HPC Acceleration with Dataflow Programming
Overview

• The challenges
• Heterogeneous dataflow computers
• Programming dataflow computers
• Application case studies
The Performance Challenge: 1 Exaflop

• 1 exaflop = $10^{18}$ FLOPS
• Using processor cores with 8FLOPS/clock at 2.5GHz
• **50M CPU cores**
• What about power?
  – Assume power envelope of 100W per chip
  – Moore’s Law scaling: 8 cores today $\rightarrow$ ~100 cores/chip
  – 500k CPU chips
• **50MW (just for CPUs!) $\rightarrow$ 100MW+ likely**
• ‘Titan’: 17PF @ 8MW $\rightarrow$ 470MW for Exascale
Extreme Parallelism

- Spatial decomposition on a $10,000^3$ regular grid
- 1.0 Terapoints
- 20k points per core
- $27^3$ region per core
- Computing a 13x13x13 convolution stencil: 66% halo

Limited by communication bandwidth
The data movement challenge

- Moving data on-chip will use as much energy as computing with it
- Moving data off-chip will use **200x more energy**!
  - And is much slower as well

<table>
<thead>
<tr>
<th></th>
<th>Today</th>
<th>2018-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double precision FLOP</td>
<td>100pj</td>
<td>10pj</td>
</tr>
<tr>
<td>Moving data on-chip: 1mm</td>
<td>6pj</td>
<td></td>
</tr>
<tr>
<td>Moving data on-chip: 20mm</td>
<td>120pj</td>
<td></td>
</tr>
<tr>
<td>Moving data to off-chip memory</td>
<td>5000pj</td>
<td>2000pj</td>
</tr>
</tbody>
</table>
Solving Real Problems

• Peak performance doesn’t matter
• “Real” performance on LINPACK doesn’t matter
• The TOP500 doesn’t matter
• We’re interested in solving real problems
  – Science
  – Commerce
  – Industry

• Can we build application-specific computers?
• How should we program them?
Control flow vs. Dataflow
Where silicon is used?

Intel 6-Core X5680 “Westmere”
Where silicon is used?

Intel 6-Core X5680 “Westmere”

Dataflow Processor

Computation

MaxelerOS

Computation
(Dataflow cores)
A Special Purpose Computer?

- A custom chip for a specific application
- No instructions $\Rightarrow$ no instruction decode logic
- No branches $\Rightarrow$ no branch prediction
- Explicit parallelism $\Rightarrow$ No out-of-order scheduling
- Data streamed onto-chip $\Rightarrow$ No multi-level caches
A Special Purpose Computer?

- But we have more than one application
- Generally impractical to have machines that are completely optimized for only one code
A Special Purpose Computer?

• Use a *reconfigurable* chip that can be reprogrammed at runtime to implement:
  – Different applications
  – Or different versions of the same application

• Allows us to dynamically (re)allocate parts of a machine to different application tasks
Heterogeneous Dataflow System
Maxeler Hardware

**MPC C Series**
- CPUs plus DFEs
- Intel Xeon CPU cores and up to 4 DFEs with 192GB 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 1-4 DFEs with up to twelve 40Gbit Ethernet connections

**MaxWorkstation**
- Desktop development system

**MaxCloud**
- On-demand scalable accelerated compute resource, hosted in London
MPC-C500

- 1U Form Factor
- 4x dataflow engines
- 12 Intel Xeon cores
- 96GB DFE RAM
- Up to 192GB CPU RAM
- MaxRing interconnect
- 3x 3.5” hard drives
- Infiniband/10GigE
MPC-X1000

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

• Optimized to balance resources for particular application challenges

• Individual MPC-X nodes typically 20-50x faster than 2 SOCKET x86 nodes

• 10-30x density and power advantage at the rack level
Programming Dataflow Systems
Acceleration Flow

Start

Original Application

Identify code for acceleration and analyze bottlenecks → Transform app, architect and model performance → Write MaxCompiler code → Integrate with CPU code → Simulate DFE

Accelerated Application

YES

NO

Meets performance goals?

Build full DFE configuration

Functions correctly?

NO

YES

NO

NO

YES

Meets performance goals?

Build full DFE configuration

Functions correctly?
Consider Data Movement first

Try to minimise runtime and development time, while maximising flexibility and precision.
Data Flow Analysis: Matrix Multiply

Loop iteration space: 150
MM-simple-inlined.c:133-133
main()133-133

Loop iteration space: 150
MM-simple-inlined.c:140-140
main()140-140

Loop iteration space: 150
MM-simple-inlined.c:115-115
main()115-115

Loop iteration space: 150
MM-simple-inlined.c:119-119
main()119-119

Loop iteration space: 160
MM-simple-inlined.c:135-135
main()135-135

Loop iteration space: 150
MM-simple-inlined.c:145-145
main()145-145

data=25 MB

data=25 MB

data=51 MB

data=102 MB
Co-design approach

- Maxeler team includes mathematicians, scientists, electronic engineers and computer scientists
- Deploy domain expertise to **co-design** application, algorithm and computer architecture
Programming with MaxCompiler

User Input

SLiC

Compiler, Linker

MaxelerOS SW, HW

HW Accelerator (.max)

Kernel Compiler

Manager

MaxIDE

Application Kernel(s) (.java)

Manager Configuration (.java)

Output

Accelerated Application (executable)
Programming with MaxCompiler

```java
import com.maxeler.maxcompiler.v2.kernels.Kernel;

class ConvolveKernel extends Kernel {
    // DFE
    float x = io.input("x", type);
    float y = io.input("y", type);
    float a = io.scalarInput("a", type);

    protected ConvolveKernel(KernelParameters parameters) {
        super(parameters);
    }

    @Override
    protected void compute() {
        int N = size;
        float sum = 0;
        for (int i = 0; i < N; i++) {
            sum += x[i] * y[i] * a;
        }
        io.output("z", sum);
    }
}
```
for (int i =0; i < DATA_SIZE; i++)
    y[i] = x[i] \times x[i] + 30;

y_i = x_i \times x_i + 30
#include "MaxSLiCInterface.h"
#include "Calc.max"

Calc(x, y, DATA_SIZE)

Manager (java)
Manager m = new Manager("Calc");
Kernel k =
    new MyKernel();
m.setKernel(k);
m.setIO(
    link("x", CPU),
    link("y", CPU));
m.createSLiCInterface();
m.build();

MyKernel (java)
DFEVar x = io.input("x", dfeInt(32));
DFEVar result = x * x + 30;
io.output("y", result, dfeInt(32));
Programming with MaxCompiler

**Manager (.java)**

Manager m = new Manager("Calc");
Kernel k =
    new MyKernel();
m.setKernel(k);
m.setIO(
    link("x", CPU),
    link("y", LMEM_LINEAR1D));
m.createSLiCInterface();
m.build();

**MyKernel (.java)**

DFEVar x = io.input("x", dfelnt(32));
DFEVar result = x * x + 30;
io.output("y", result, dfelnt(32));

**CPUCode (.c)**

```
#include "MaxSLiCInterface.h"
#include "Calc.max"
int *x, *y;

Calc(x, DATA_SIZE)
```
public class MyKernel extends Kernel {

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

        HWVar x = io.input("x", hwInt(32));

        HWVar result = x * x + 30;

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

The Full Kernel
Kernel Streaming: Programmer’s View
Kernel Streaming: Programmer’s View
Kernel Streaming: Programmer’s View
Kernel Streaming: Programmer’s View

5 4 3 2 1 0

x

3

x

9

+

39

y

30 31 34 39
Kernel Streaming: Programmer’s View
Kernel Streaming: Programmer’s View
Kernel Streaming: In Hardware
Kernel Streaming: In Hardware

5 4 3 2 1 0

x

0

x

+ 30

y

[Diagram of a flowchart with labeled nodes and arrows indicating the flow of data or operations.]
Kernel Streaming: In Hardware
Kernel Streaming: In Hardware
Kernel Streaming: In Hardware

5 4 3 2 1 0

x

3

x

4

31

+

30

y

30
Kernel Streaming: In Hardware
Kernel Streaming: In Hardware
Kernel Streaming: In Hardware

5 4 3 2 1 0

30 31 34 39

x

25

+ 30

y

46
Kernel Streaming: In Hardware

```
5 4 3 2 1 0
```

```
55
```

```
30 31 34 39 46
```
Kernel Streaming: In Hardware
Real data flow graph as generated by MaxCompiler
4866 nodes;
10,000s of stages/cycles
Seismic 3D Finite Difference Modeling

• Geophysical Model
  – 3D acoustic wave equation
  \[
  \frac{\partial^2 p}{\partial t^2} = K \nabla \cdot \left( \frac{1}{\rho} \nabla p \right) + S(t)
  \]
  – Variable velocity and density
  – Isotropic medium

• Numerical Model
  – Finite differences (12\textsuperscript{th} order convolution)
  – 4\textsuperscript{th} order in time
  – Point source, absorbing boundary conditions
Modeling Results

- Up to 240x speedup for 1 MAX2 card compared to single CPU core
- Speedup increases with cube size
- 1 billion point modeling domain using single card
Sparse matrices are used in a variety of important applications

Matrix solving. Given matrix $A$, vector $b$, find vector $x$ in:

$$Ax = b$$

- Direct or iterative solver
- Structured vs. unstructured matrices
Sparse Matrix in DFE

Compression Ratio vs. Speedup per 1U Node

Maxeler Domain Specific Address and Data Encoding

SPEEDUP is 20x-40x per 1U at 200MHz
Credit Derivatives Valuation & Risk

- Compute value of complex financial derivatives (CDOs)
- Typically run overnight, but beneficial to compute in real-time
- Many independent jobs
- Speedup: 220-270x
- Power consumption per node drops from 250W to 235W/node
Bitcoin mining

- Cryptographic application
- Repeated SHA-256 hashing to search for hash value that meets certain criteria
- “Miner” who finds hash is rewarded with bitcoins
- Minimizing energy costs per computation is key to scalability & profitability
- MPC-X2000 provides 12GHash/s @ 15MHash/W
  - ~10x power efficiency compared to GPU

Structure of one of 64 stages of SHA-256 function (Image: wikipedia)
Meteorological Modelling

- Local Area Model
- Computationally intensive process
- Accurate local weather forecasts require dense grids
- Dataflow computing can provide order of magnitude performance boosts

- Example: **1 dataflow node** equivalent to **381 cores**
  - Joint work between Maxeler and CRS4 Lab, Italy
Maxeler University Program

[University logos and names]

- Korea University
- Stanford University
- Universität Hamburg
- FAU Friedrich-Alexander Universität Erlangen-Nürnberg
- MAX-PLANCK-GESELLSCHAFT
- University of Lisboa
- Universität Paderborn
- Nanyang Technological University
- Politecnico di Milano
- The University of Tokyo
- Nagasaki University
- Heriot Watt University
- Berkeley University
- Goethe Universität Frankfurt am Main
- University of Windsor
- Chalmers
- University of Kragujevac
- Technion Israel Institute of Technology
- National University of Singapore
- University of Oxford
- LMU Ludwig-Maximilians-Universität München
- Hong Kong Polytechnic University
- The University of Manchester
- Tohoku University
- University of Illinois at Chicago
- JAIST
- Cyril & Methodius University of Skopje, Macedonia
- UiO University of Oslo
- Imperial College London
- The University of Hong Kong
- UCL
- The University of Manchester
- Manchester Metropolitan University
- UBC
- TUDelft Delft University of Technology
- MAXELER Technologies

AGH University of Science and Technology
- Chosun University
- Northeastern University
Summary & Conclusions

• Data movement is the fundamental challenge for current and future computer systems

• Dataflow computing achieves high performance through:
  – Explicitly putting data movement at the heart of the program
  – Employing massive parallelism at low clock frequencies

• Many scientific applications can benefit from this approach