Introduction to Dataflow Computing

Code Carpentry Workshop

Peter Sanders, July 2015

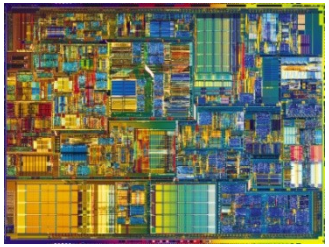
Computing on FPGAs

Programmable Spectrum

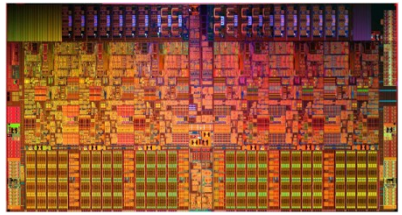
Control-flow processors

Dataflow processor
e.g. FPGA

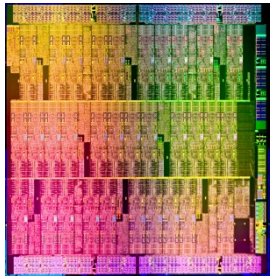
GK110



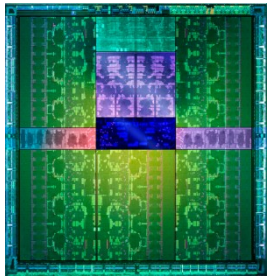
Single-Core CPU



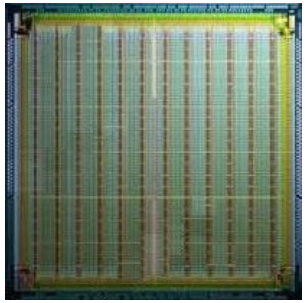
Multi-Core



Several-Cores



Many-Cores



Dataflow

Increasing Parallelism (#cores)

Increasing Core Complexity

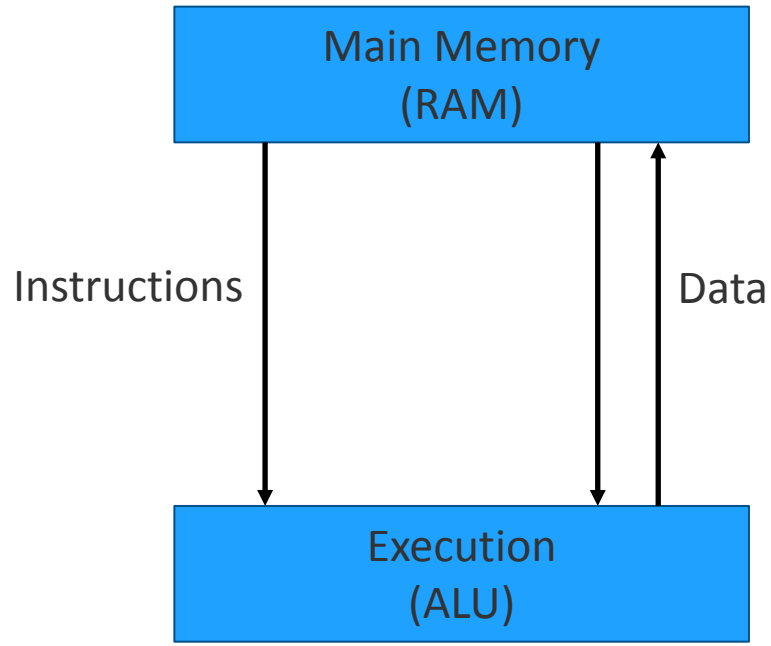
Intel, AMD

GPU (NVIDIA, AMD)
Tilera, XMOS etc...

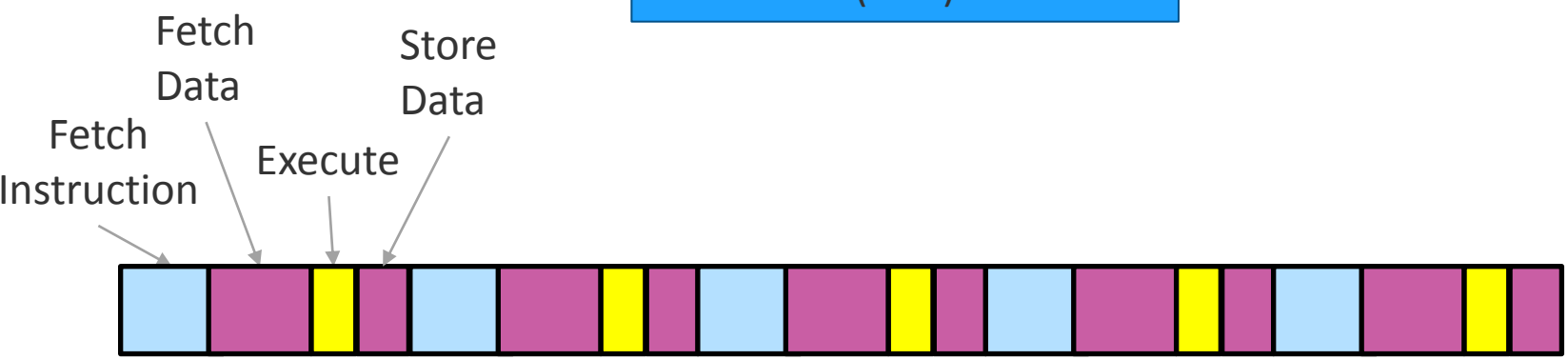
Maxeler

Hybrid e.g. AMD Fusion, IBM Cell

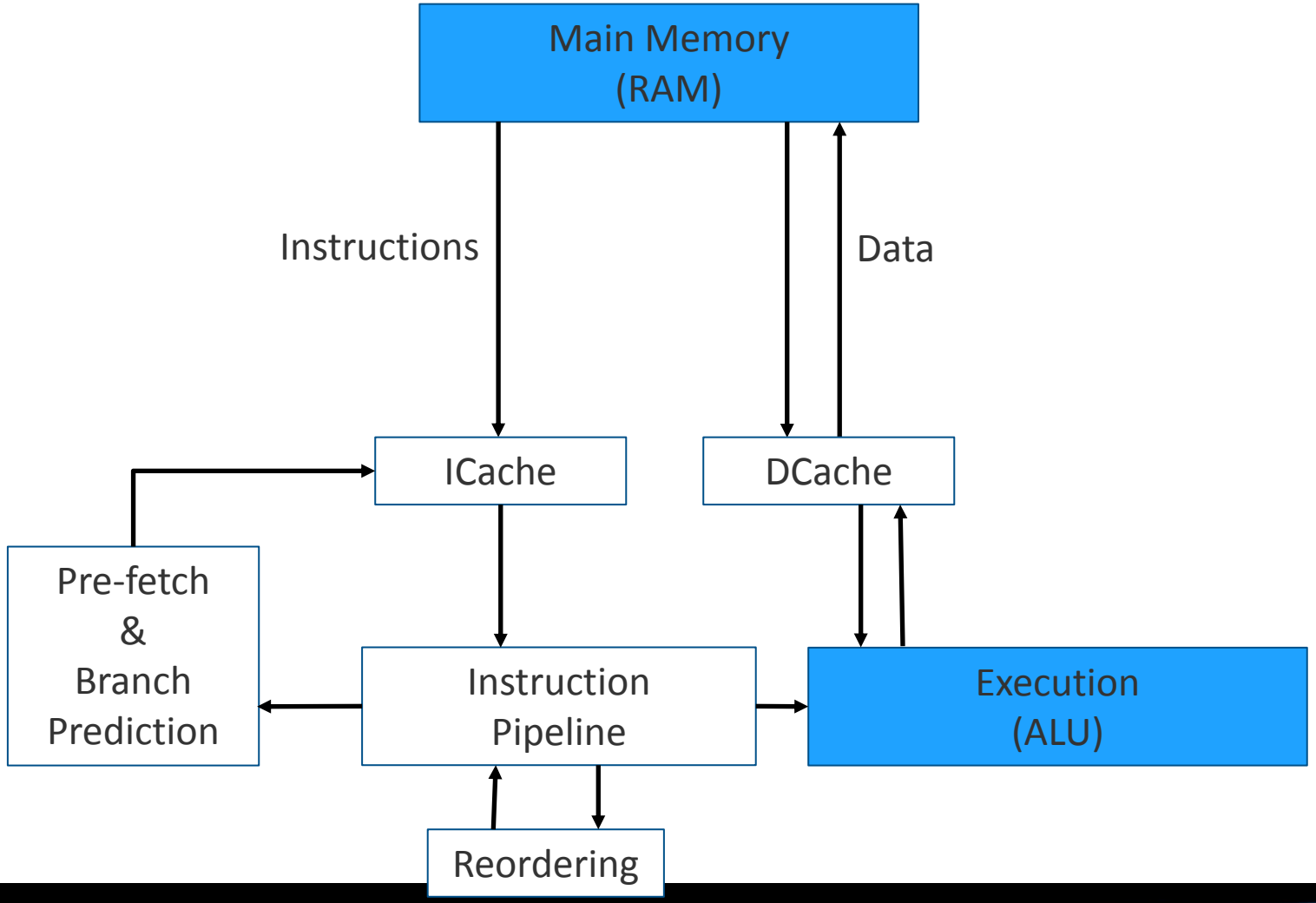
Control-flow processor (CPU)



Von Neumann
Architecture

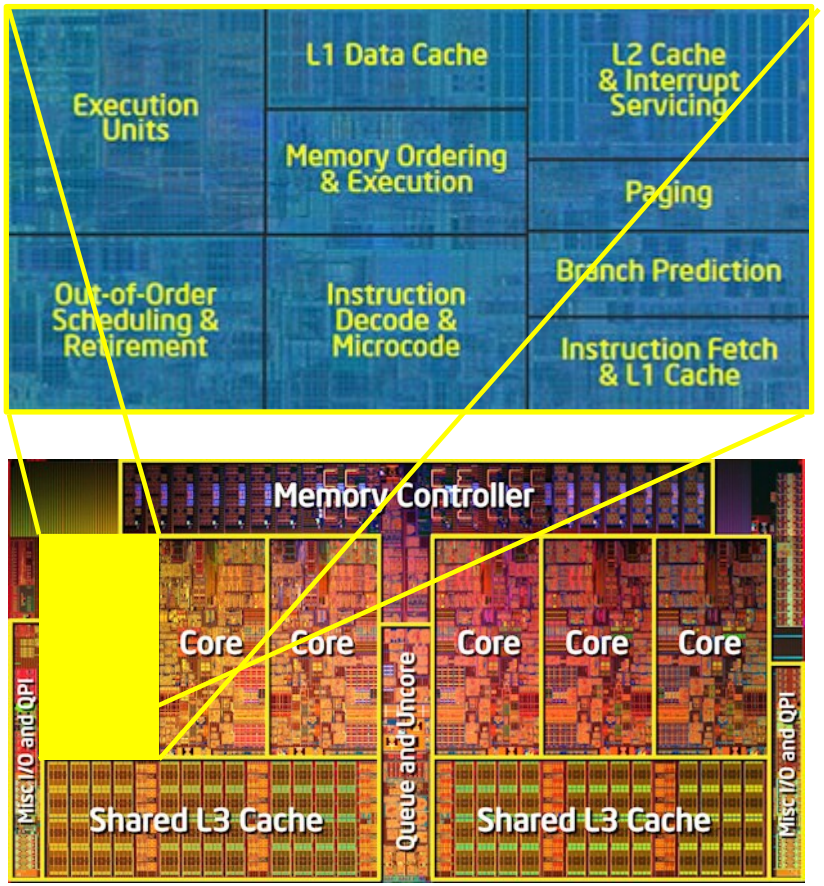


Modern Control-flow processor (CPU)



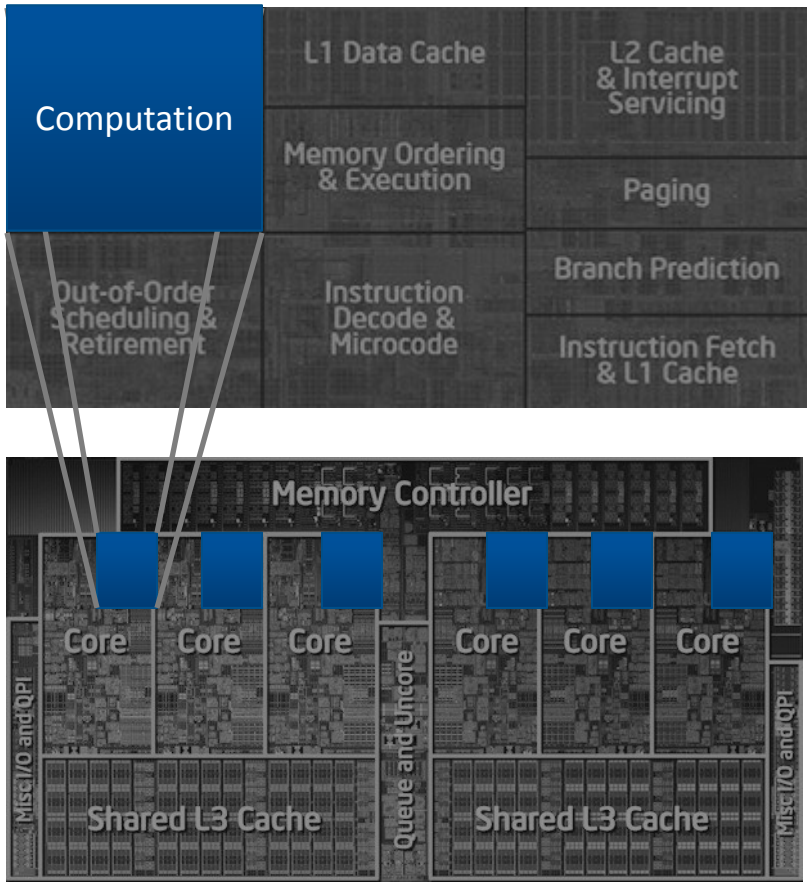
Where silicon is used?

Intel 6-Core X5680 “Westmere”

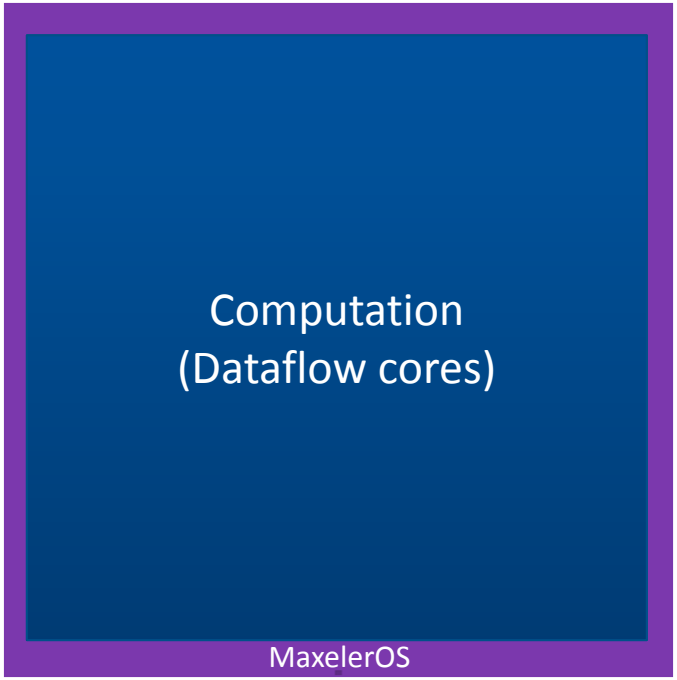


Where silicon is used?

Intel 6-Core X5680 “Westmere”

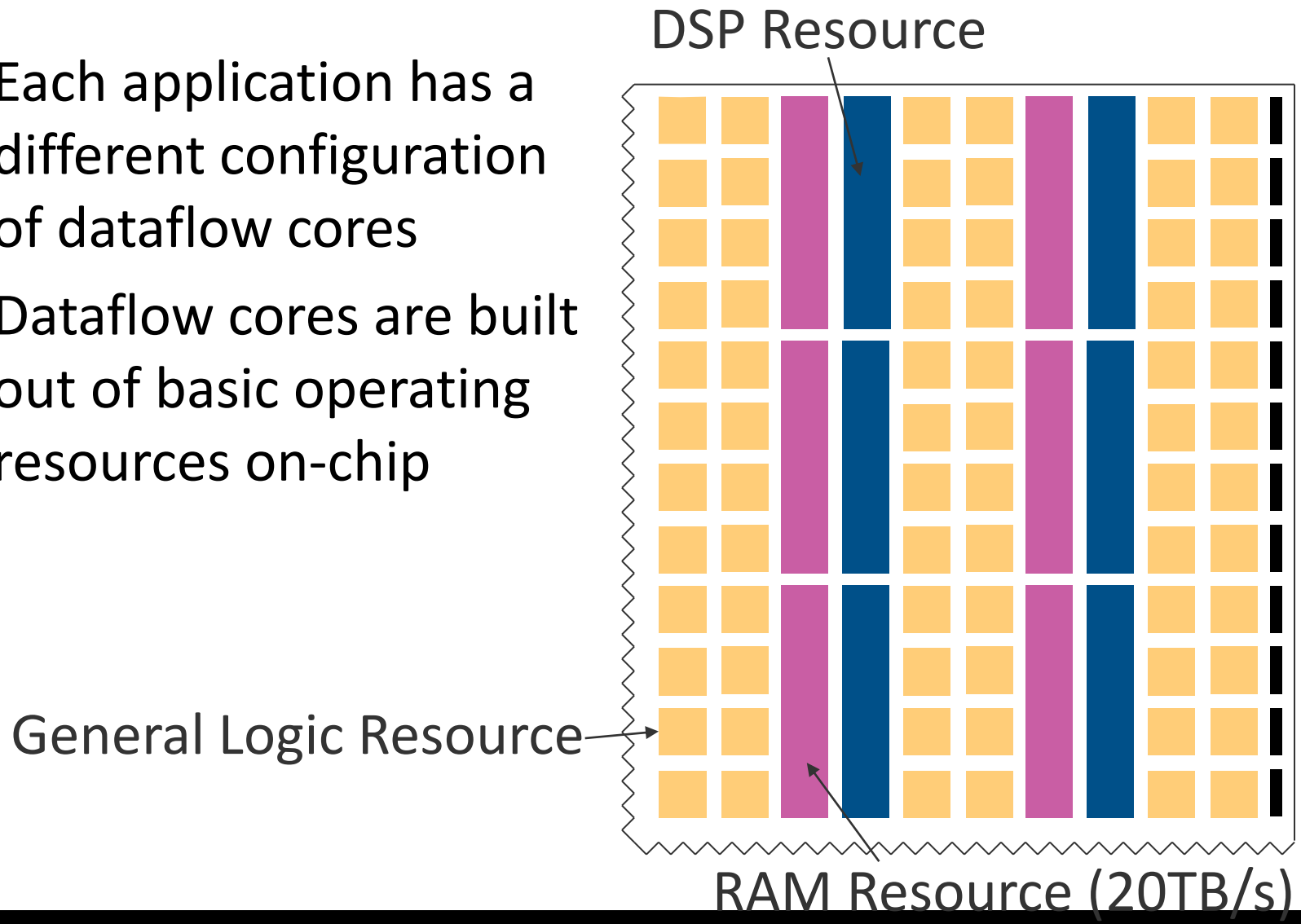


Dataflow Processor
e.g. FPGA

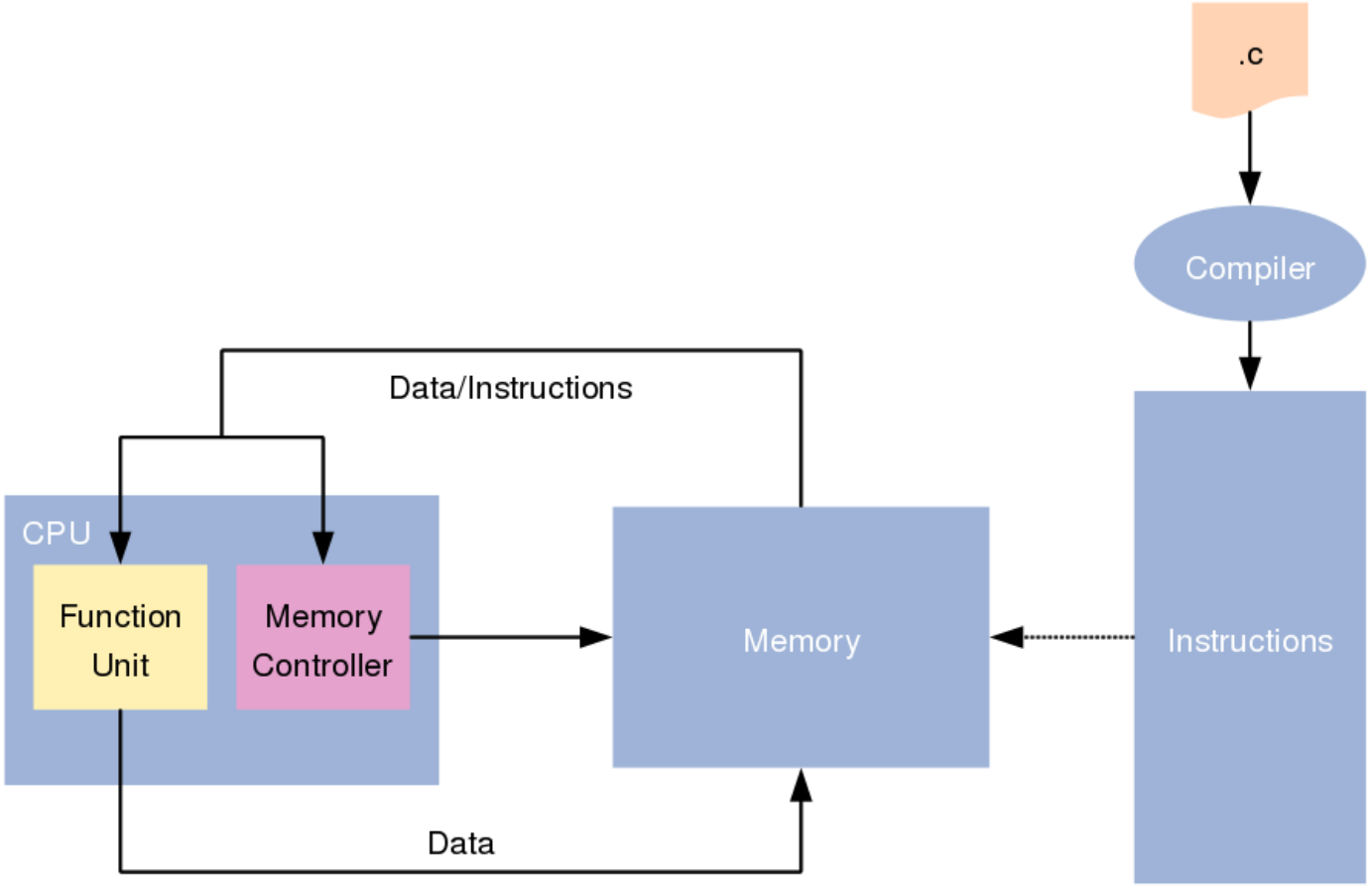


On chip resources

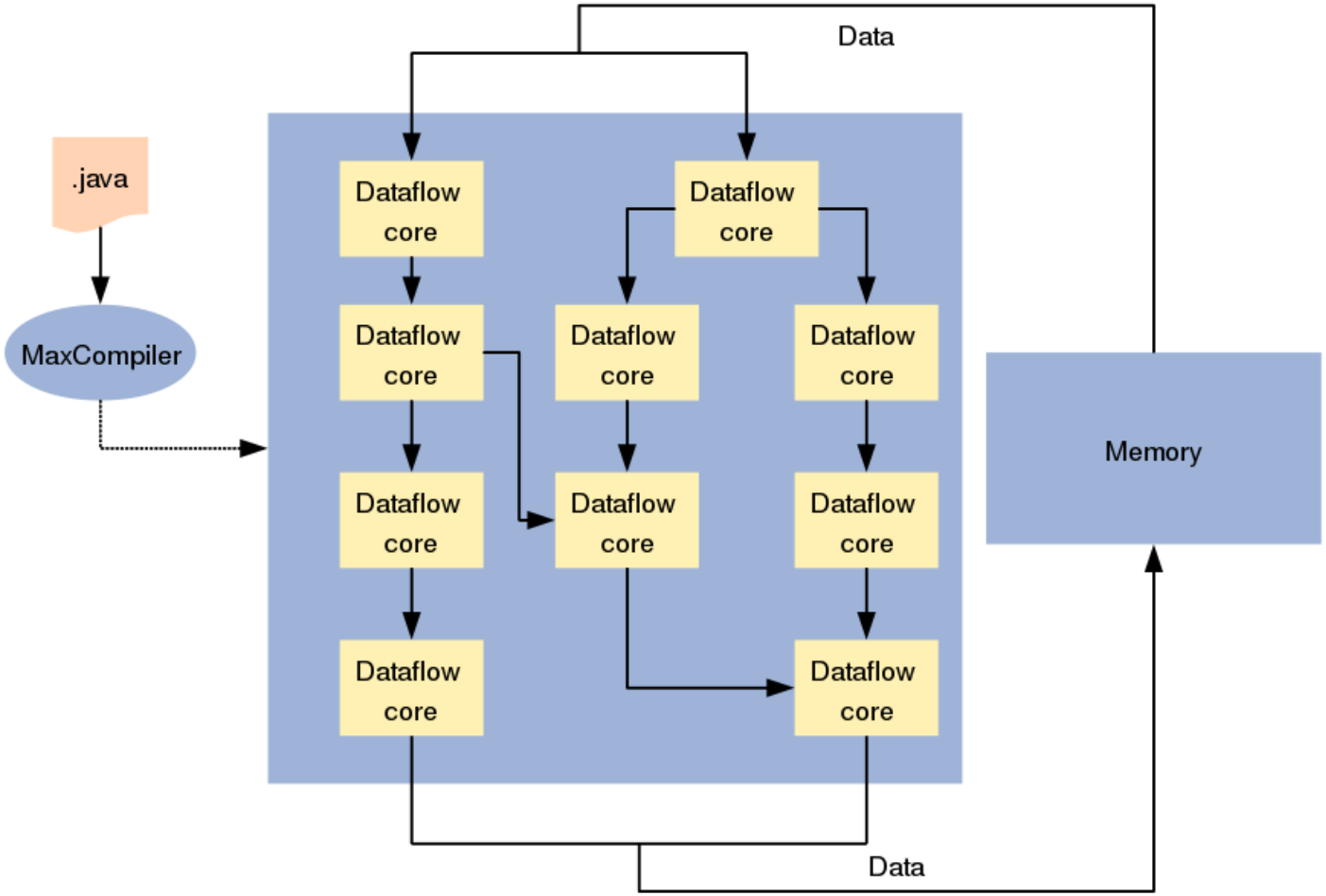
- Each application has a different configuration of dataflow cores
- Dataflow cores are built out of basic operating resources on-chip



Control flow Microprocessor (CPU)



Dataflow Engine (DFE) - ‘Spatial Computing’



Acceleration Potential

-10×

+100×

Dataflow

Processor Performance

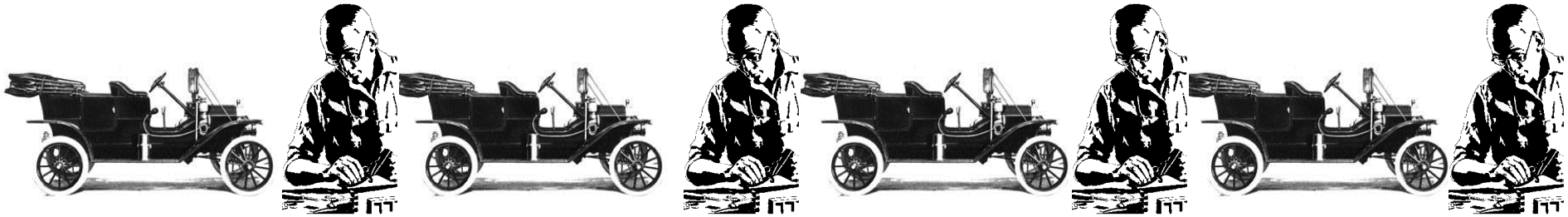
Dataflow

- Ten times slower clock

- Degrees of Freedom
  - Architecture
  - Data type
- Massive parallelism
  - Bit level
  - Pipeline level
  - Architecture level
  - System level

Explaining Control Flow versus Data Flow

Analogy 1: The Ford Production Line



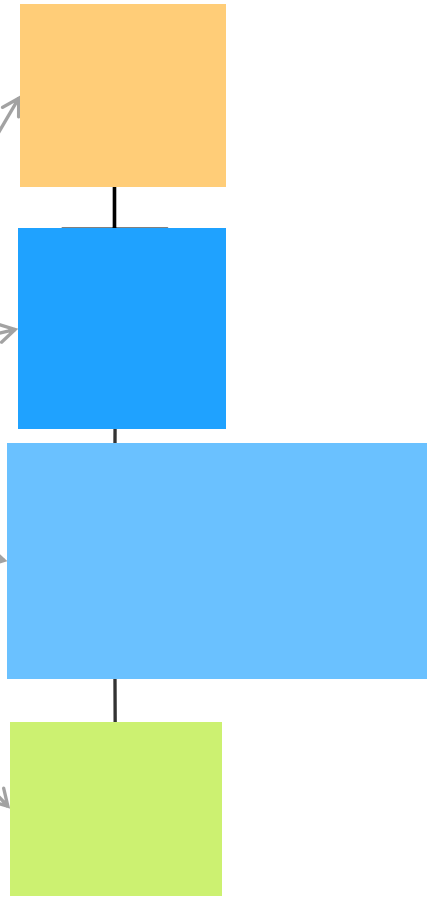
- Experts are expensive and slow (control flow)
- Many specialized workers are more efficient (data flow)

Dataflow Computing

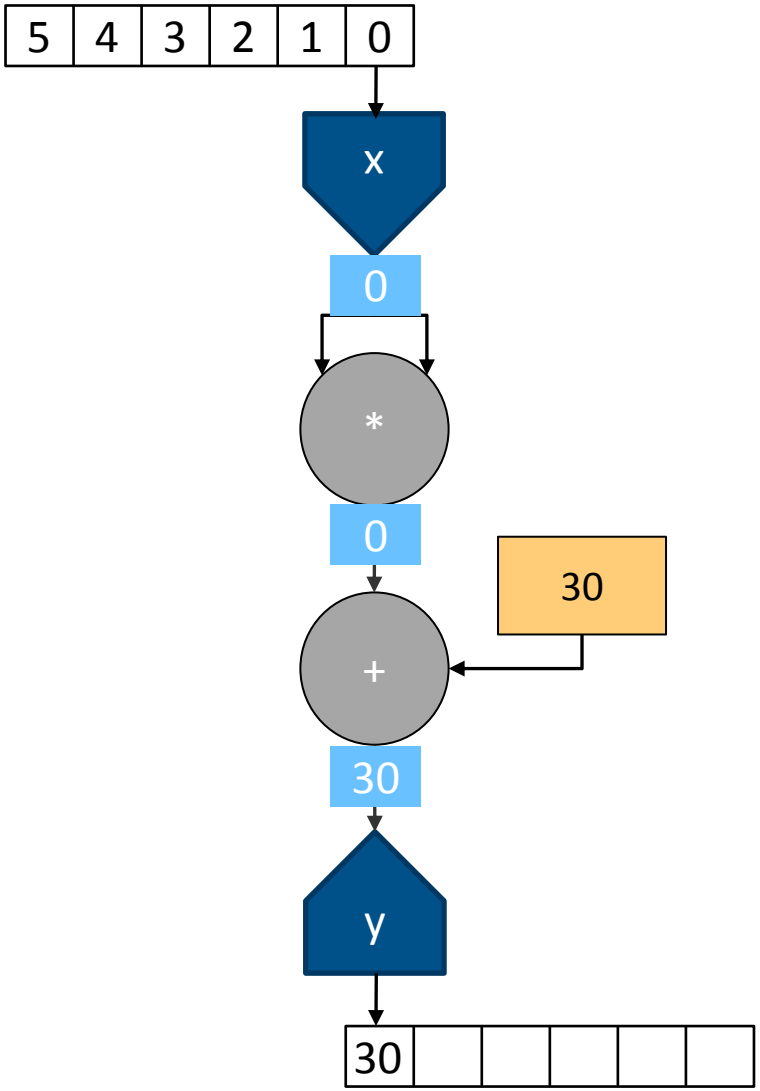
MAXELER
Technologies
MAXIMUM PERFORMANCE COMPUTING

A Dataflow Kernel

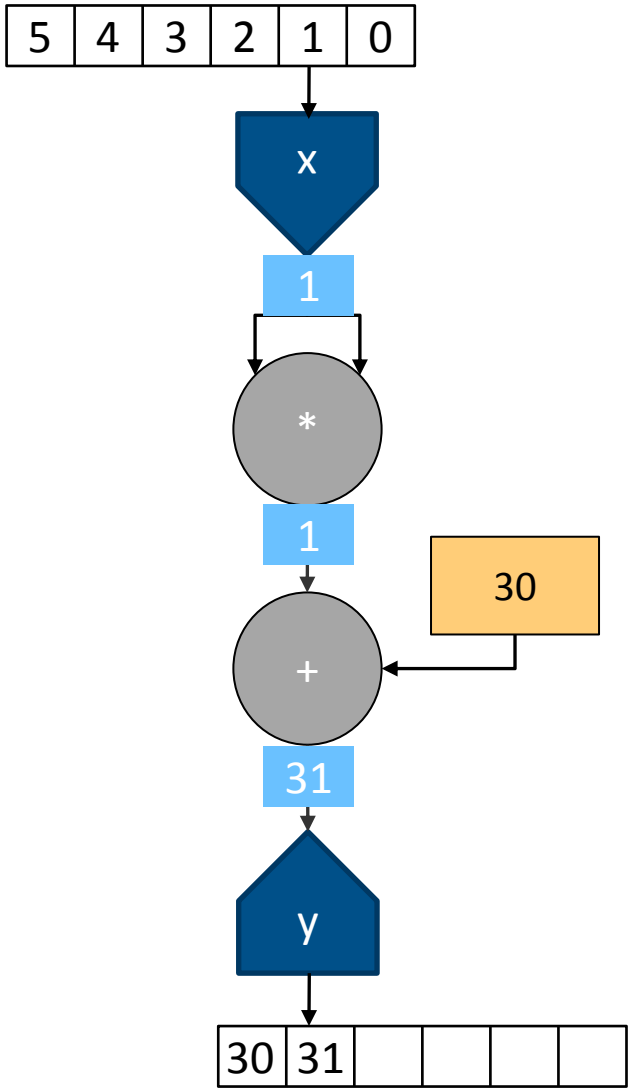
```
public class MyKernel extends Kernel {

    public MyKernel (KernelParameters parameters) {
        super(parameters);

        DFEVar x = io.input("x", dfeInt(32));

        DFEVar result = x * x + 30;

        io.output("y", result, dfeInt(32));
    }
}
```



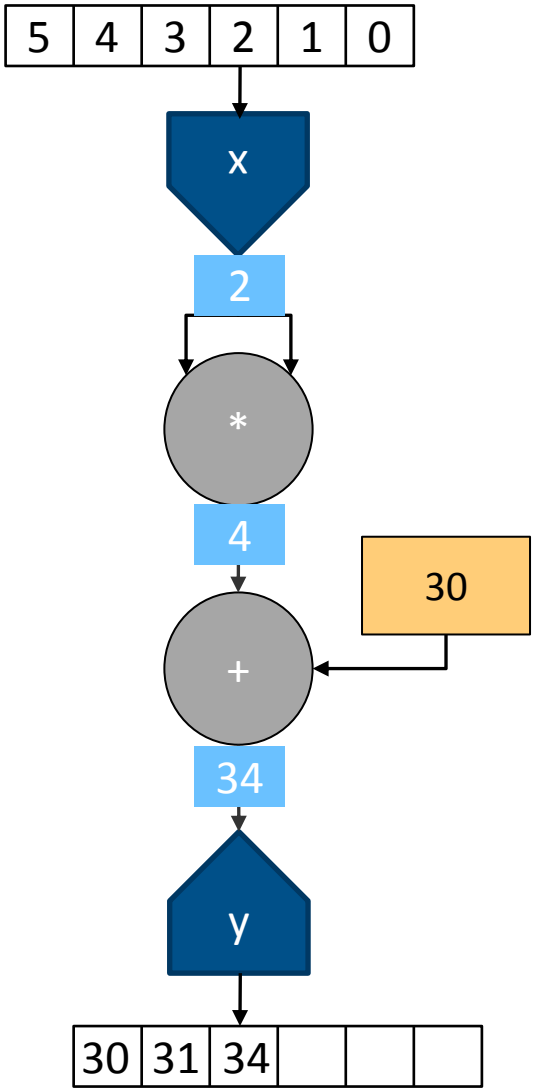
Streaming Data through the Kernel



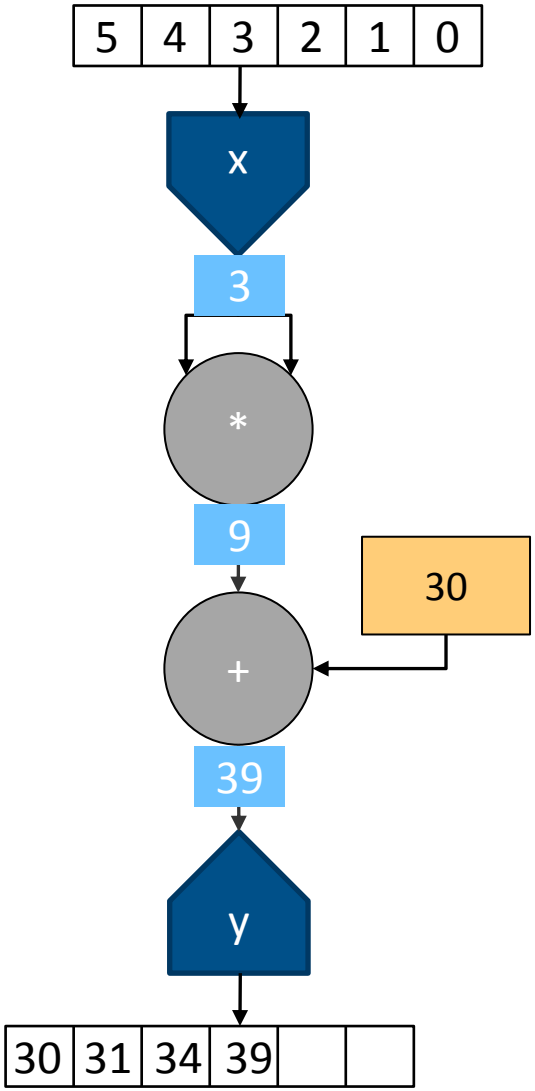
Streaming Data through the Kernel



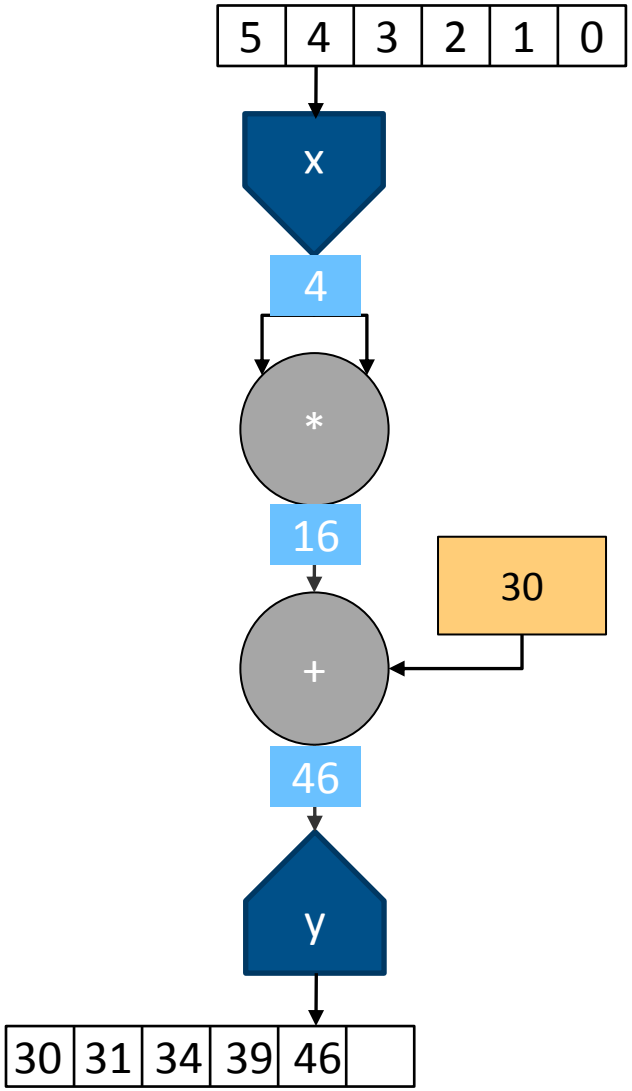
Streaming Data through the Kernel



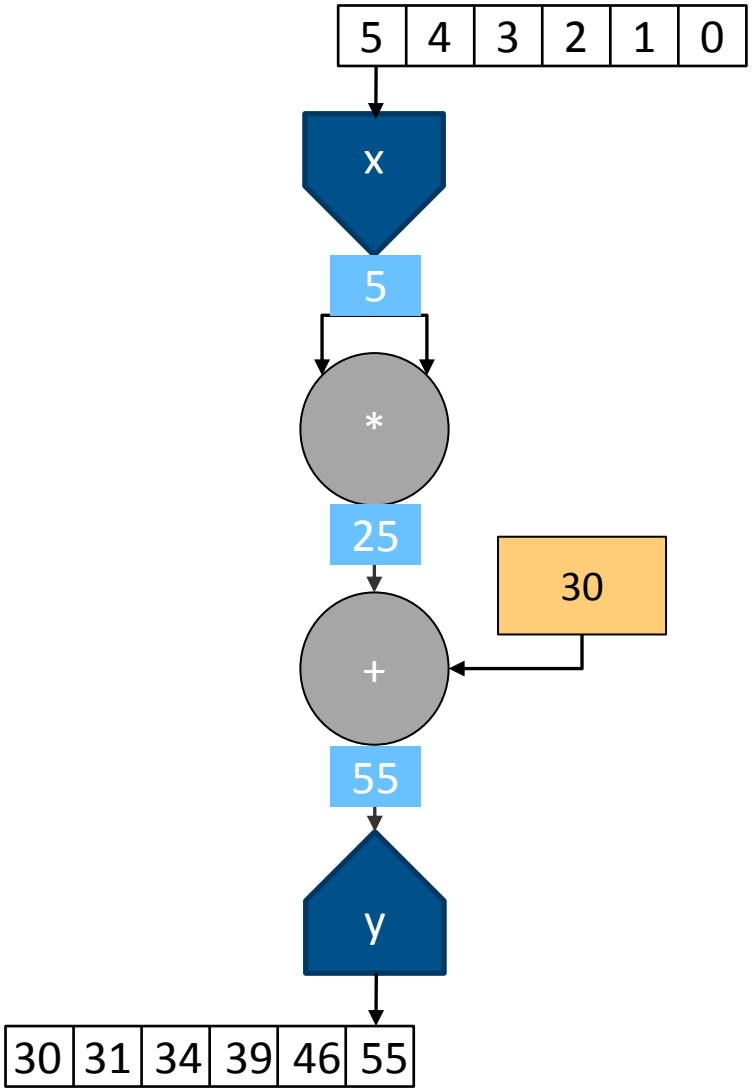
Streaming Data through the Kernel



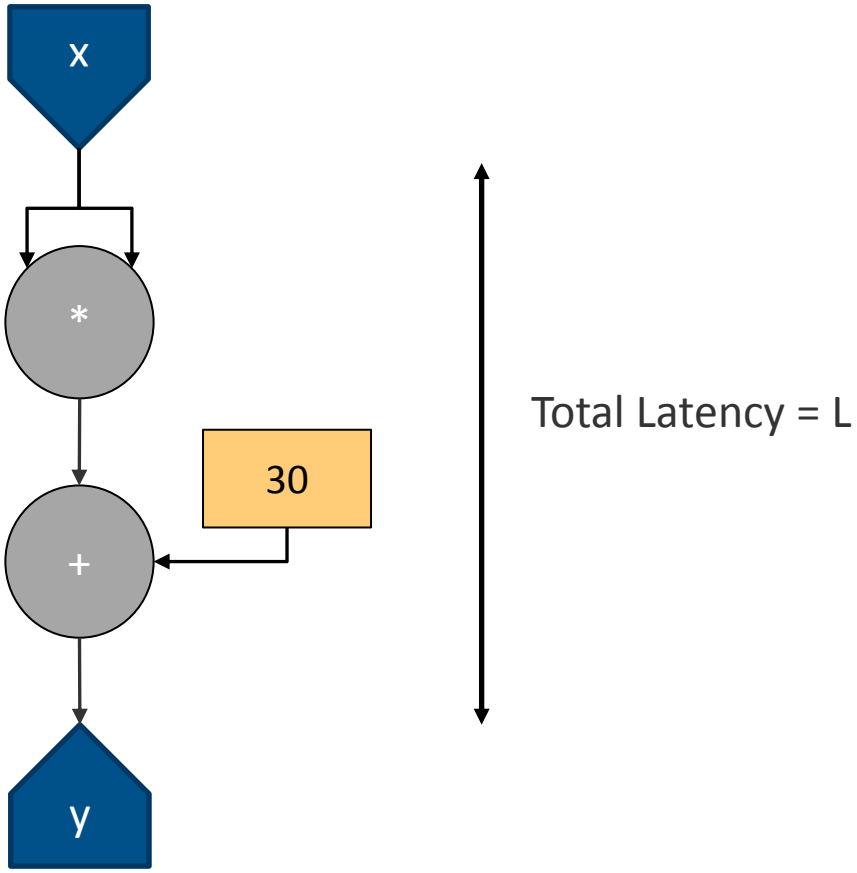
Streaming Data through the Kernel



Streaming Data through the Kernel

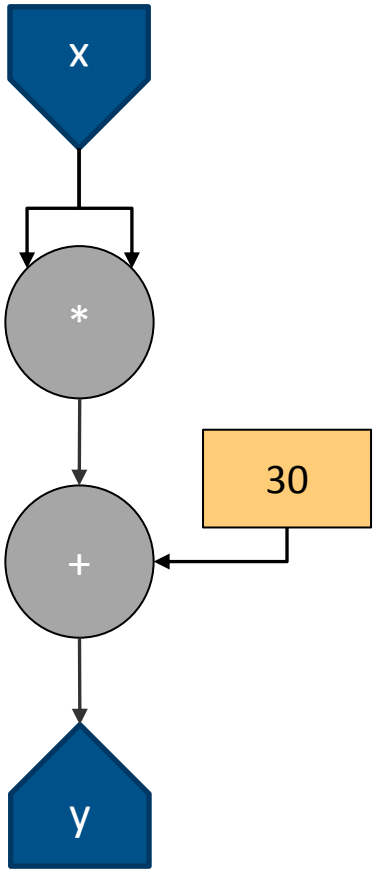


Streaming Data through the Kernel



Pipelining Data through the Kernel

Add registers into the Dataflow graph.



Fanout Latency = L1

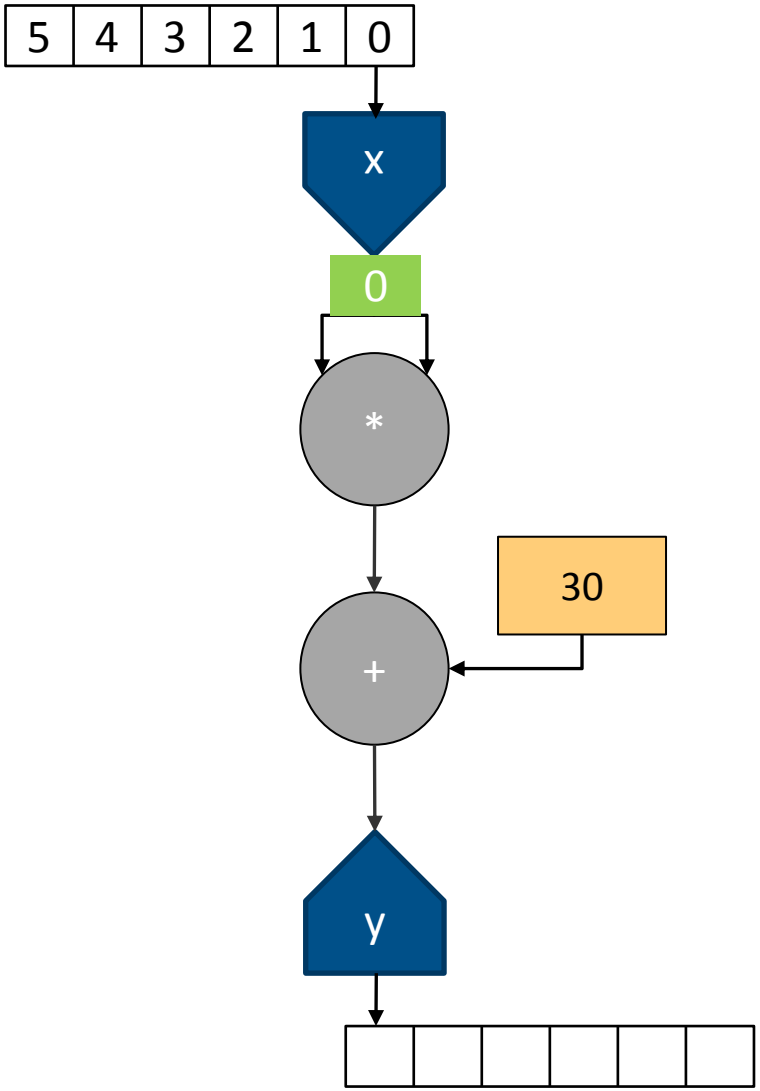
Multiply Latency = L2

Add Latency = L3

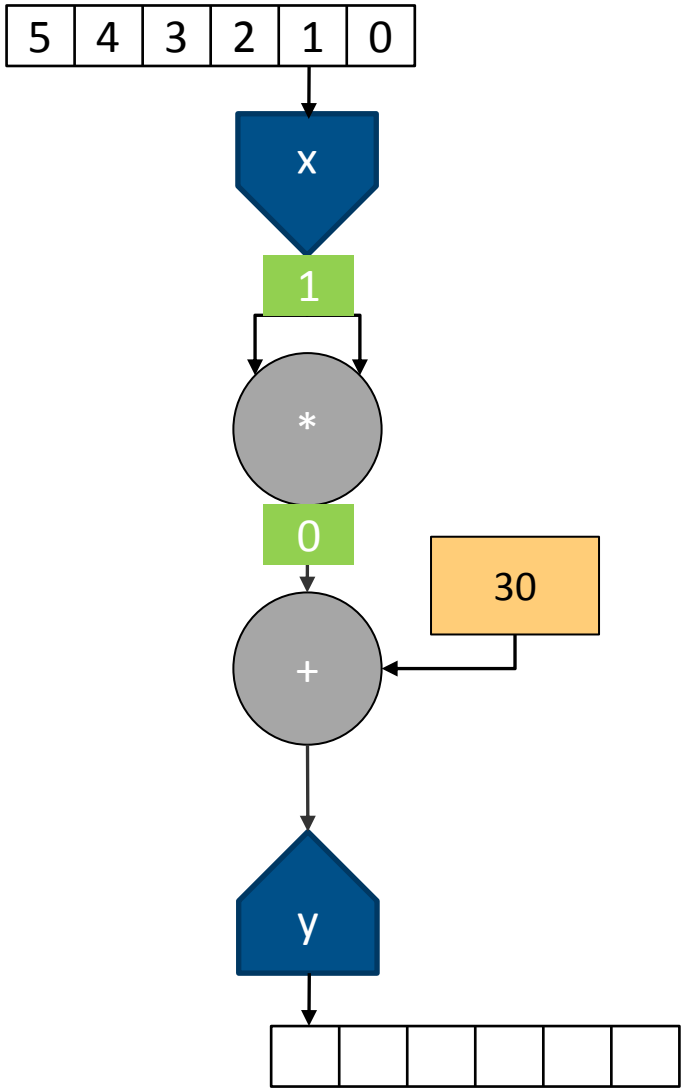
On FPGAs the shorter the physical distance between registers the higher the clock frequency can be.

L1 + L2 + L3 > L

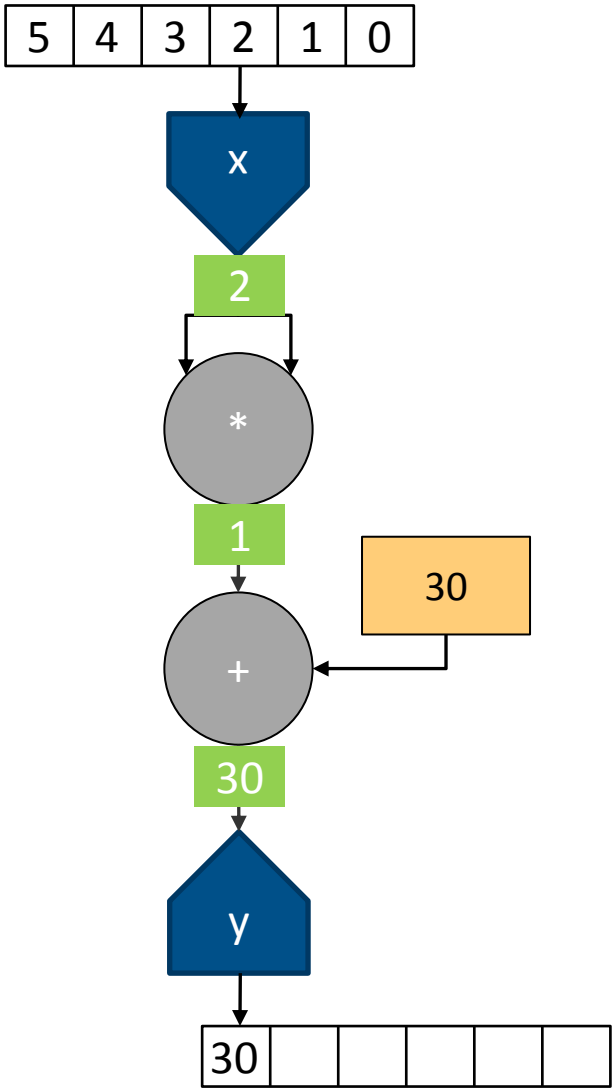
Pipelining Data through the Kernel



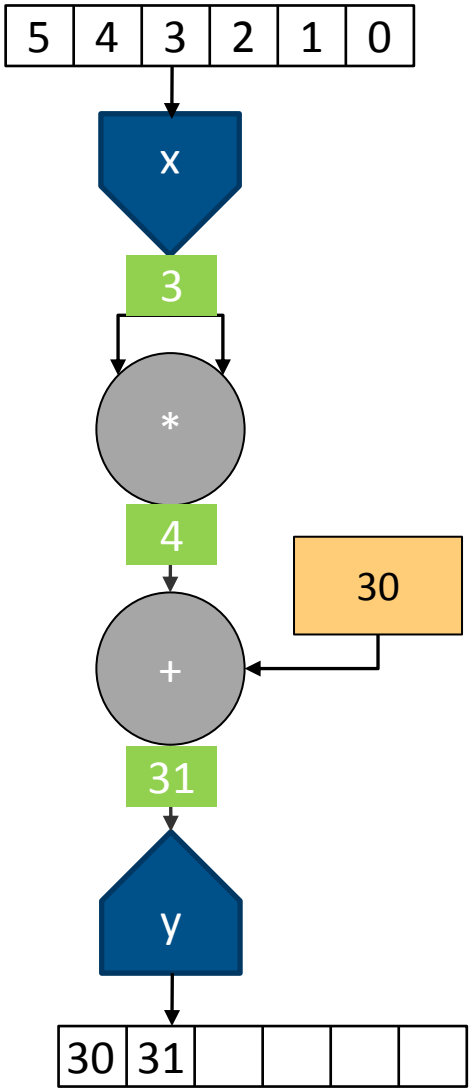
Pipelining Data through the Kernel



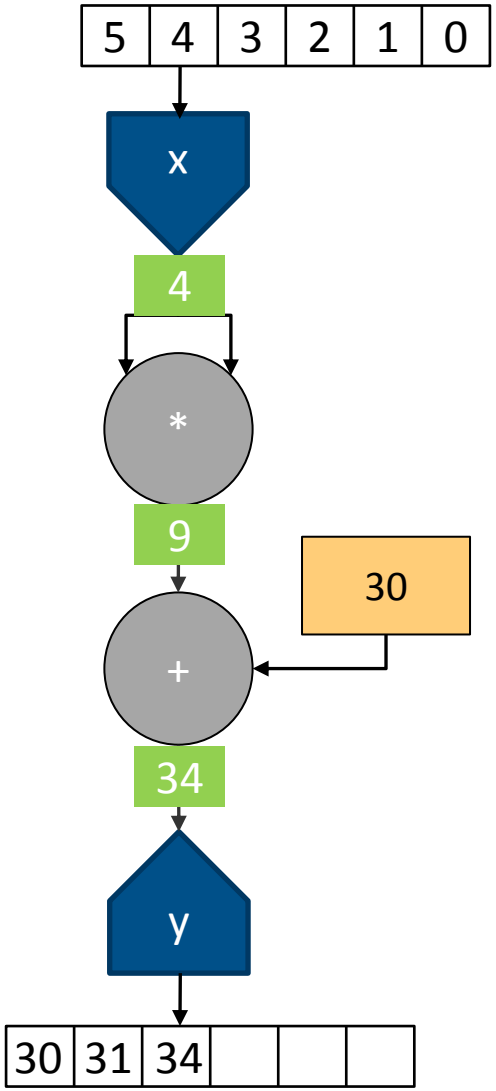
Pipelining Data through the Kernel



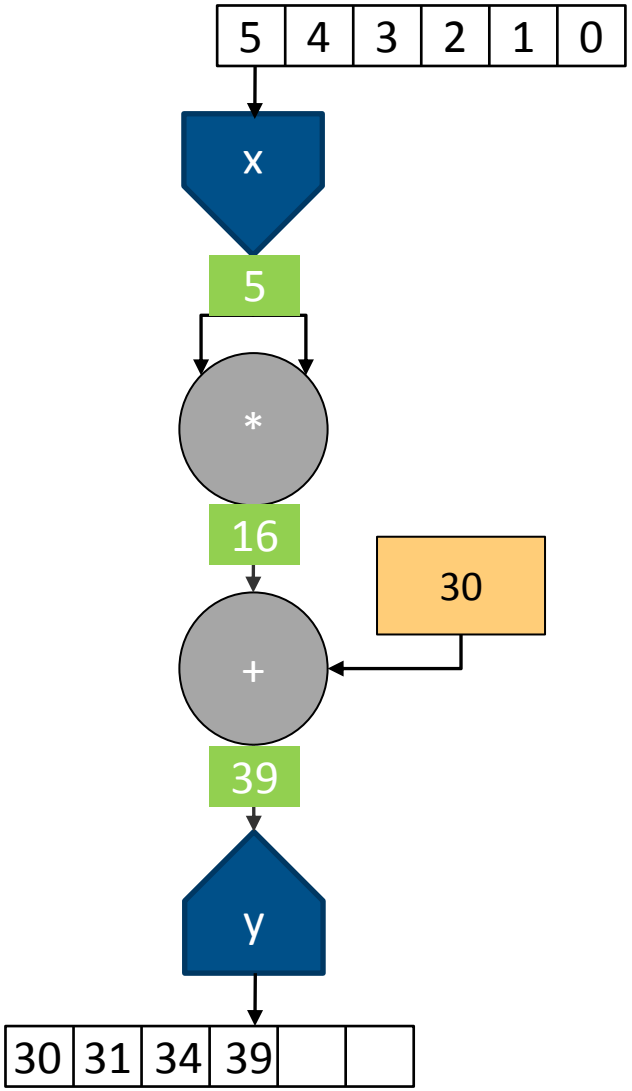
Pipelining Data through the Kernel



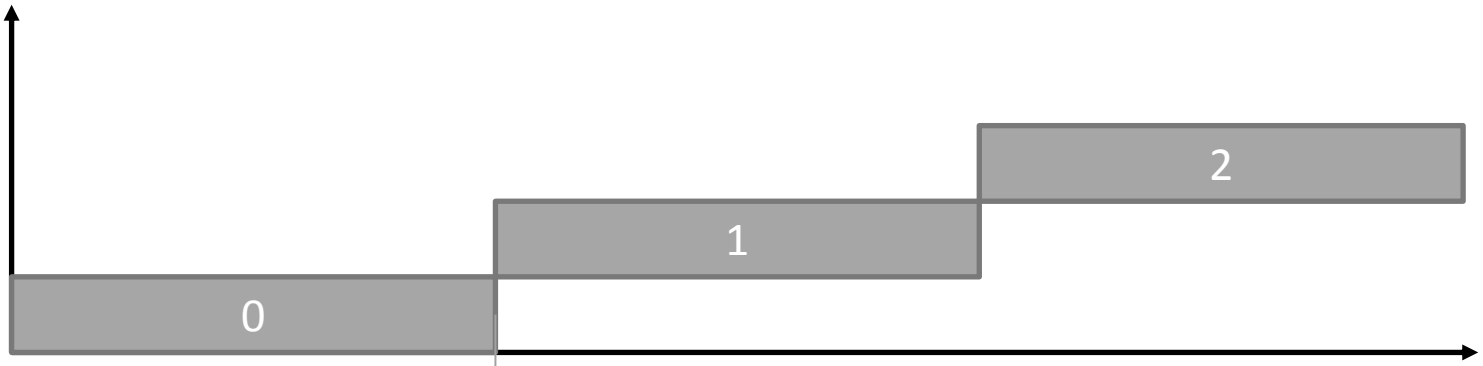
Pipelining Data through the Kernel



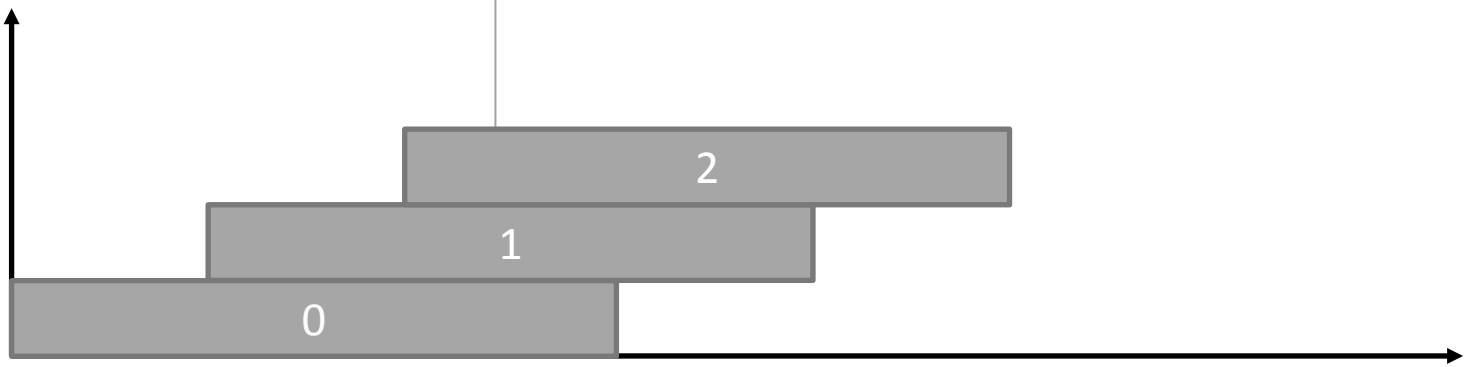
Pipelining Data through the Kernel



No registers – low latency



Pipelined – highly parallelised



Explaining Control Flow Versus Dataflow

Analogy 2: Trucks versus Pipeline

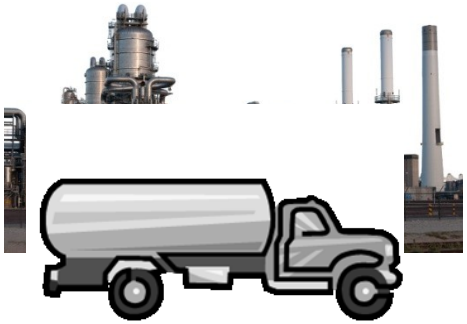
Processor

Oil Refinery

Click to advance to next slide

Memory

Oil Well



So we get a truck

We fetch the data in small chunks

Explaining Control Flow Versus Dataflow

Analogy 2: Trucks versus Pipeline

Processor

Oil Refinery

Click to advance to next slide

Memory

Oil Well

So we splash out on a Ferrari to carry our oil!

Explaining Control Flow Versus Dataflow

Analogy 2: Trucks versus Pipeline

Processor

Oil Refinery

Click to advance to next slide

Memory

Oil Well

Once we starting pumping, it takes a while to fill up...

The latency of the first result can be high...

Explaining Control Flow Versus Dataflow

Analogy 2: Trucks versus Pipeline

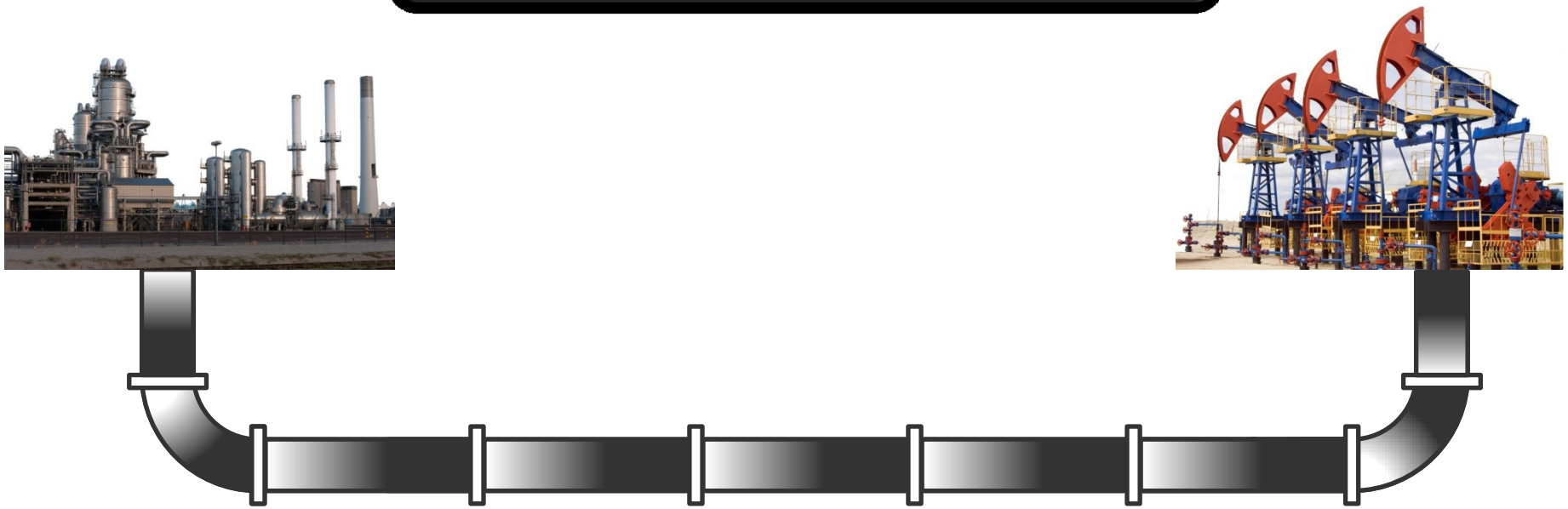
Processor

Oil Refinery

Click to advance to next slide

Memory

Oil Well



But then the oil flows constantly.

And we get a result every clock cycle.

Using FPGAs

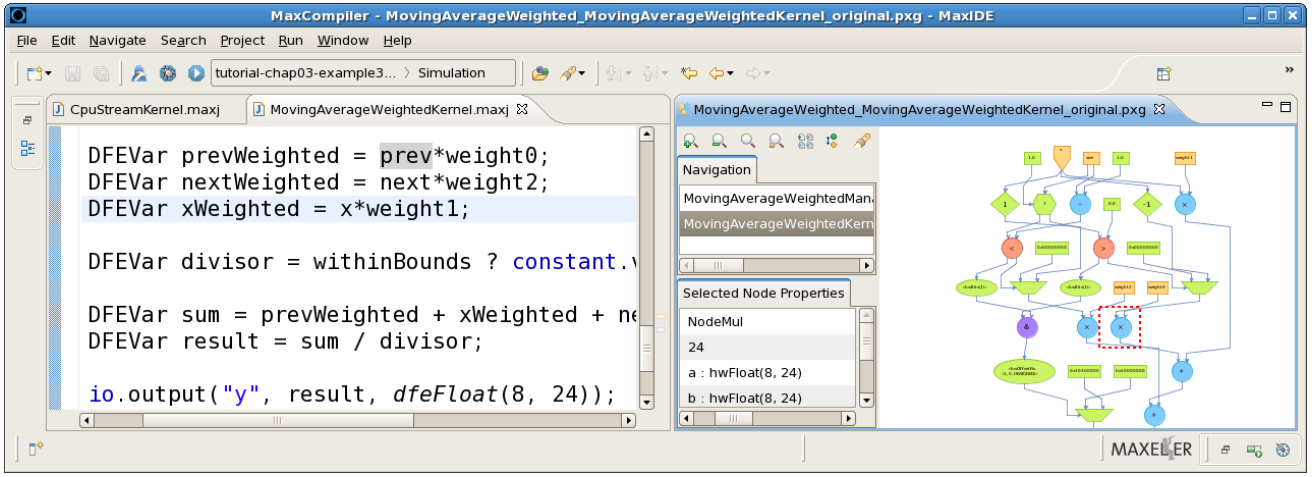
MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Traditionally FPGA were for specialists

- Can’t use FPGA on its own …
  - Need interface to Computer & Data
  - Need interface to local memory
  - Need to design bespoke HW
- FPGAs difficult to program …
  - Specialist languages VHDL, Verilog
  - Need Electronics training & understand FPGAs
  - Simulation only at HW level, modelsim
  - Need also to engineer the interfaces
- FPGAs difficult to use …
  - Need low level drivers to reconfigure and setup

Maxeler solutions

Standard Hardware across FPGA generations and vendors.



Development environment in extended Java language, MaxJ, with fast simulation and debugging tools.

Runtime library and drivers for reconfiguration, monitoring and probing.

```
>maxtop -r 10.101.101.33
MaxTop Tool 2014.1
Found 2 Maxeler card(s) running MaxelerOS 2014.1
Card 0: Maia (P/N: 4848) S/N: 2487402010013 Mem: 48GB
Card 1: Maia (P/N: 4848) S/N: 2487402010049 Mem: 48GB

Load average: 0.12, 0.02, 0.01

DFE  %BUSY  MAXFILE      HOST         PID    USER       TIME      COMMAND
 0   0.0%   feda2885...  thor.cluster 27947  psanders   00:00:11  async_sessio
 1   0.0%   IDLE (r7)    -            -      -          -         -
```

DFE Hardware

Maxeler Data Flow Engines (DFEs)

- Stream to/from Host (PCIe)
- Stream to/from DRAM
- Stream to/from DFE (MaxRing)

MAIA

- 28nm process
- 250MHz clock frequency
- 6.25MB SRAM
- 4,000 multipliers
- 700K logic cells
- 3GB/s CPU bandwidth
- 96GB DRAM

- Stream to/from network
  - UDP & TCP

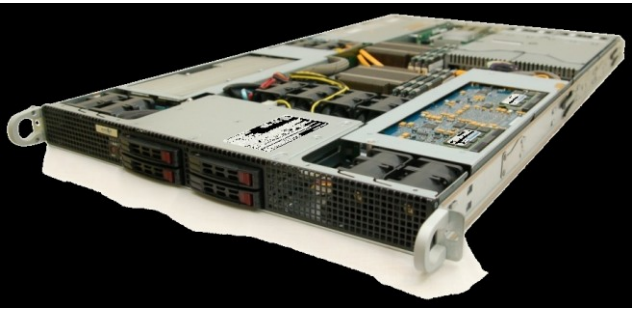
ISCA

- 28nm process
- 250MHz clock
- Sub 1us latency
- 16K TCP connections

Maxeler Hardware Solutions

MPC C Series

CPUs plus DFEs
Intel Xeon CPU cores and up to 6 DFEs with 288GB of RAM



MPC X Series

DFEs shared over Infiniband
Up to 8 DFEs with 384GB of RAM and dynamic allocation of DFEs to CPU servers

MPC N Series

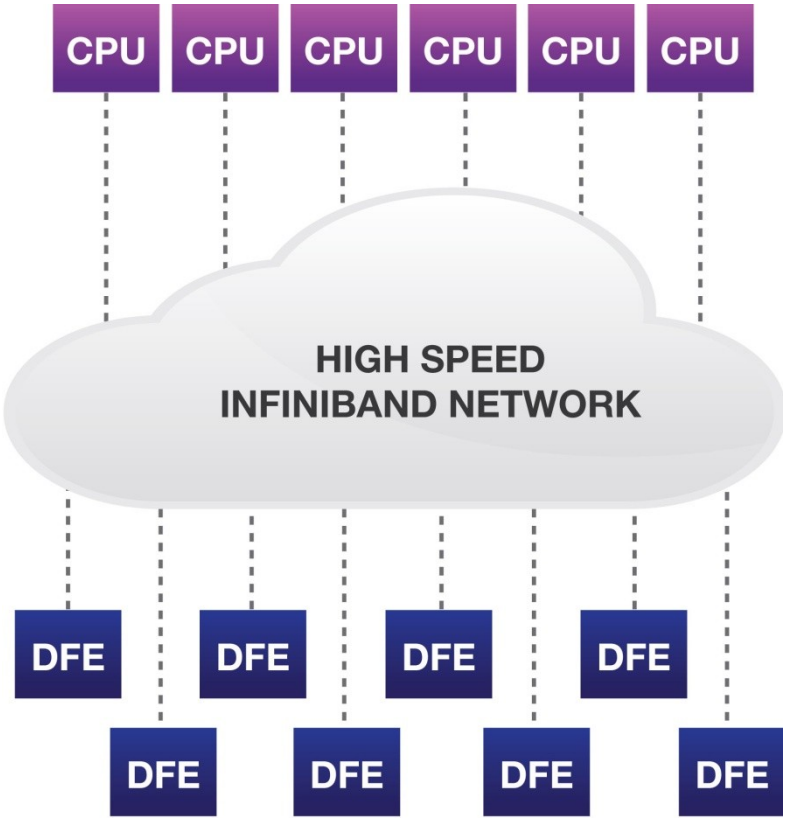
Low latency connectivity
Intel Xeon CPUs and 1-2 DFEs with up to six 10Gbit Ethernet connections

MaxWorkstation
Desktop development system

MaxCloud
On-demand scalable accelerated compute resource, hosted in London

MPC-X2000

- 8 dataflow engines (192-384GB RAM)
- High-speed MaxRing
- Zero-copy RDMA between CPUs and DFEs over Infiniband
- Dynamic CPU/DFE balancing



STFC

Hartree Centre

Software

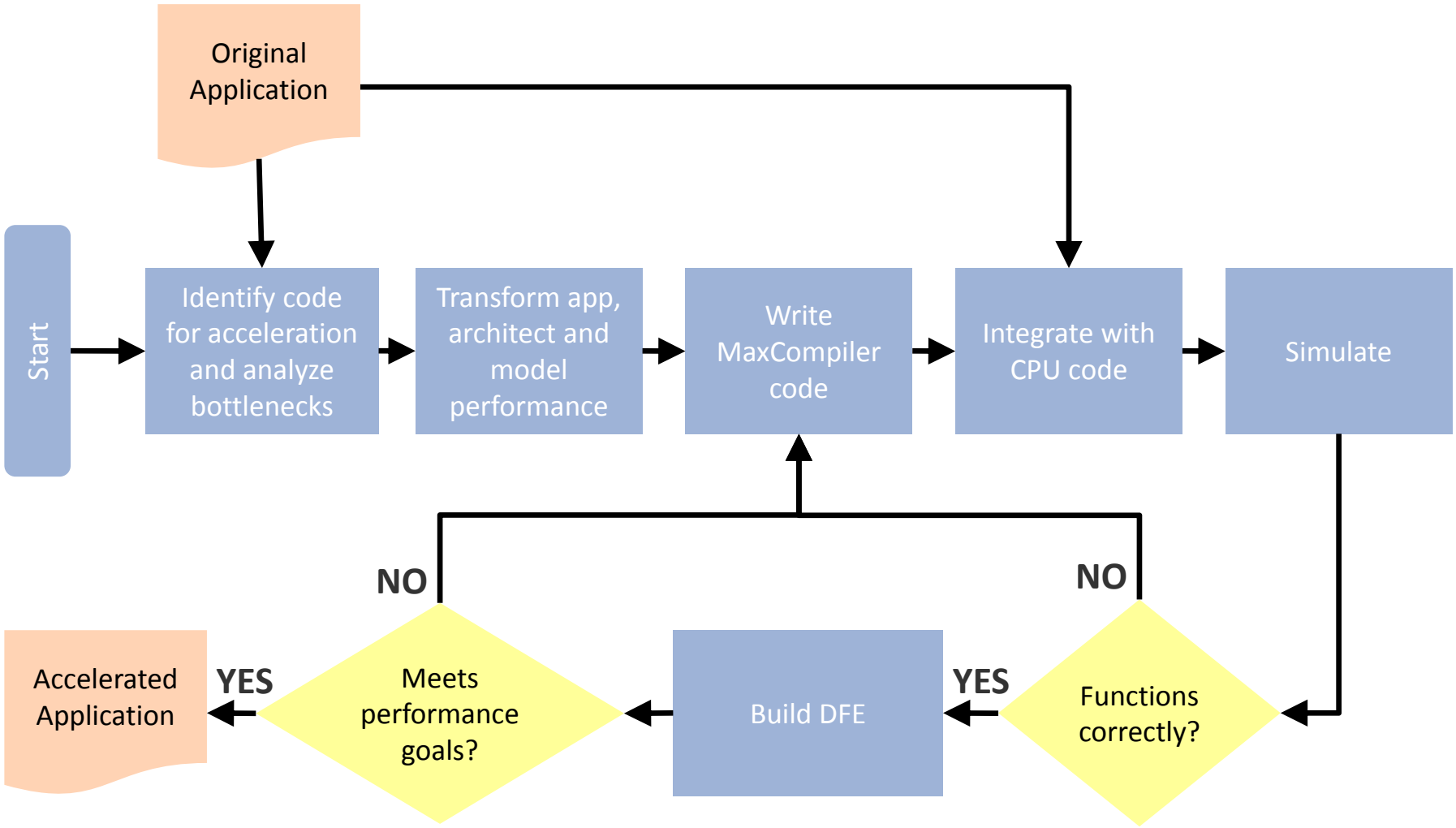
Accelerating Real Applications

- The majority of lines of code in most applications are unchanged
- CPUs are good for: latency-sensitive, control-intensive, non-repetitive code
- Dataflow engines are good for: high throughput repetitive processing on large data volumes

➔ A system should contain both

	Lines of code
Total Application	1,000,000
Kernel to accelerate	2,000
Software to restructure	20,000

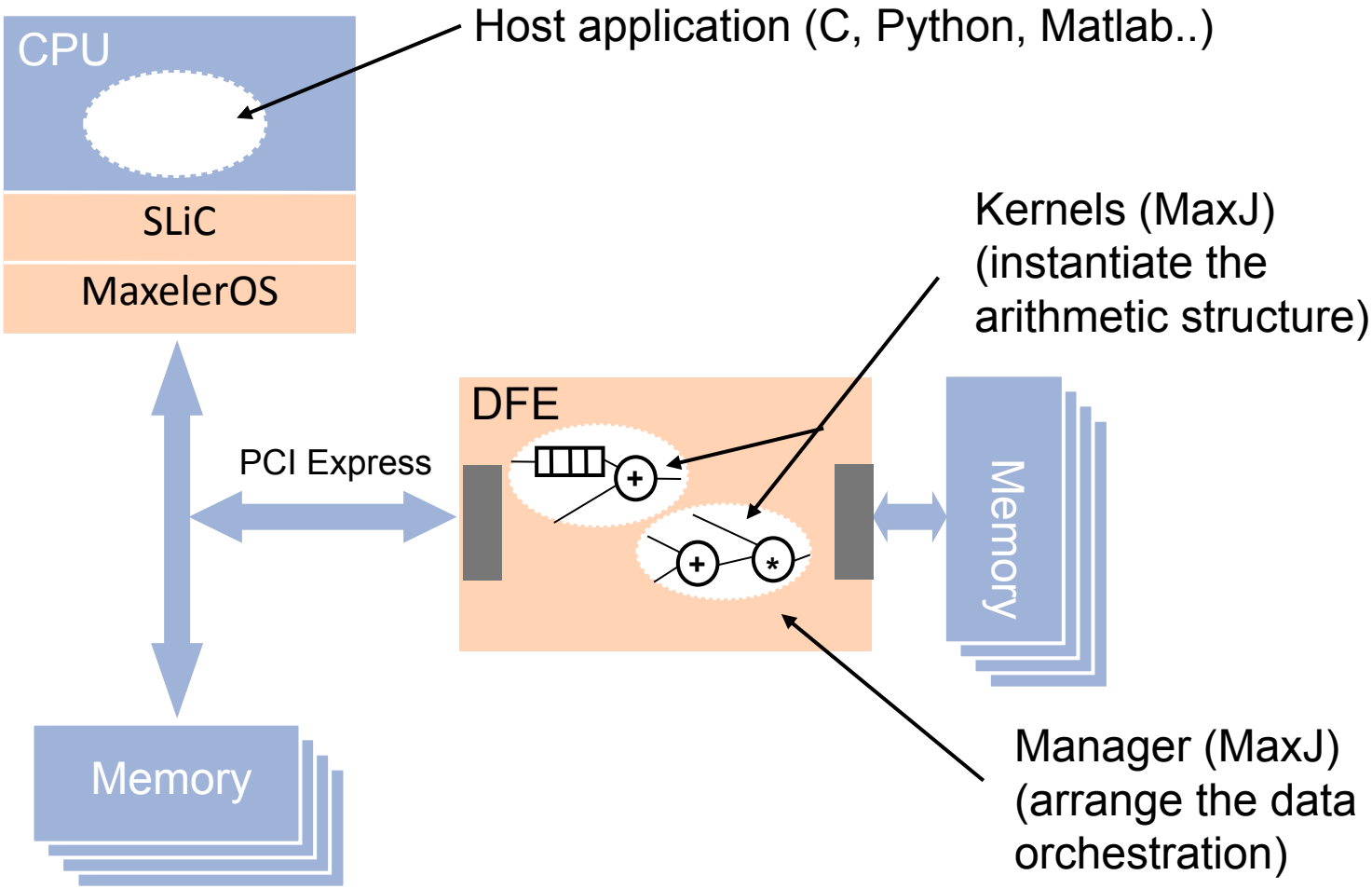
Creating custom DFE configurations



MaxCompiler & MaxIDE

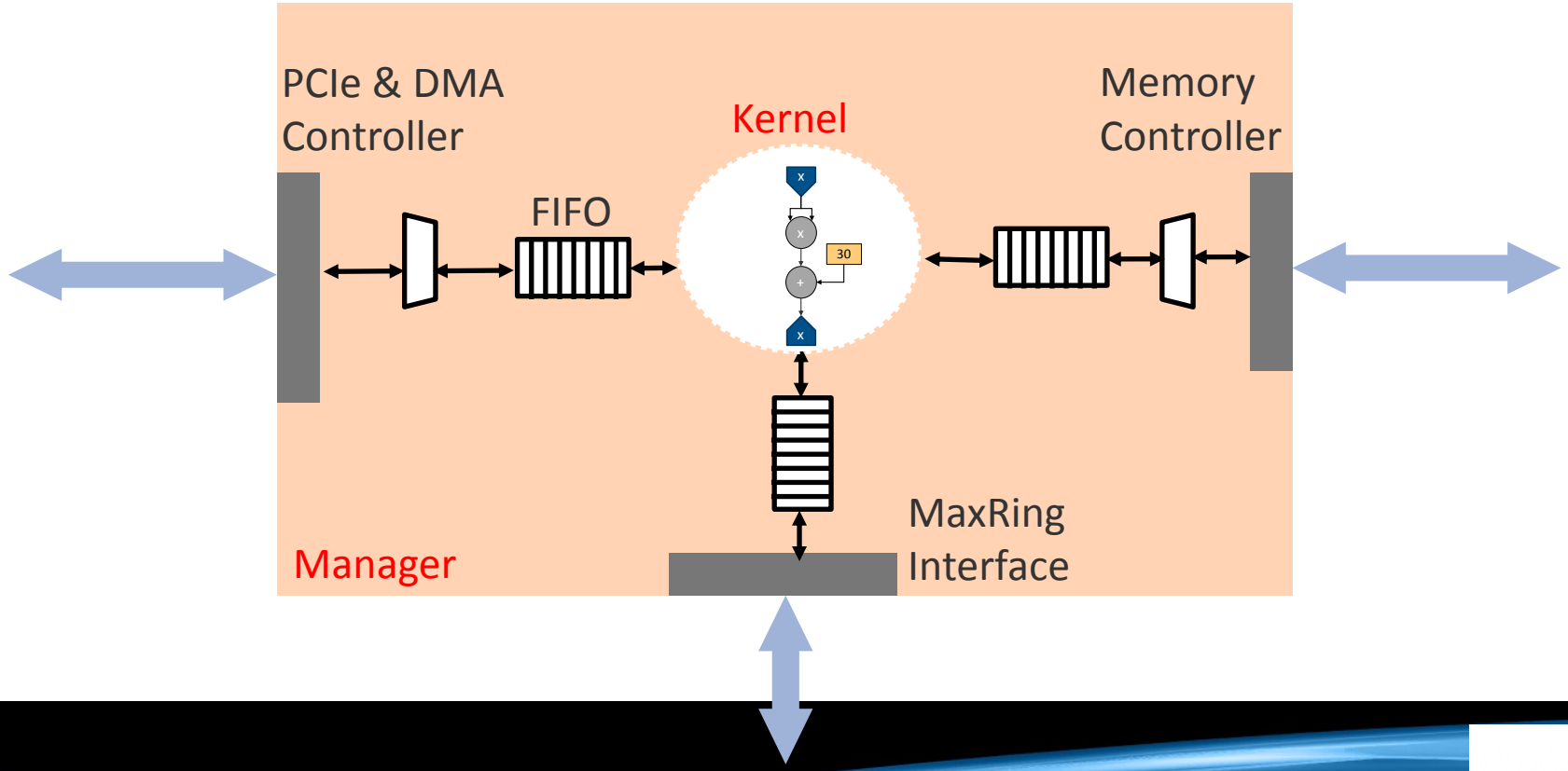
- Complete development environment for Maxeler DFE accelerator platforms
- Write *MaxJ* code to describe the dataflow graph
  - *MaxJ* is an extension of Java for MaxCompiler
  - *Execute* the Java to *generate* the DFE image (bitstream)
  - Meta-programming. Java does NOT execute when running final application.
- Compiler generates C API for CPUs to use *the DFE*
  - *C API called SLiC*
  - *Basic Static interface – single function*
    - *loads DFE with bitstream*
    - *Sets scalars and streams data in/out*

Application Components



DFE contains a Manager and Kernels

- Globally Asynchronous Locally Synchronous (GALS) architecture.
- Manager has full flow control.
- Manager made from standard blocks.
- Kernels are fully synchronous.
- Kernels runs while data at inputs and space at output, else stalls.



The required parts to create a DFE

MaxCompiler - StructureExample_EngineCode/src/cpustream/CpuStreamManager.maxj - MaxIDE

File Edit Source Refactor Navigate Search Project Run Window Help

tutorial-chap03-example3... 〉 Simulation

*CpuStreamCpuCode.c

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    const int size = 384;
    int sizeBytes = size * sizeof(int32_t);
    int32_t *x = malloc(sizeBytes);
    int32_t *y = malloc(sizeBytes);
    int32_t *s = malloc(sizeBytes);

    for(int i = 0; i < size; ++i) {
        x[i] = random() % 100;
        y[i] = random() % 100;
    }

    printf("Running on DFE.\n");
    int scalar = 3;
    CpuStream(scalar, size, x, y, s);

    for(int i = 0; i < size; ++i)
        if ( s[i] != x[i] + y[i] + scalar)
            return 1;

    printf("Done.\n");
    return 0;
}
```

*CpuStreamManager.maxj

```java
public class CpuStreamManager {

    private static final String s_kernelName = "CpuStreamKernel";

    public static void main(String[] args) {
        CpuStreamEngineParameters params = new CpuStreamEngineParame
        Manager manager = new Manager(params);
        Kernel kernel   = new CpuStreamKernel(manager.makeKernelPara
        manager.setKernel(kernel);
        manager.setIO(
            link("x",   IODestination.CPU),
            link("y",   IODestination.CPU),
            link("s", IODestination.CPU));

        manager.createSLiCinterface(interfaceDefault());

        configBuild(manager, params);

        manager.build();
    }

    private static EngineInterface interfaceDefault() {
        EngineInterface engine_interface = new EngineInterface();
        CPUTypes   type = CPUTypes.INT32;
        int        size = type.sizeInBytes();

        InterfaceParam  a    = engine_interface.addParam("A", CPUTyp
        InterfaceParam  N    = engine_interface.addParam("N", CPUTyp

        engine_interface.setScalar(s_kernelName, "a", a);

        engine_interface.setTicks(s_kernelName, N);
        engine_interface.setStream("x",   type, N * size);
        engine_interface.setStream("y",   type, N * size);
        engine_interface.setStream("s", type, N * size);
        return engine_interface;
    }

    private static void configBuild(Manager manager, CpuStreamEngine
        manager.setEnableStreamStatusBlocks(false);
        BuildConfig buildConfig = manager.getBuildConfig();
        buildConfig.setMPPRCostTableSearchRange(params.getMPPRStartC
        buildConfig.setMPPRParallelism(params.getMPPRThreads());
        buildConfig.setMPPRRetryNearMissesThreshold(params.getMPPRRe
    }
}
```

*CpuStreamKernel.maxj

```java
class CpuStreamKernel extends Kernel {

    private static final SCSType type = scsInt(32);

    protected CpuStreamKernel(KernelParameters parameters) {
        super(parameters);

        SCSVar x = io.input("x", type);
        SCSVar y = io.input("y", type);
        SCSVar a = io.scalarInput("a", type);

        SCSVar sum = x + y + a;

        io.output("s", sum, type);
    }

}
```

Writable | Smart Insert | 47 : 1

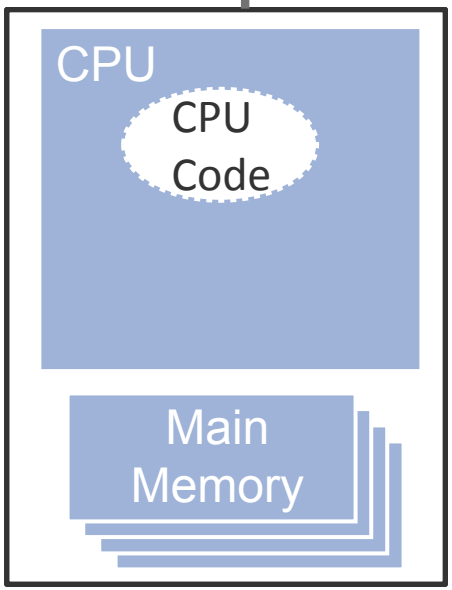
CPU Code

Manager Code

Kernel Code

Technologies

Simple Application Example

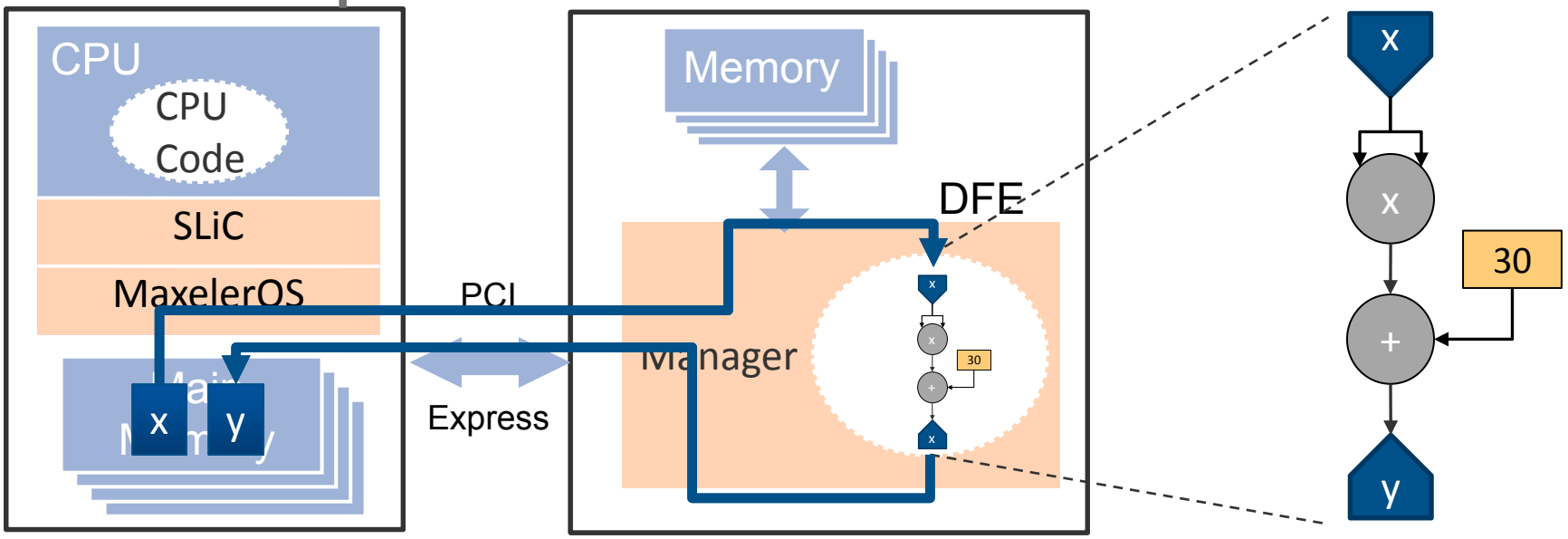


Host Code (.c)

```
int*x, *y;
for (int i =0; i < DATA_SIZE; i++)
   y[i]= x[i] * x[i] + 30;
```

$$y_i = x_i \times x_i + 30$$

Development Process



Host Code (.c)

```
int*x, *y;
for (int i =0; i < DATA_SIZE; i++)
   y[i]= x[i] * x[i] + 30;
                y, DATA_SIZE*4);
```

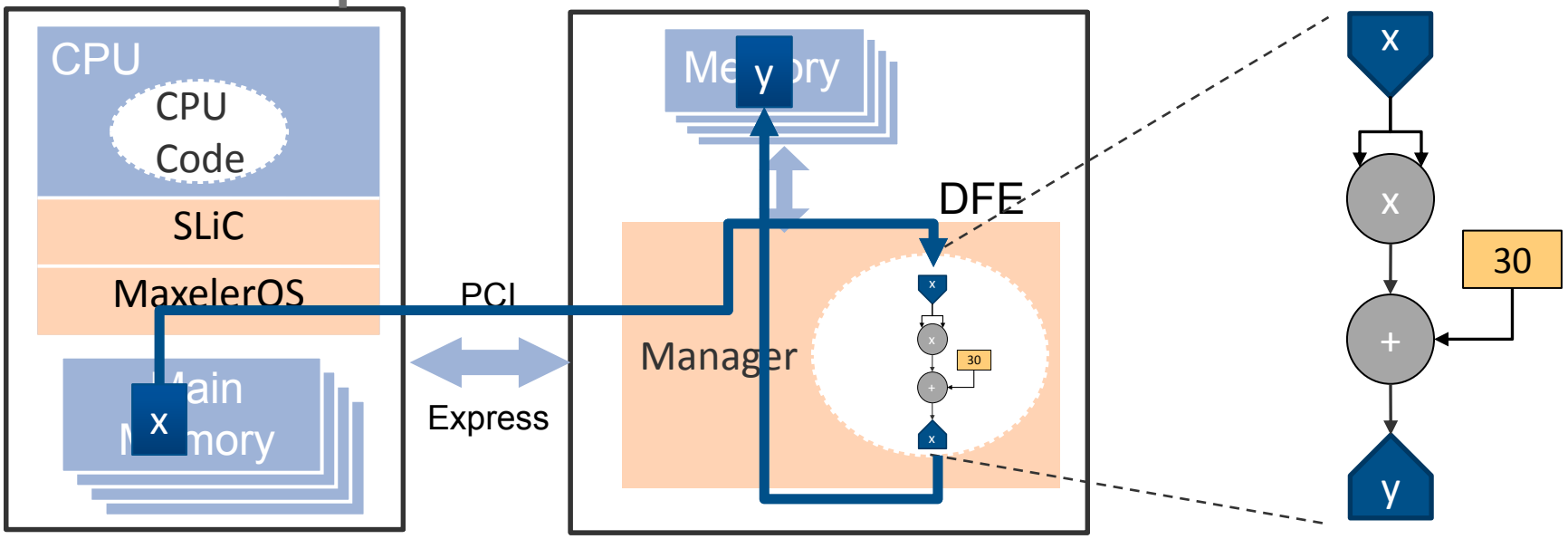
MyManager (.maxj)

```
Manager m = new Manager();
Kernel k =
    new MyKernel();

m.setKernel(k);
m.setIO(
   link("x", CPU),
   link("y", CPU));
m.build();
```

MyKernel (.maxj)

```
DFEVar x = io.input("x", dfeInt(32));

DFEVar result = x * x + 30;

io.output("y", result, dfeInt(32));
```

Development Process



Host Code (.c)

```
int*x, *y;
MyKernel( DATA_SIZE,
          x, DATA_SIZE*4);
```

MyManager (.maxj)

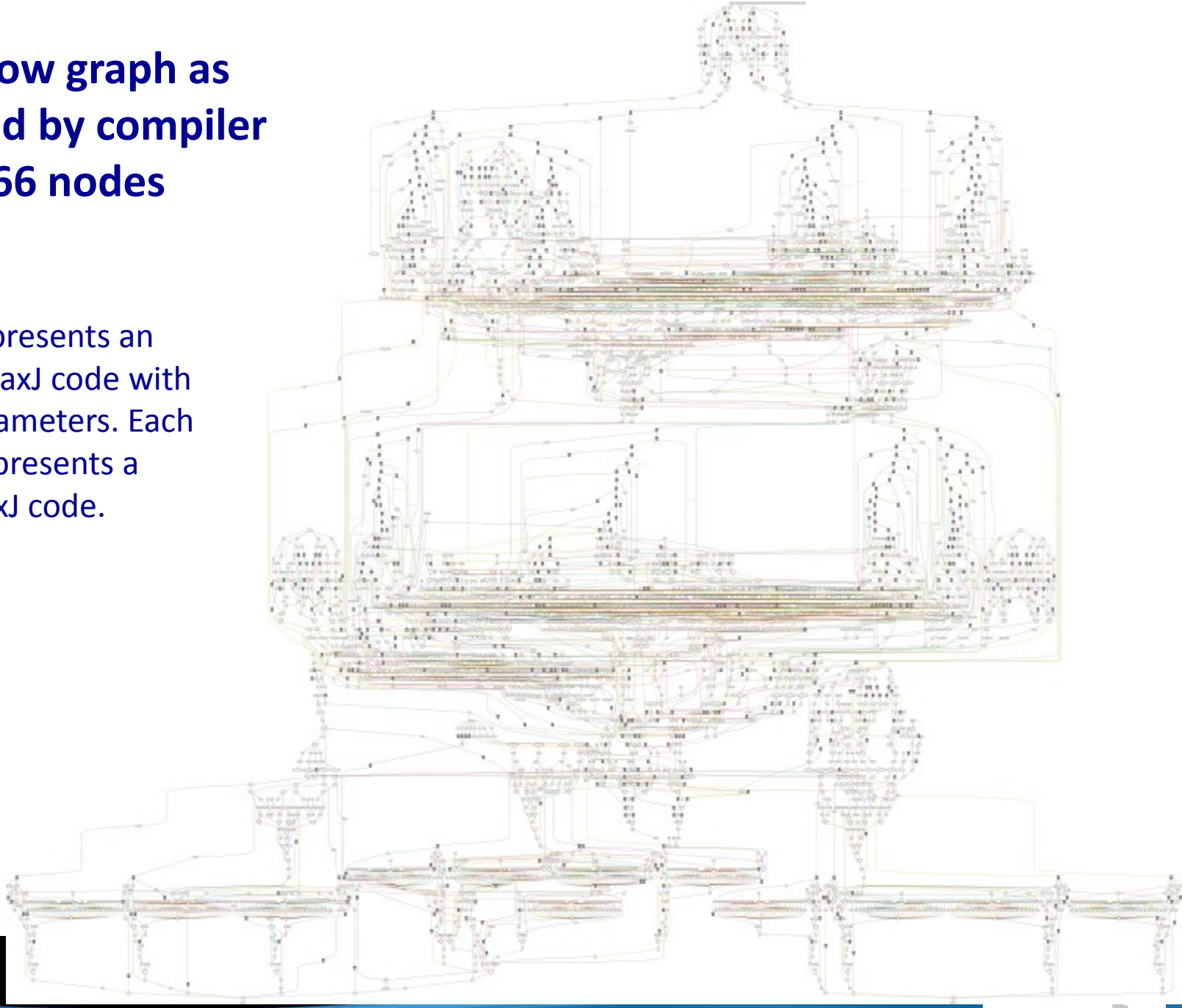
```
Manager m = new Manager();
Kernel k =
    new MyKernel();

m.setKernel(k);
m.setIO(
   link("x", CPU),
   link("y", DRAM_LINEAR1D));
m.build();
```

MyKernel (.maxj)

```
DFEVar x = io.input("x", dfeInt(32));

DFEVar result = x * x + 30;

io.output("y", result, dfeInt(32));
```

Data flow graph as generated by compiler 4866 nodes

Each node represents an operator in MaxJ code with area time parameters. Each line (edge) represents a DFEVar in MaxJ code.



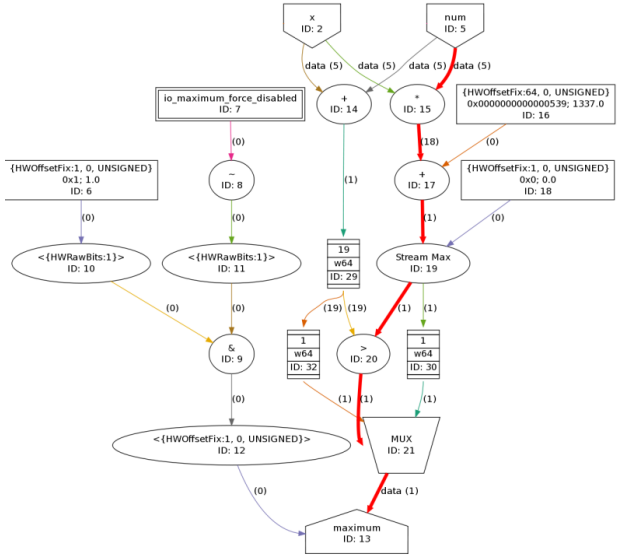
Path Latency Reporting

- MaxCompiler gives detailed latency annotation back to the programmer

```
27        :
28    12.8:      d.Buy = ask.Price <= lowPrice & order_book.securityId === secId;
29        :      d.Sell = bid.Price >= highPrice & order_book.securityId === secId;
30     6.4:      d.Quantity = d.Buy ? ask.Quantity : bid.Quantity;
31        :      d.Price = d.Buy ? ask.Price : bid.Price;
```

12.8ns + 6.4ns = 19.2ns (total compute latency)

- Evaluate precise effect of code on latency



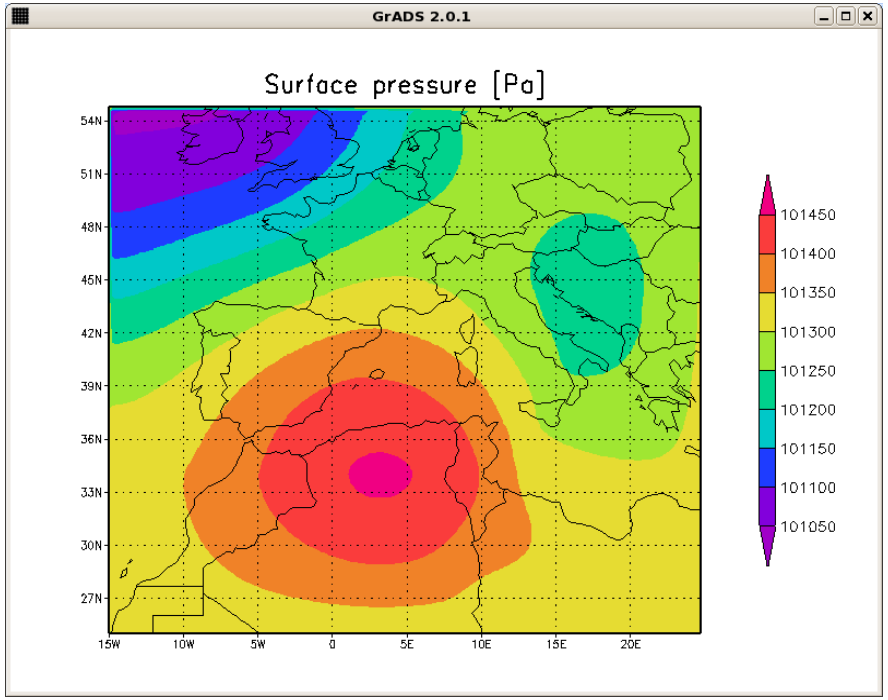
Resource Usage Reporting

- Allows you to see what lines of code are using what resources and focus optimization
  - Separate reports for each kernel and for the manager

```
  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));
                               :   }
                               : }
```

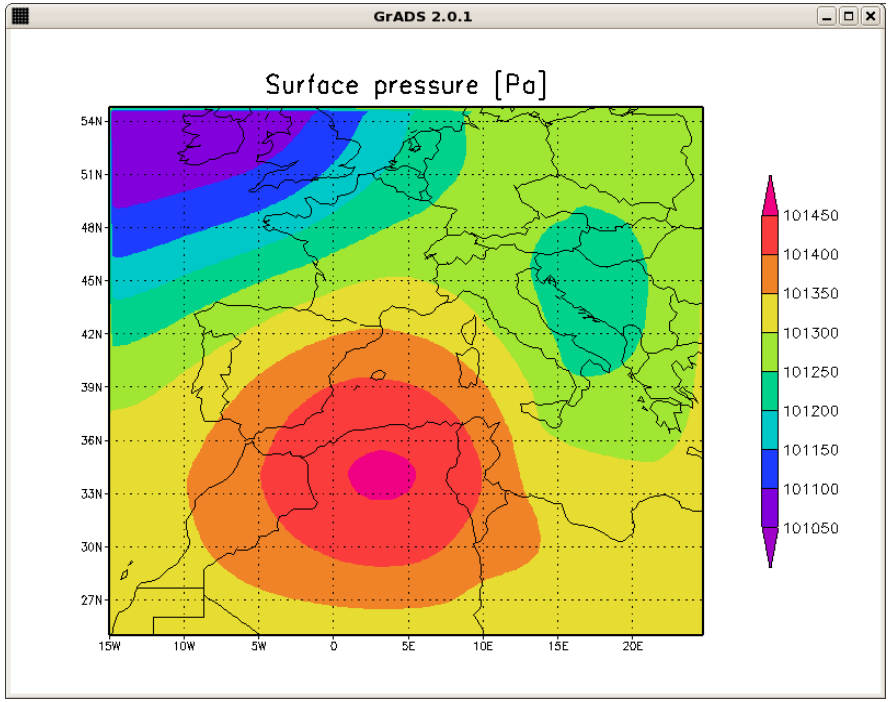
Example Projects

Imaging Platform Example: Weather



1U CPU Node

Wall Clock Time: 2 hours



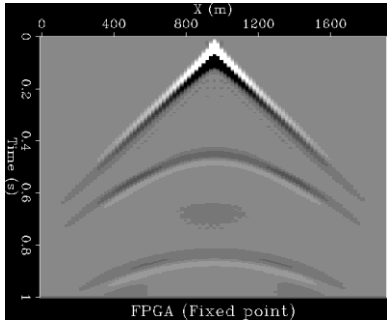
1U Dataflow Node

less than 2 minutes

Problem size: (Longitude) 13,600 Km x (Latitude) 3330 Km
Simulation of baroclinic instability after 500 time steps.

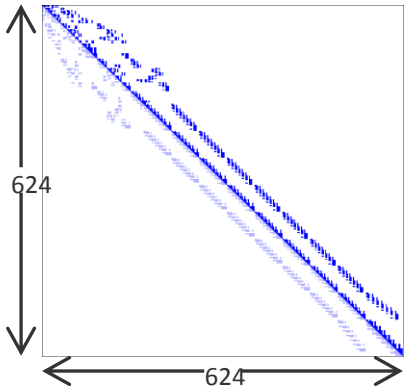
Acceleration of a Meteorological Limited Area Model with Dataflow Engines, D. Oriato, S. Tilbury (Maxeler), M. Marrocu, G. Pusceddu (CRS4), SAAHPC Conference, May 2012.

Achieved Computational Speedup for the entire application (not just the kernel) compared to Intel server



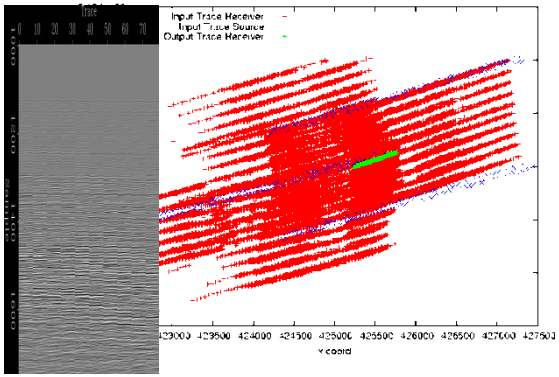
RTM with Chevron

VTI 19x and TTI 25x



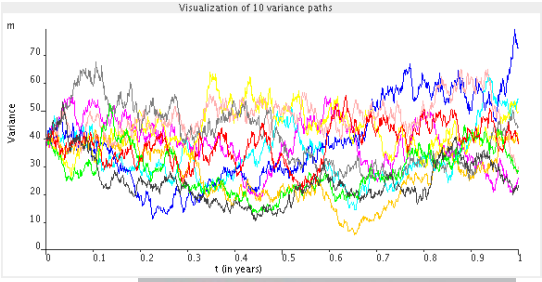
Sparse Matrix

20-40x



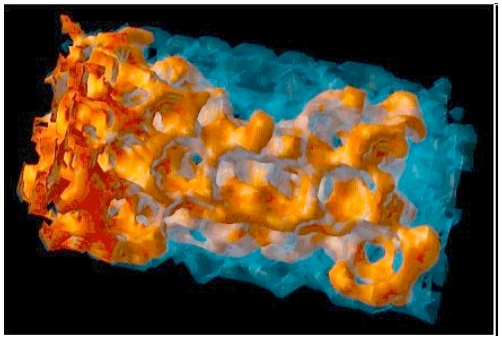
Seismic Trace Processing

24x



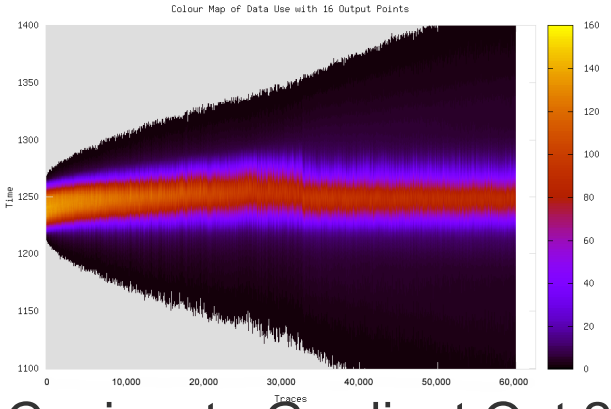
J.P.Morgan

Credit 32x and Rates 26x



Lattice Boltzman

Fluid Flow 30x



Conjugate Gradient Opt 26x

Technologies

MaxAcademy

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Maxeler UP

- University program has over 150 university members
- Membership is free
- Hardware can be bought with discount; software free
  - Low cost Galava DFE
- Possible to access via simulator and cloud
- Shared research among the members.

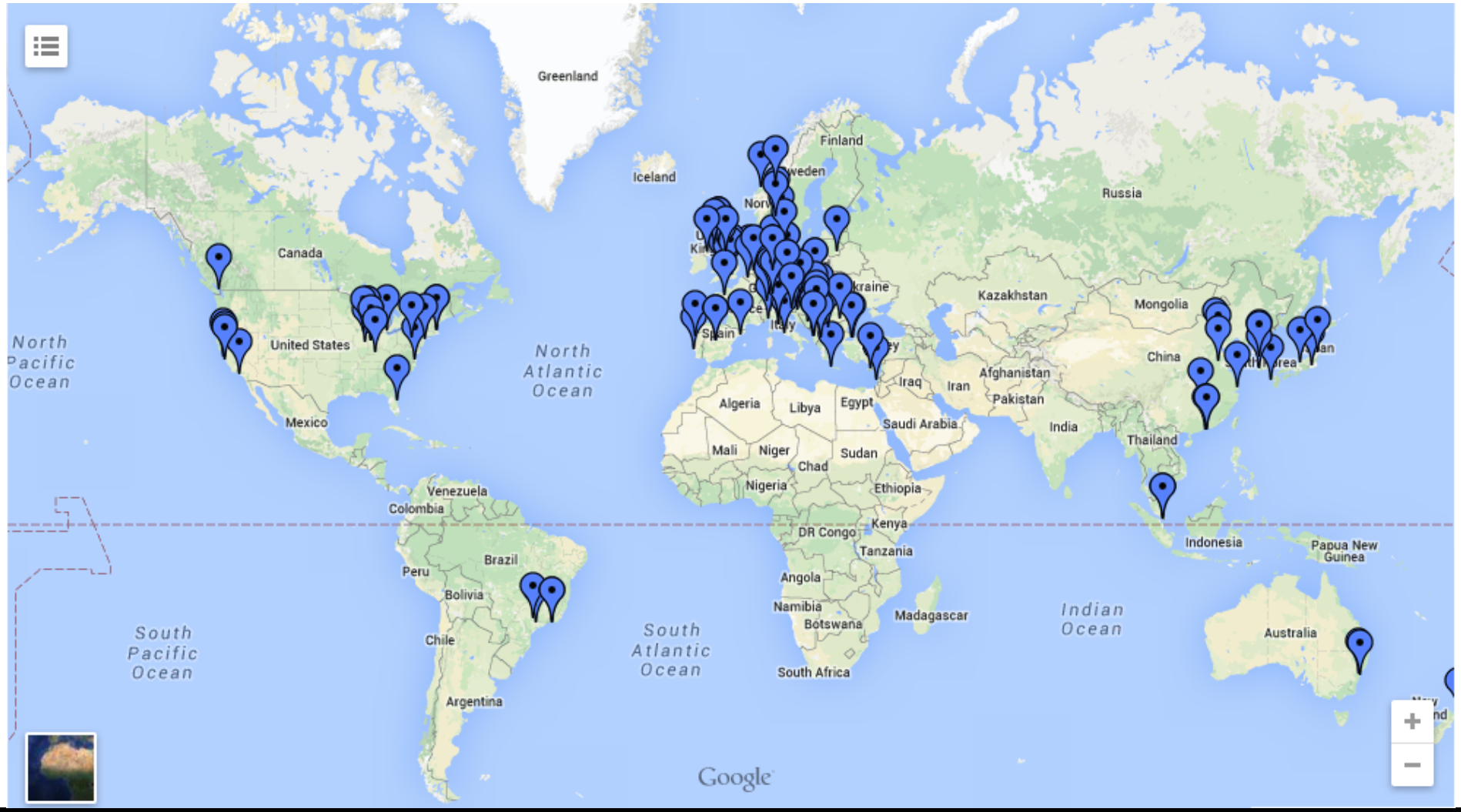
Maxeler University Program Members

Korea University
Stanford University
Universität Hamburg — Der Forschung | Der Lehre | Der Bildung
FAU Friedrich-Alexander Universität Erlangen-Nürnberg
Max-Planck-Gesellschaft
Manchester Metropolitan University
Nanyang Technological University
Universität Paderborn
香港中文大學 The Chinese University of Hong Kong
Universidade de Lisboa
KIT Karlsruhe Institute of Technology
Politecnico di Milano
The University of Tokyo
Nagasaki University 長崎大学
Heriot Watt University
Berkeley University of California
Fudan University 復旦大學
Goethe Universität Frankfurt am Main
University of Windsor thinking forward
Chalmers
University of Kragujevac
Technion Israel Institute of Technology
NUS National University of Singapore
University of Oxford
LMU Ludwig-Maximilians-Universität München
The Hong Kong Polytechnic University 香港理工大學
Technische Universität Kaiserslautern
東北大学 Tohoku University
UIC University of Illinois at Chicago
JAIST Japan Advanced Institute of Science and Technology
UBC
Technical University of Crete
清華大學 Tsinghua University
The University of Manchester Manchester Business School
UiO University of Oslo
Ss. Cyril & Methodius University of Skopje Macedonia
Imperial College London
The University of Hong Kong
UCL
UBC
TU Delft Delft University of Technology
AGH University of Science and Technology
Chosun University 조선대학교
Northeastern University
UCLA

Technologies

Maxeler University Program Members

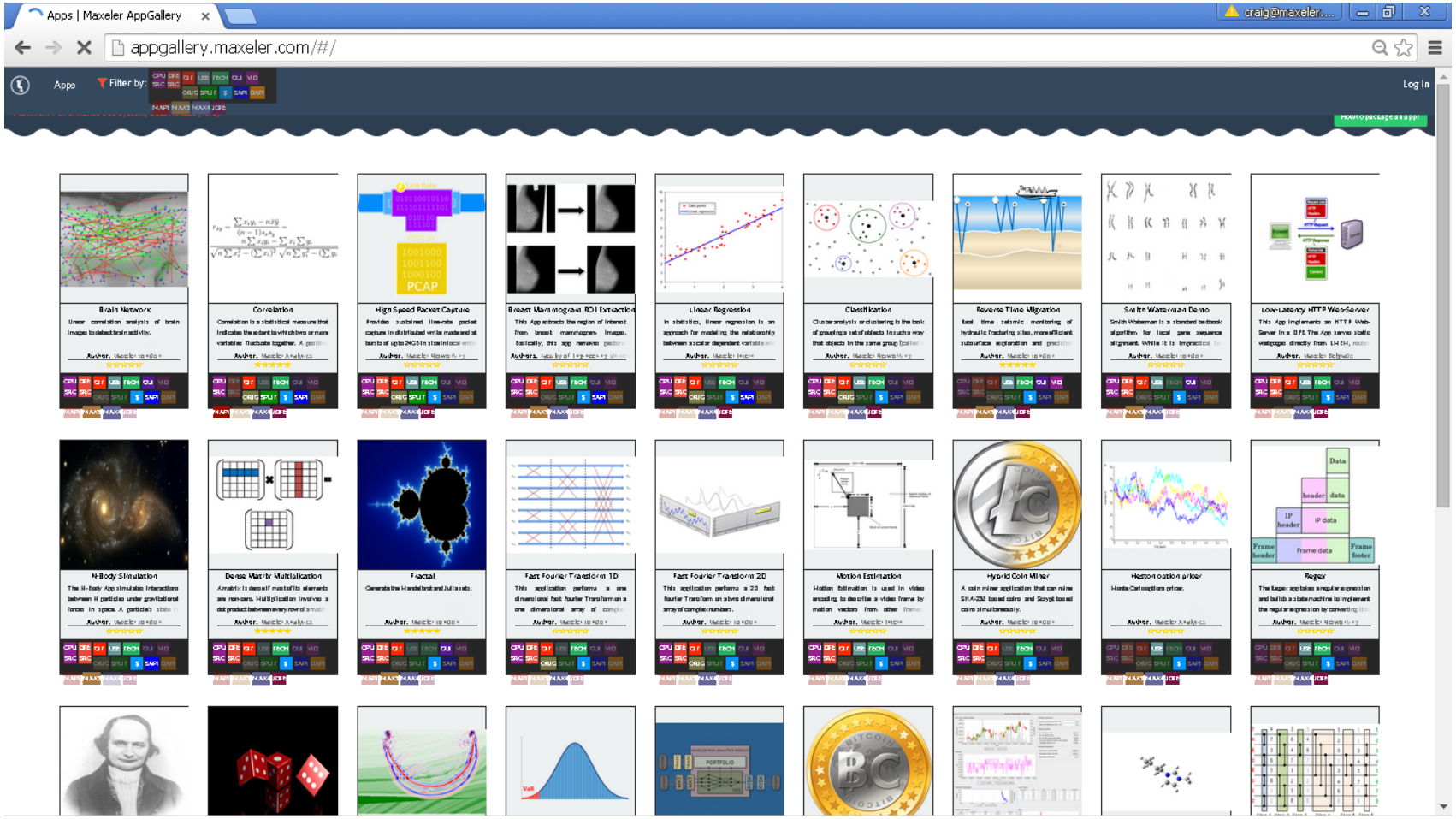
150 Universities on 5 continents.



Technologies

appgallery.maxeler.com

Number of example applications with, in many cases, access to source and docs.



github.com/maxeler/maxpower

Open source project of kernel utilities and functional blocks.

maxpower/src/maxpower at

GitHub, Inc. [US] https://github.com/maxeler/maxpower/tree/master/src/maxpower

GitHub | This repository | Search | Explore | Features | Enterprise | Blog | Sign up | Sign in

maxeler / maxpower | Watch 8 | Star 4 | Fork 1

branch: master | maxpower / src / maxpower / +

Addressing Chris's comments about comments.

Stephen Girdlestone authored 7 days ago | latest commit 600d4a0a83

..		
blas/l3	Increase LSO threshold in TillAcc.	a month ago
hash	Use a config file for environment setup	3 months ago
kernel	Addressing Chris's comments about comments.	7 days ago
lmem	Framer and SuperFIFO in MaxPower	28 days ago
manager	Added more javadoc.	4 months ago
network/tcp/manyconn/framer	Framer and SuperFIFO in MaxPower	28 days ago
ops	Added more javadoc.	4 months ago
statemachine/collections	Remove outdated Javadoc.	5 months ago

Technologies

Maxeler Developer Exchange (MDX)

Google group for Q&A amongst developers and Maxeler staff.

Developer Exchange | Maxel

https://www.maxeler.com/mymaxeler/mdx/

MAXELER Technologies

Platforms Products Technology About Us

Search

MyMaxeler

Developer Exchange

Developer Exchange

The Maxeler Developer Exchange (MDX) is an on-line forum for the exchange of technical information, questions and answers between developers working with Maxeler acceleration solutions. Membership is open to MaxCompiler developers at Maxeler partners, clients and MAX-UP university program member universities.

You will need a Google account to access MDX.

If you are already logged into your Google account you can access MDX directly **here**.

If you are not logged in, access MDX **here**.

MAXELER DEVELOPER EXCHANGE

© Maxeler technologies | **Contact us** |

Follow us

Technologies

Questions ?

Break ?