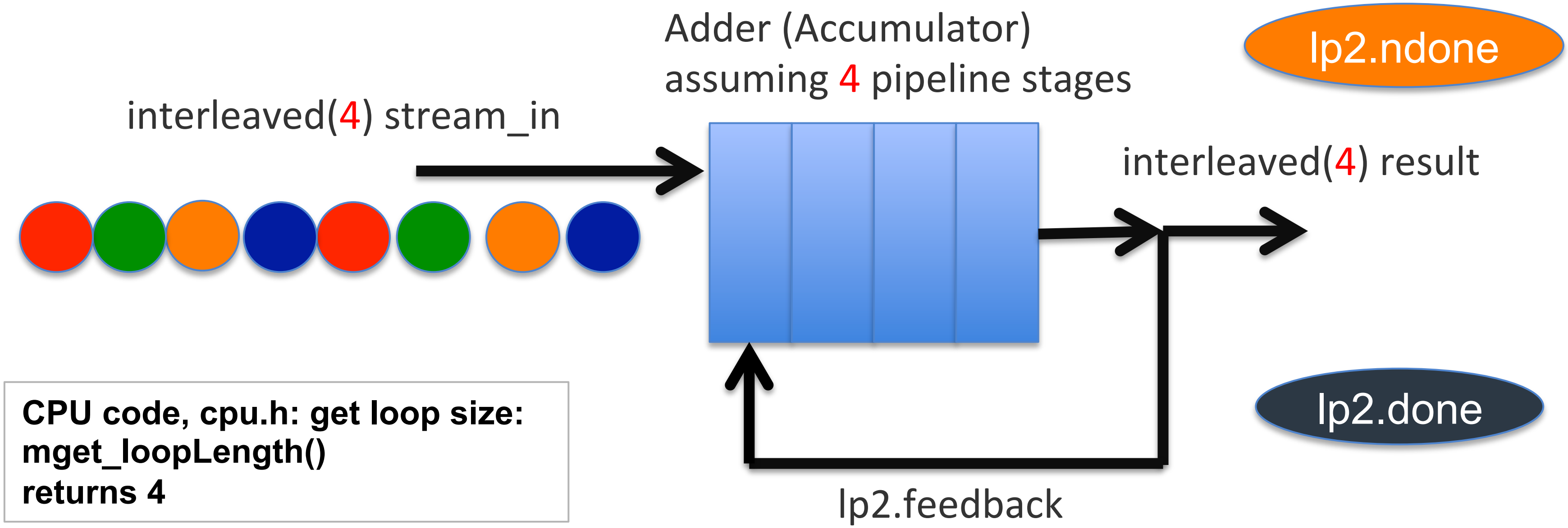
MAXELER Technologies



Kalman Filter Kernel

Parallel Loop with Dependency

4 Accumulators in 1 pipelined unit

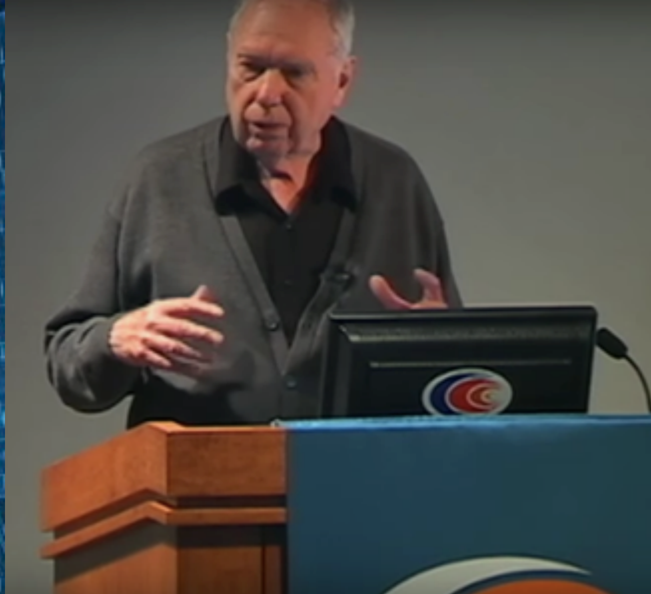


CPU code, cpu.h: get loop size:
mget_loopLength()
returns 4

```
for j=1 to 4: out[j]=0.0;
for i=1 to N:    // sequential loop
  for j=1 to 4: // data parallel loop
      out[j] = out[j]+stream_in[i];
```

Of course j could be a lot larger, but we do **4 at a time** here since we assume 4 stages in a

```
DFEParLoop lp2 = new DFEParLoop(this, "lp2");
   DFEVar in = io.input("in", dfeFloat(8, 24), lp2.ndone);
   lp2.set_input(dfeFloat(8,24), 0.0);
     DFEVar result = in + lp2.feedback;
   lp2.set_output(result);
   io.output("result", result, dfeFloat(8, 24), lp2.done);
```



Prof Mike Flynn, Stanford

https://www.youtube.com/watch?v=ybnOuI9jNgE
from 4’50’’

Slotnick’s law (of effort)

from the great debate with Gene Amdahl, 1967

“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

Maxeler Dataflow Engines (DFEs)

MPC C Series

High Density DFEs

Intel Xeon CPU cores and up to 6 DFEs with 576GB of RAM

MPC X Series

The Dataflow Appliance

Dense compute with 8 DFEs, 768GB of RAM and dynamic allocation of DFEs to CPU servers with zero-copy RDMA access

MPC N Series

The Low Latency Appliance

Intel Xeon CPUs and 1-2 DFEs with direct links to up to six 10Gbit Ethernet connections

MaxWorkstation

Desktop dataflow development system

MaxRack

10, 20 or 40 node rack systems integrating compute, networking & storage

MaxCloud

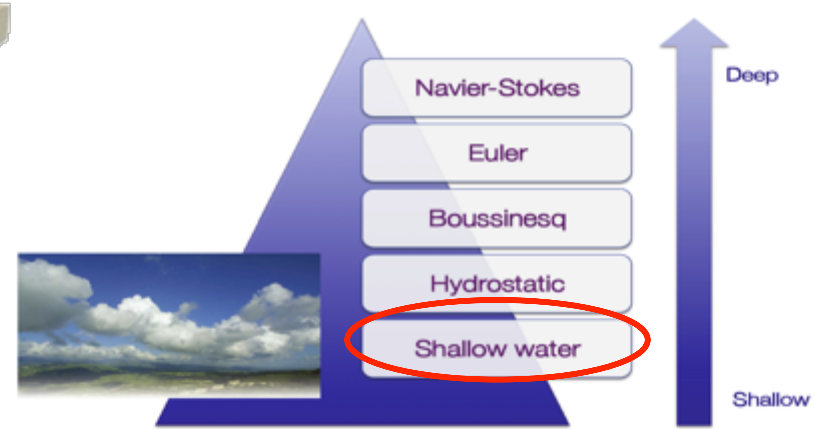
Hosted, on-demand, scalable accelerated compute

Dataflow Engines

48GB DDR3, high-speed connectivity and dense configurable logic

2013 ... getting another 10-20x of speedup if you optimize Flow of data with Maxeler hardware

Simulating the Atmosphere via the Shallow Water Equation



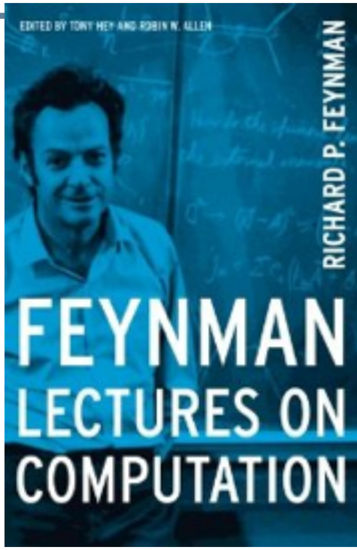
[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL Conference 2013]

Platform	Performance	Speedup	Efficiency	Energy Improvement
6-core CPU	4.66K	1	20.71	1
Tianhe-1A node	110.38K	23x	306.6	14.8x
MaxWorkstation	468.1K	**100x**	2.52K	**121.6x**
Maxeler MPC-X	1.54M	**330x**	3K	**144.9x**

9x

14x

How is it possible to beat the worlds fastest computer by 14x on speed and 9x on energy



Paraphrased: In theory, computing units can be constructed which use no energy.

Energy is only needed when **information** is lost.

Reordering of **information** does not require energy from a pure physics perspective.

Of course, moving **information** takes Energy…

Dataflow minimizes Data Movement!

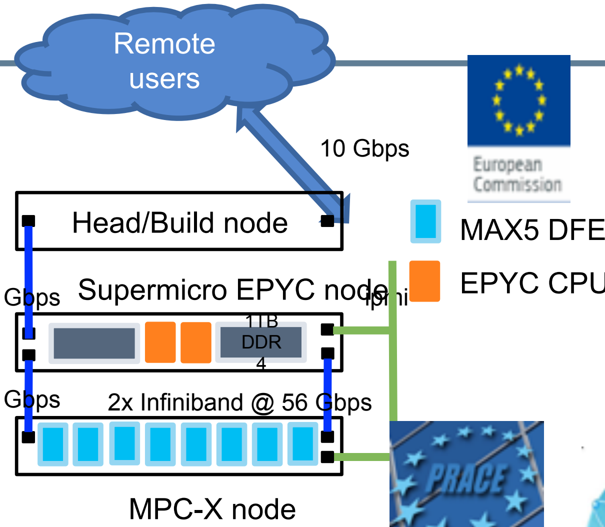
2015: First Maxeler Dataflow Supercomputer installed at UK Government Laboratory

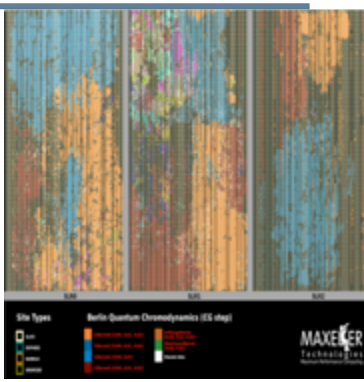
20-50x increased compute capability
per cubic-foot of data center space

2017: Maxeler Dataflow Machine at Jülich in Germany

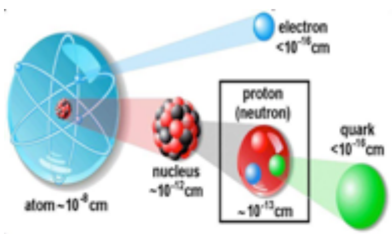
Pilot system deployed in Oct '17

- one 1U MPC-X with 8 MAX5 DFEs
- one 1U AMD EPYC based server
- one 1U login head node



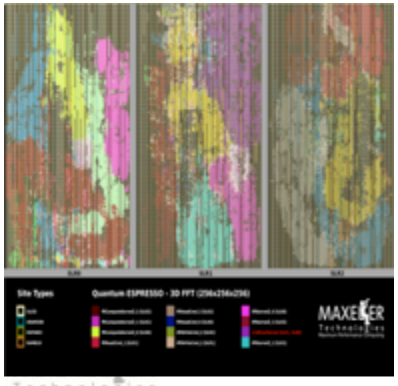


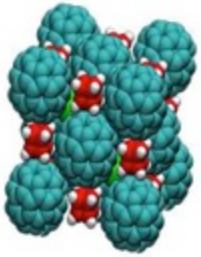
Applications	small case		big case		
	TTS [sec]	ETS [kWh]	TTS [sec]	ETS [kWh]	
BQCD	105	0.44	1,704.00	4.3	9x
BQCD (Ref: 1 rack Blue Gene/Q)			2,311.00	37.5	
NEMO (DFE)	388	0.164	1,945.00	3.8	7x
NEMO (Ref: 8,192 cores Cray XC30)			1,942.00	28.0	
QUANTUM ESPRESSO	32	0.013	3,210.00	7.6	5x
QUANTUM ESPRESSO (Ref: 2 racks BlueGene/Q)			3,200.00	36.4	
SPECFEM3D	232	0.096	5,150.00	77.2	4x
SPECFEM3D (Ref: 468 nodes Curie cluster)			15,500.00	297.5	



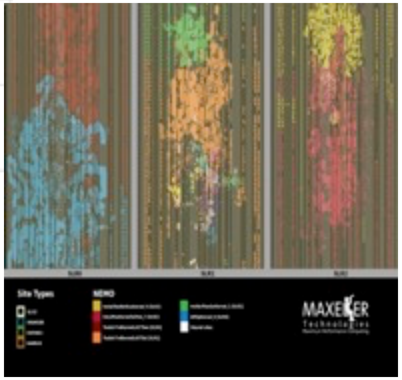
Berlin Quantum Chromo Dynamics

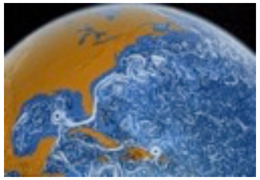
Energy Efficiency Improvements



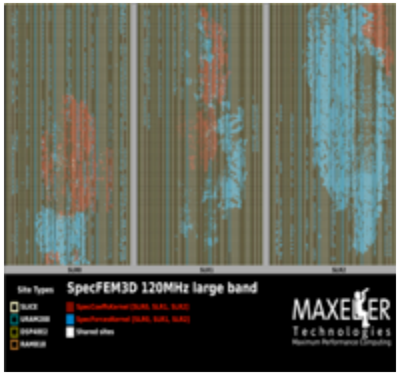


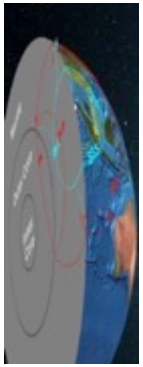
Quantum Espresso





NEMO

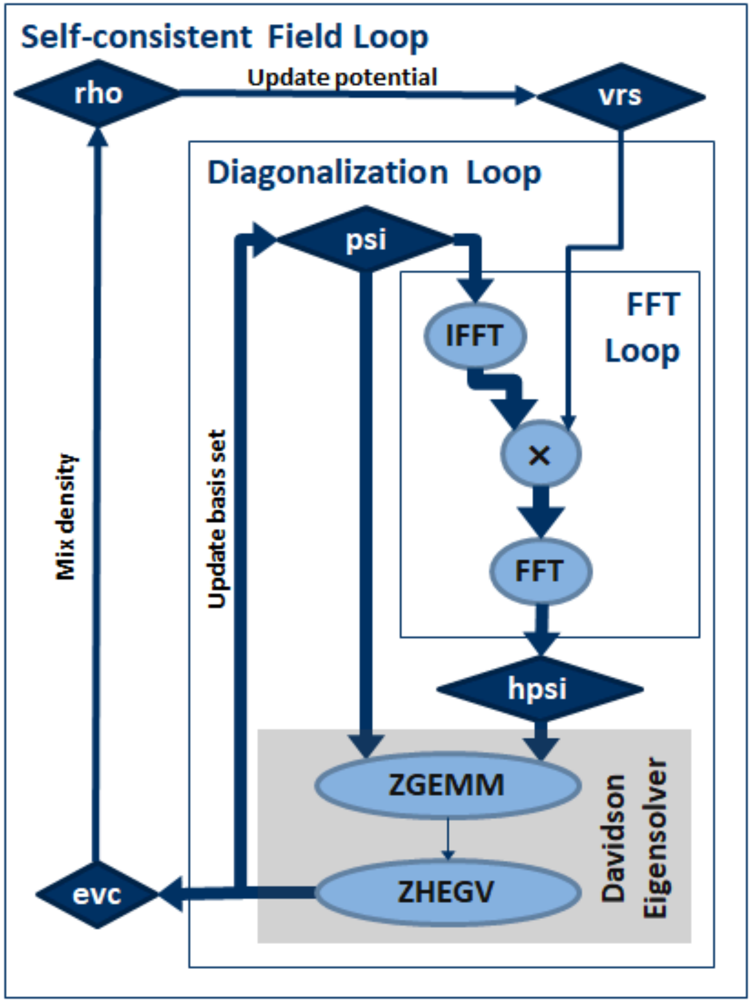




SPECFEM3D

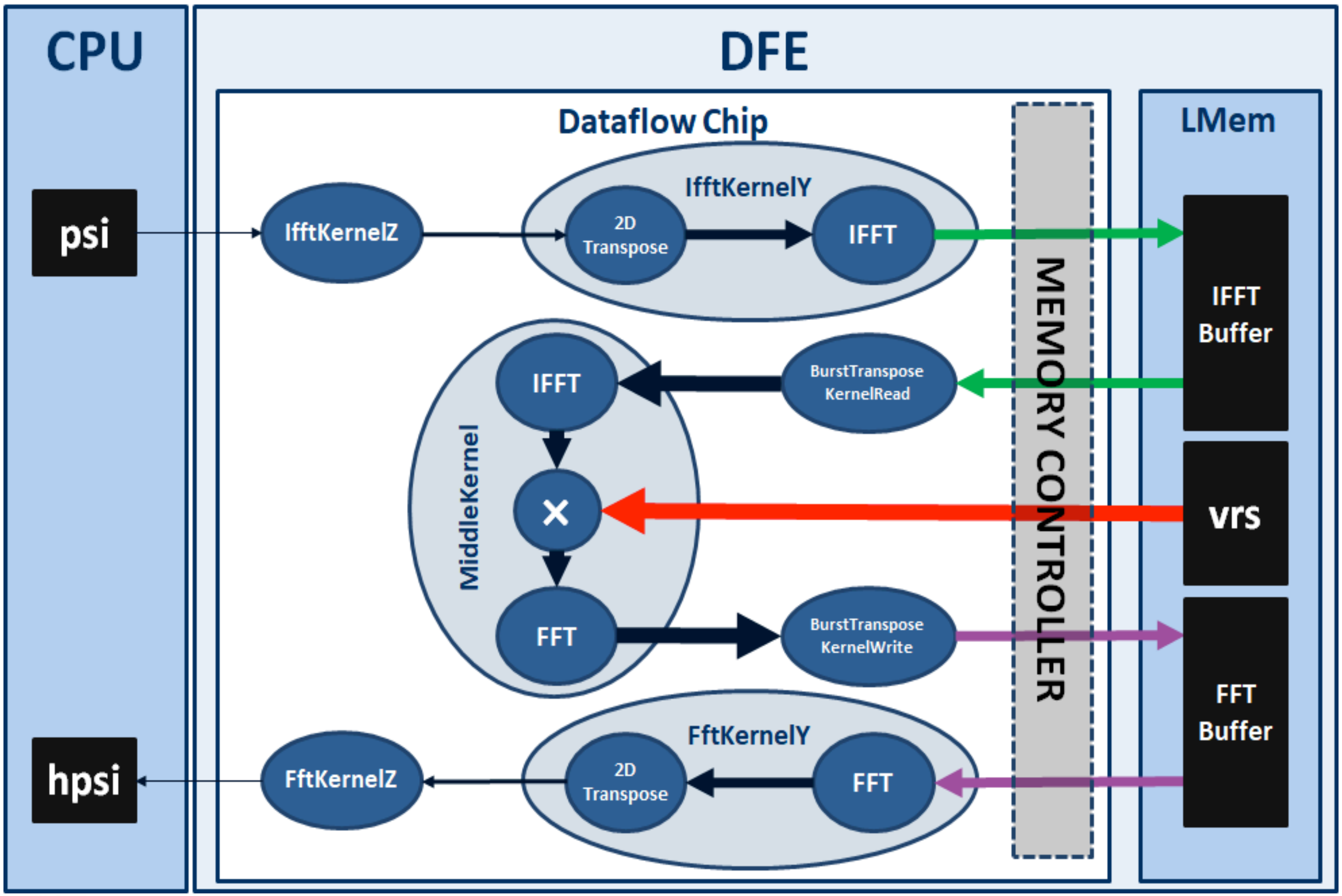
How do we migrate applications to Static Dataflow

Start with a Maxeler Loop Flow Graph

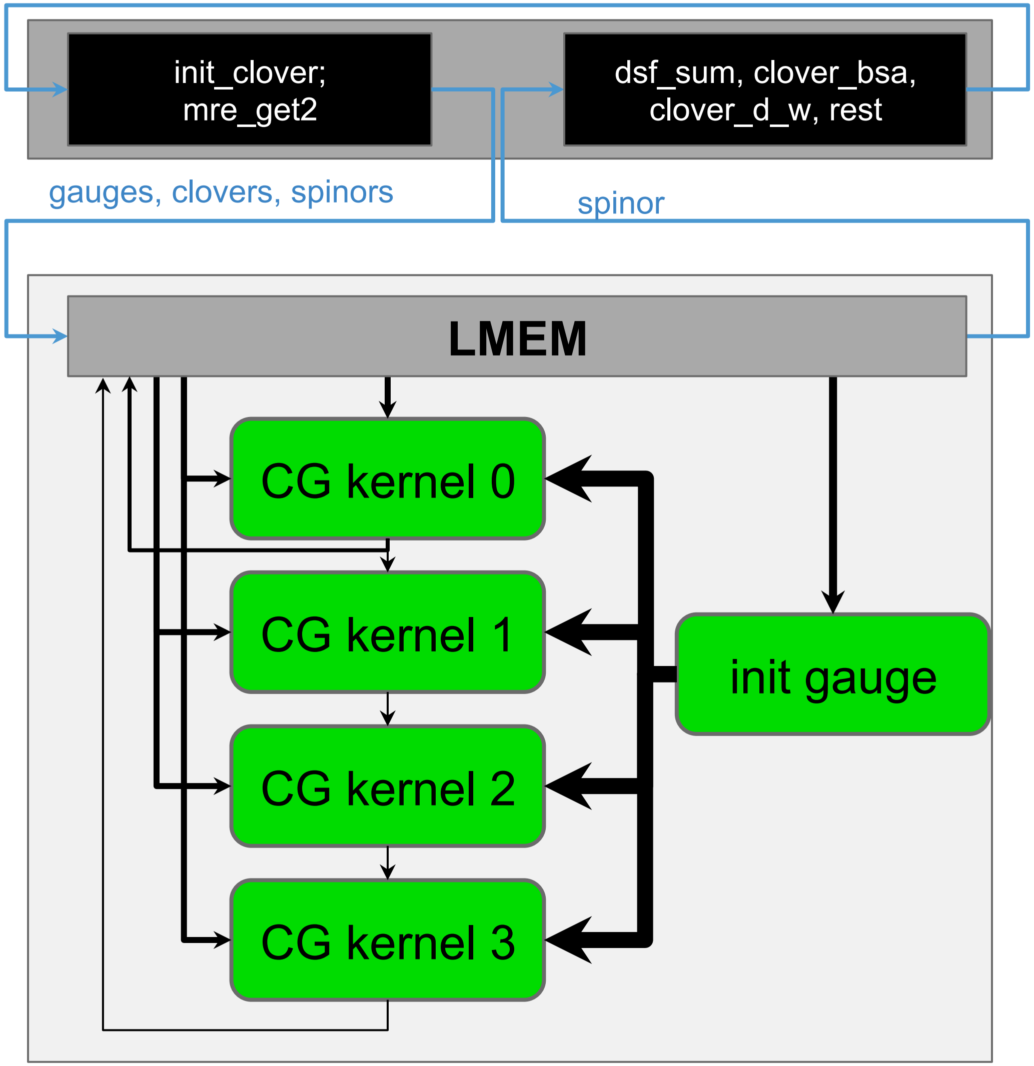


MAXELER Technologies

Convert the Loop Flow Graph into an Architecture

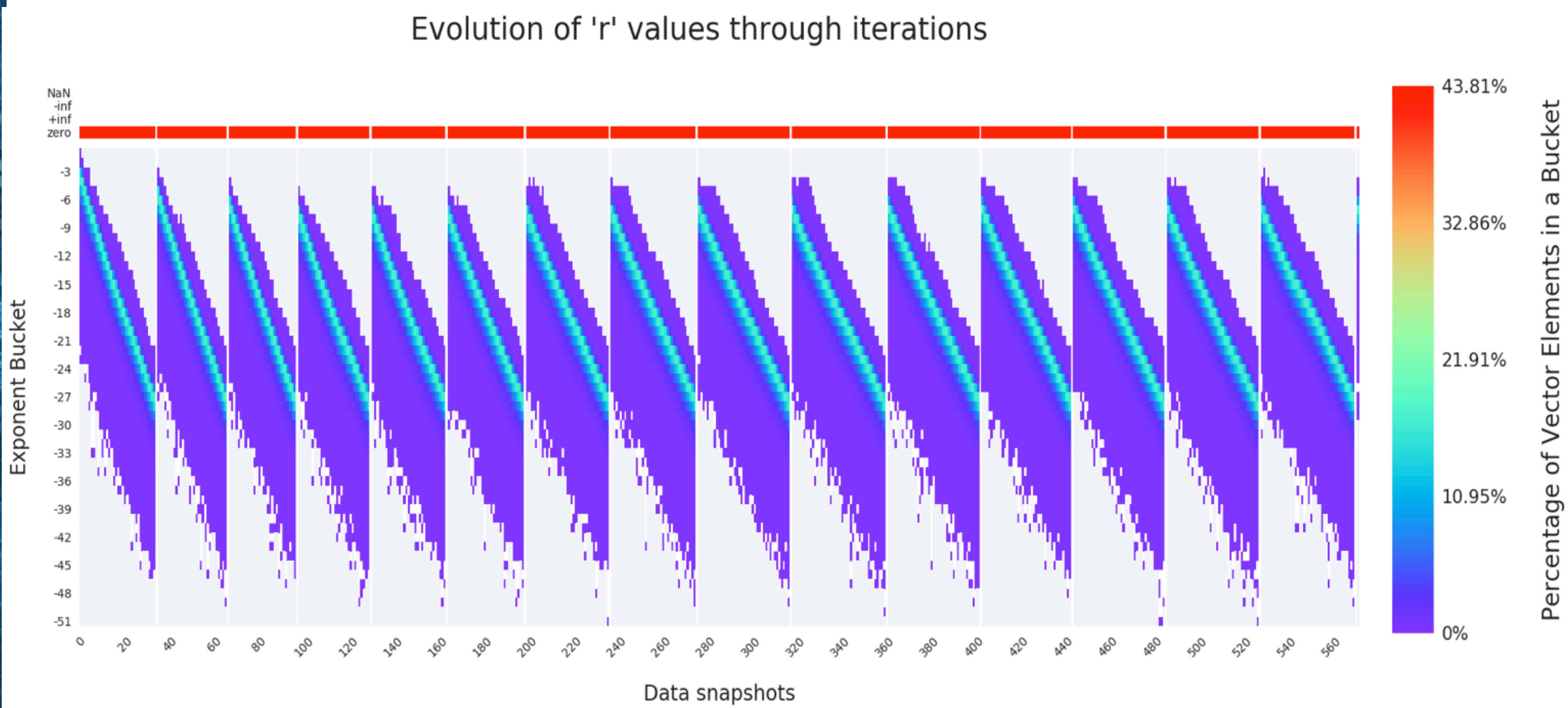


Example: Quantum Chromodynamics



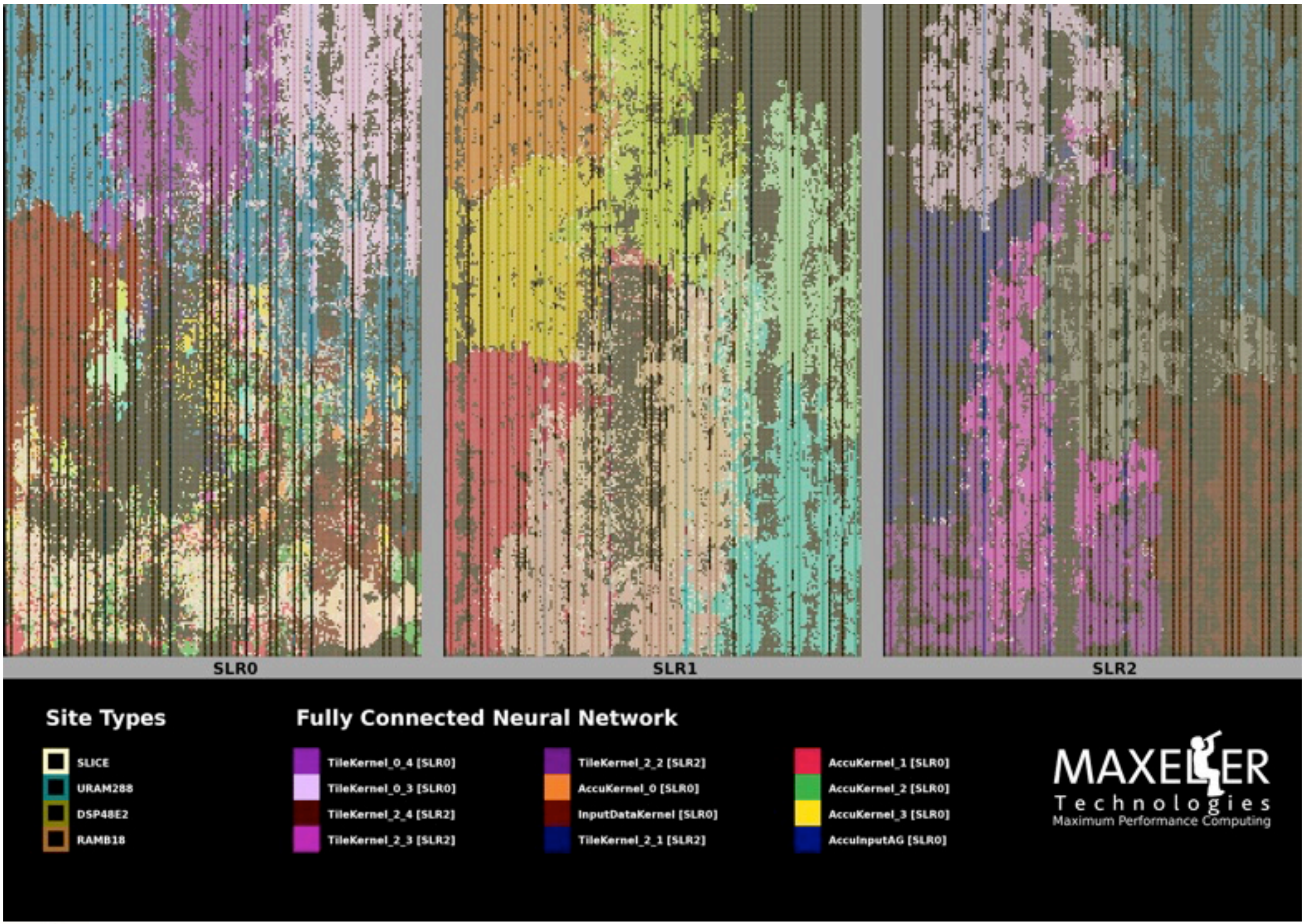
Numerical Analysis: Maxeler Value Profiling

We had to change how we implement (represent) numbers in computers, and manage numerics



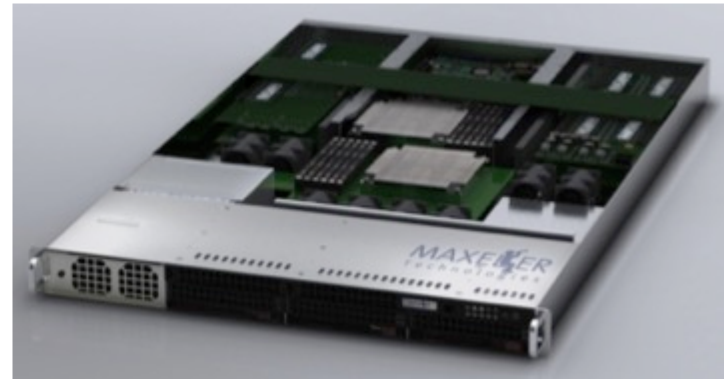
Changing the way we represent numbers in computers

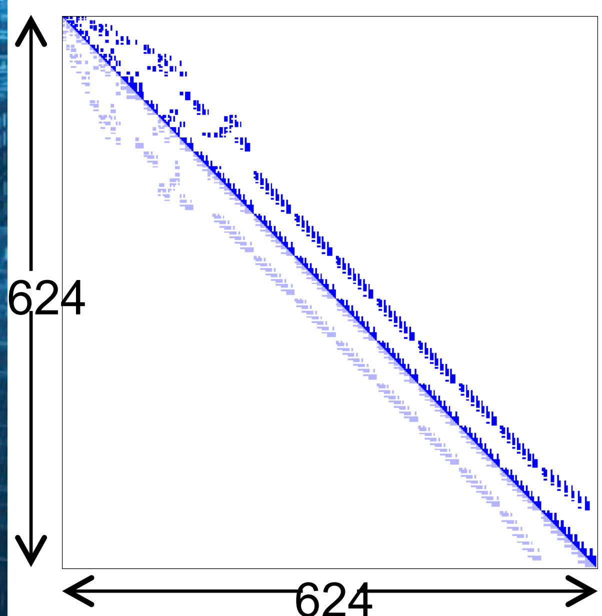
Aerial View of a Neural Network on FPGA

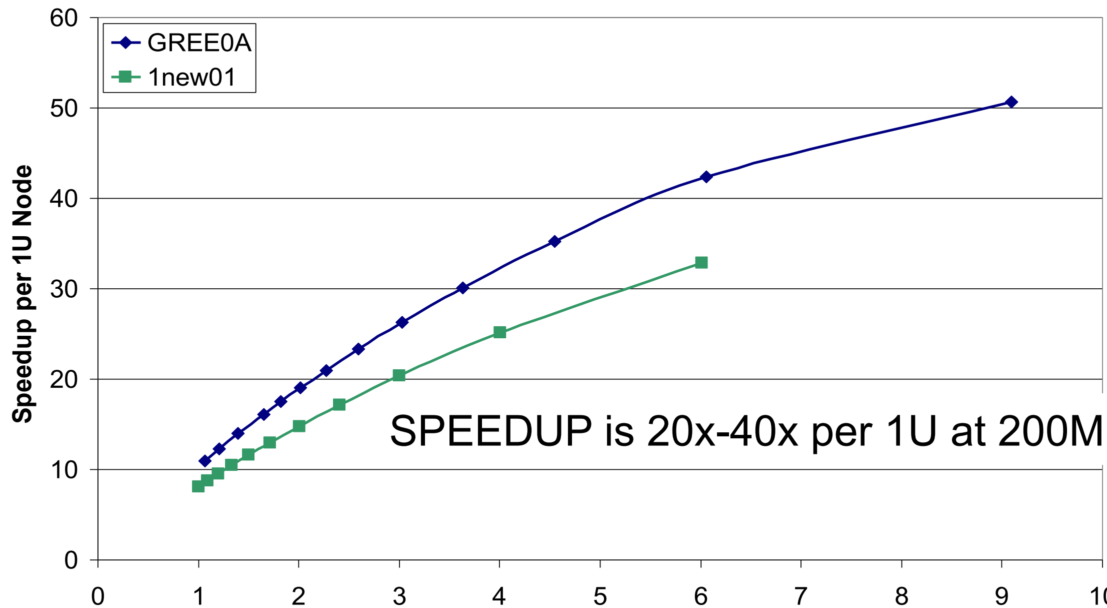


MAXELER Technologies

Sparse Matrix Dataflow – can't be done?





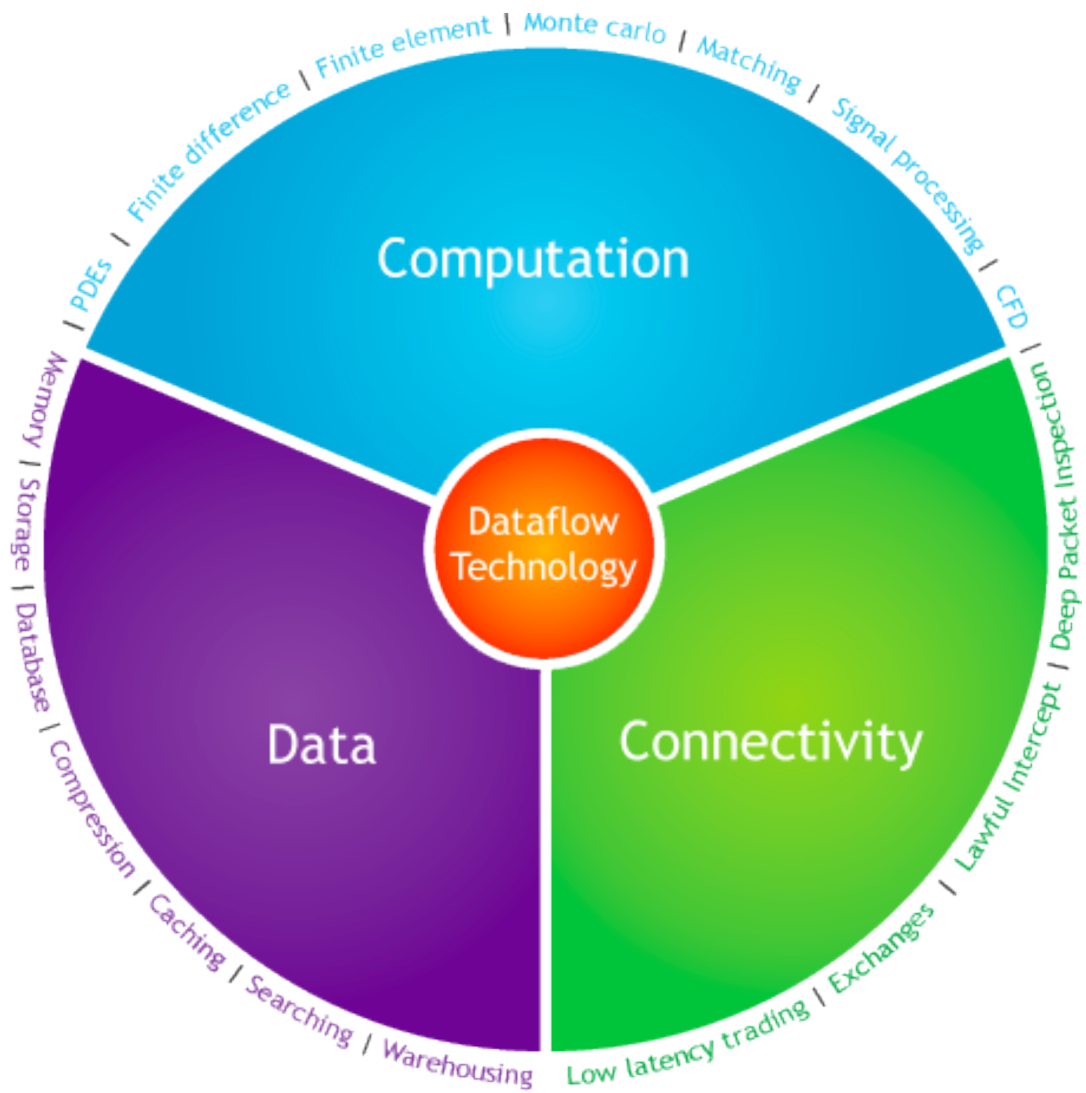


SPEEDUP is 20x-40x per 1U at 200MHz

Maxeler Domain Specific Address and Data Encoding published in IEEE Micro in 2011.

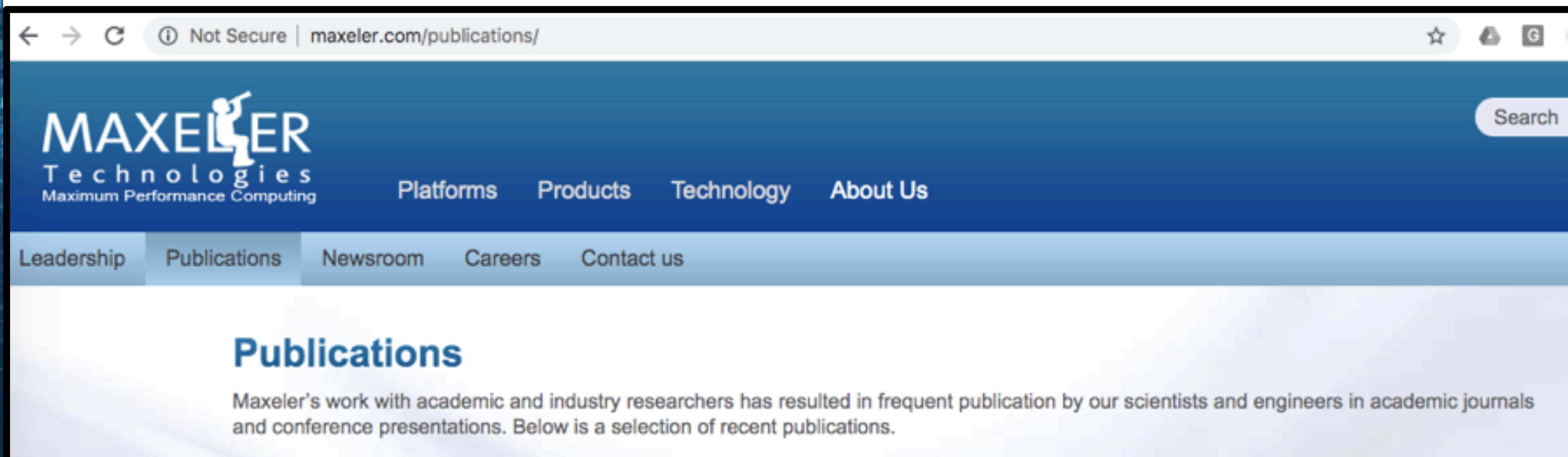
Schlumberger

Think Big



MAXELER Technologies

2020: Inspired over 1,000 publication



2020: Static Dataflow inspiring modern Chips

The Groq Architecture, see: Think Fast, A Tensor Streaming Processor for accelerating Deep Learning Networks, ISCA 2020.

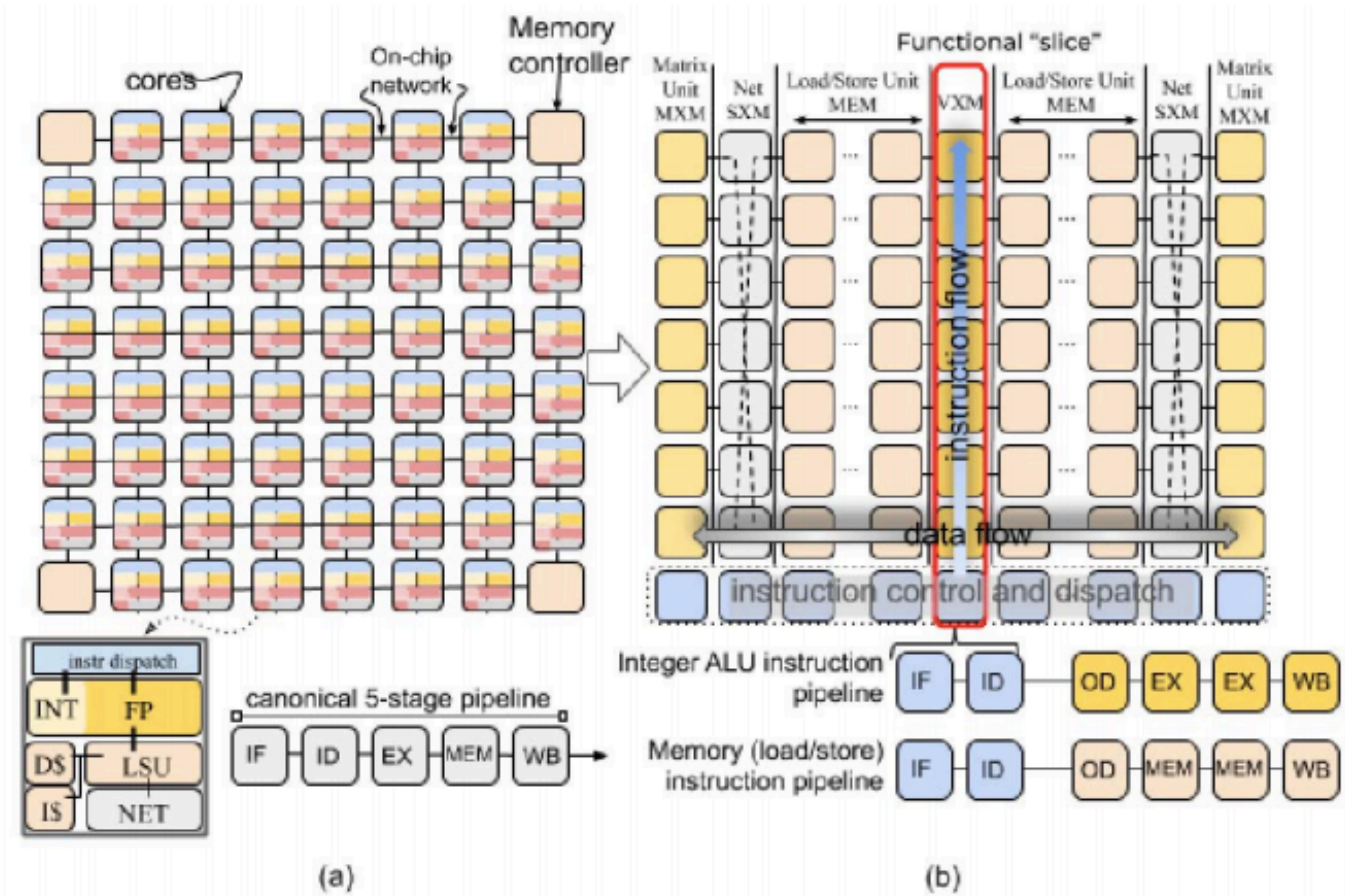


Fig. 1. Conventional 2D mesh of cores (a) reorganized into a functionally sliced arrangement of tiles (b).

“The hallmark of [Data]flow is a feeling of spontaneous joy”