Accelerating computations in very large applications using data flow based accelerators

Michael J. Flynn
Maxeler Technologies and Stanford University

The (multi core) Parallel Processor Problem

- Efficient distribution of tasks
- Inter-node communications (data assembly & dispatch) reduces computational efficiency: speedup/nodes
- Memory limitations
- Layers of abstraction hide critical sources of and limits to efficient parallel execution
- Result: scaled up cost, power, cooling and reliability concerns

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Hardware and Software Alternatives

- Hardware:
  A more generalized (and reconfigurable) heterogeneous accelerator array model

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

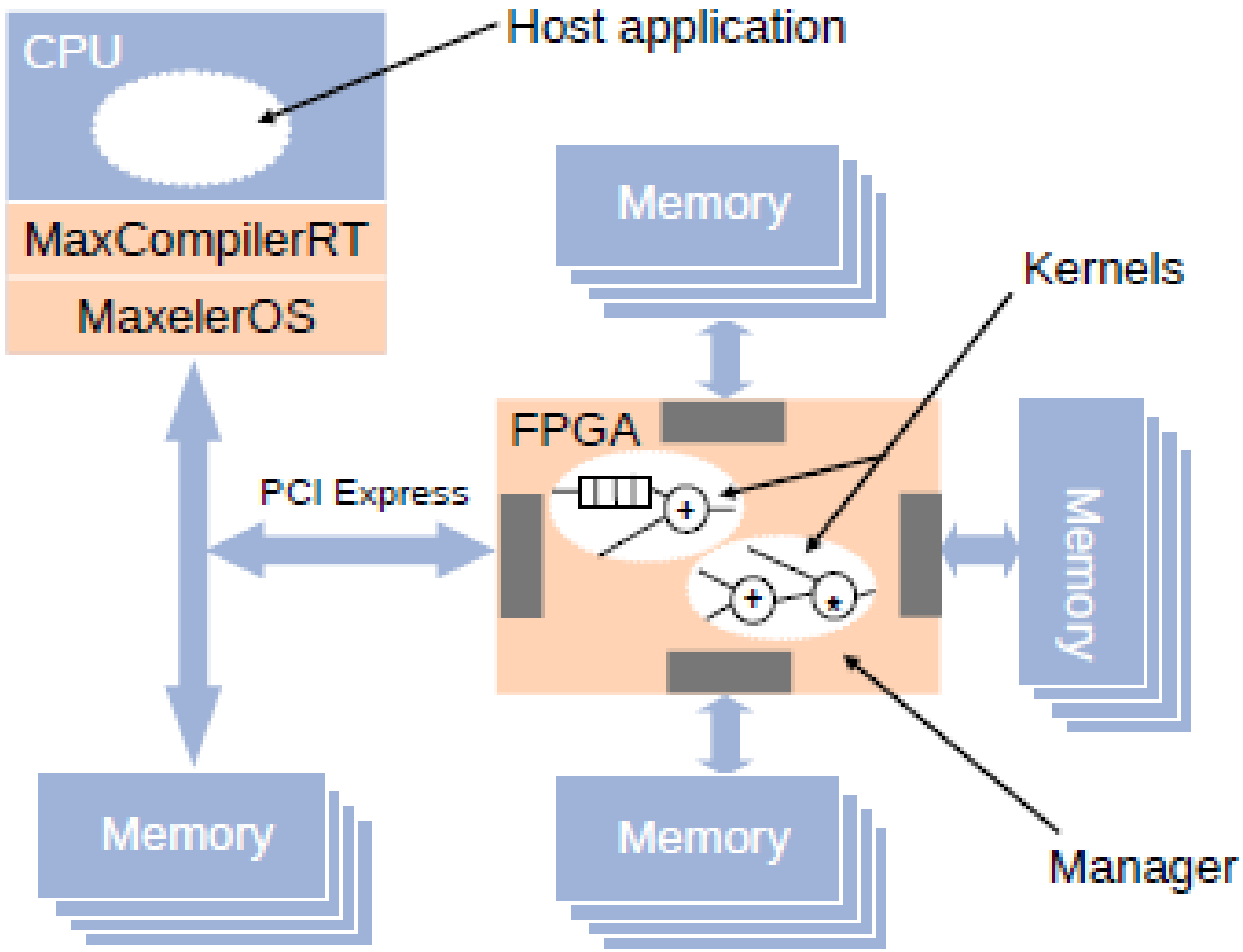
MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Heterogeneous Accelerator Hardware Model

- Assumes host CPU + accelerator
- Application consists of two parts
  - Essential (high usage, >99%) part
  - Exceptional part (<1% dynamic activity)
- Essential part is executed on accelerator; exceptional part on host

MAXELER
Technologies
MAXIMUM PERFORMANCE COMPUTING

FPGA accelerator hardware model: server with acceleration cards



MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Programs, DFGs and Hardware

- Each (kernel) program has a data flow graph (DFG)
- The ideal HW to execute the DFG is a data flow machine that exactly matches the DFG
- A compiler / translator attempts to transform the DFG so that it resembles the HW

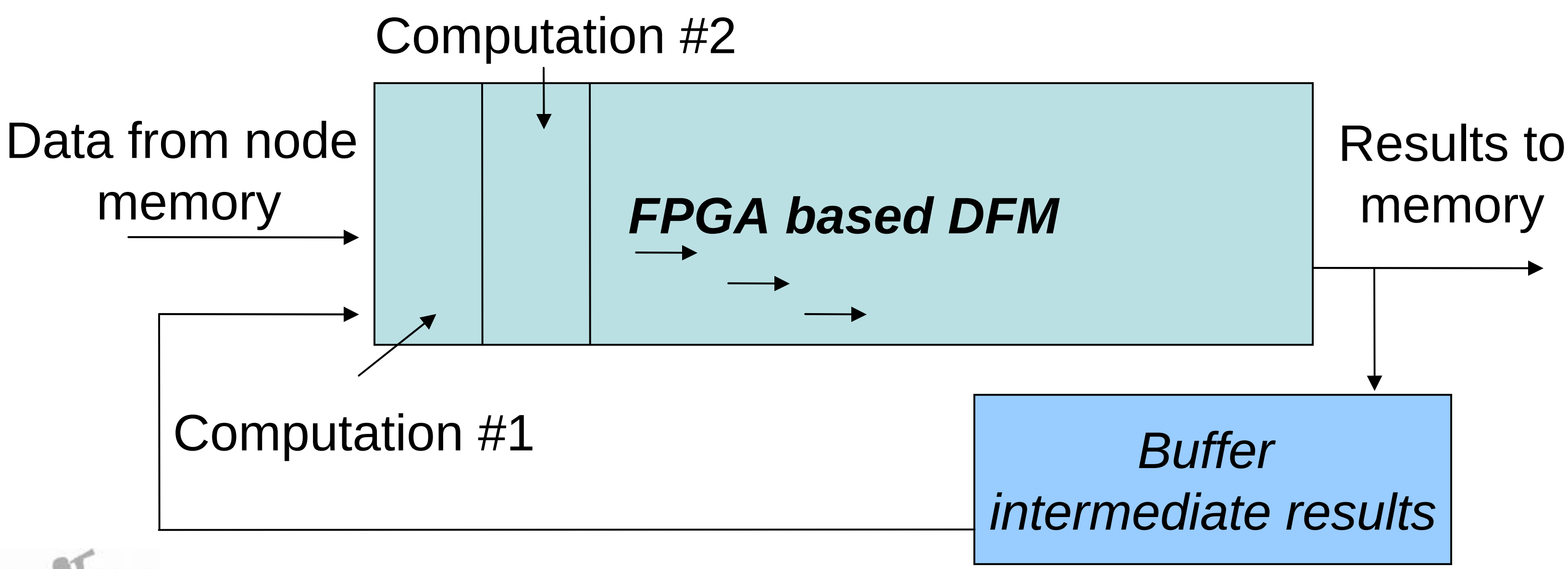
Transforming DFGs to Match the HW

- FPGA based accelerators, while slow in cycle time, offer much more flexibility in matching DFGs
- Goal is to create a static DFM and stream data across (MISD style)
- Limitation 1: The DFG is limited in (static) size to O (10^4) nodes
- Limitation 2: Only the control structure is matched not the data access patterns; so memory choreography must be managed additionally

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Accelerate Tasks by FPGA-based DFMs

- Create a fully synchronous data flow machine synchronized to multiple memory channels, then stream computations across a long array

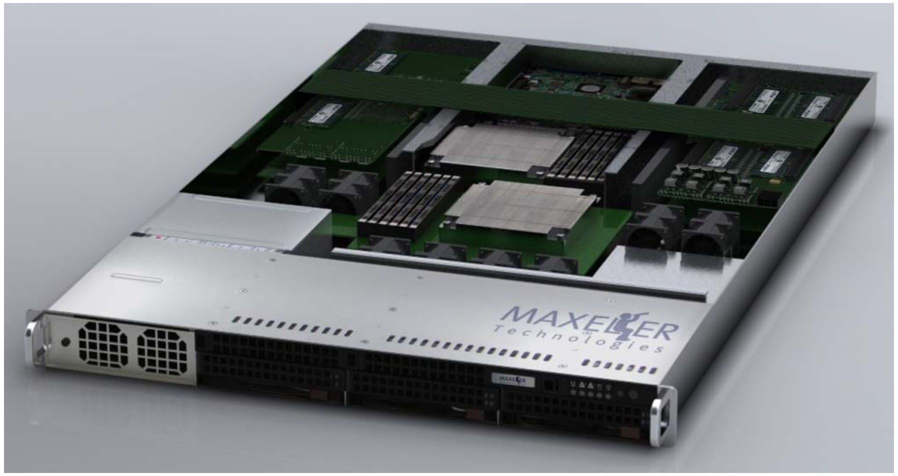


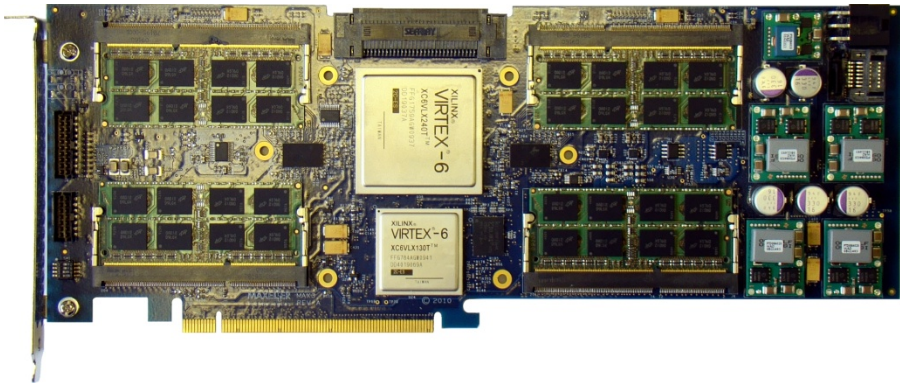
FPGA Acceleration

- One tenth the frequency with 10^6 cells per die
- Magnitude of parallelism overcomes frequency limitations
- Stream data across large cell array, minimizing memory bandwidth
- Customized data structures; e.g., 17 bit floating point -- always just enough precision
- A software (re)configurable technology

MaxNode- with MAX3

- 1U Form Factor
- 4x MAX3 cards
  with Virtex-6 FPGAs
- 2x Intel Xeon CPUs
- Up to 96GB host RAM
- Up to 96GB FPGA RAM
- 3x 3.5” disks
- ~700W Power

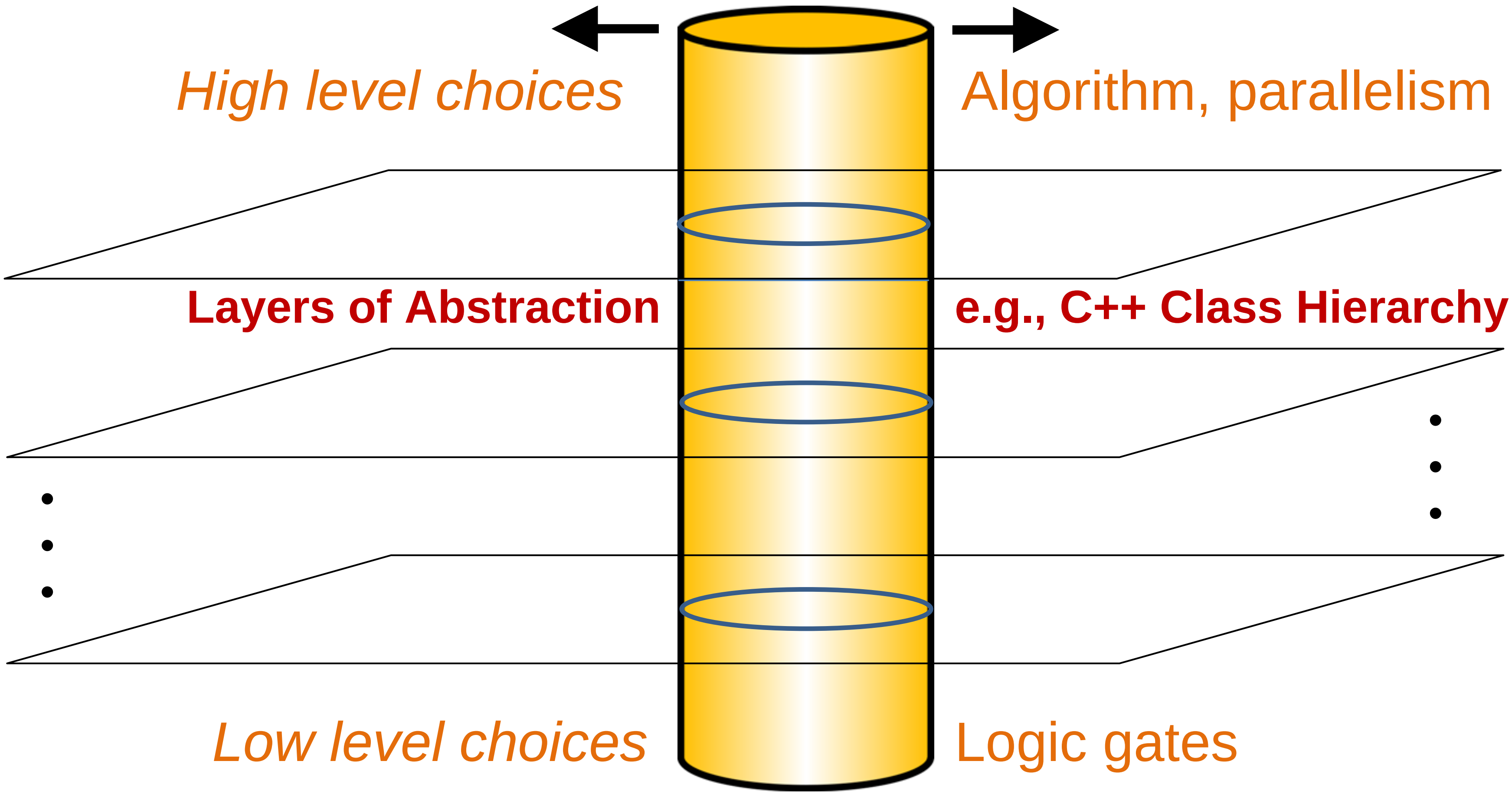




SW: A Different Programming Model

- A cylindrical rather than a layered model suits static applications
- Create a synchronous data flow machine (DFM) based on the application data flow graph
- Use a streaming computational model in data centric applications

Cylindrical Model for Vertical Acceleration



- First cut / accelerate a small vertical kernel / cylinder
- Later extend kernel size to achieve full application speedup

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Acceleration via Streaming and On-Demand Dataflow Machines

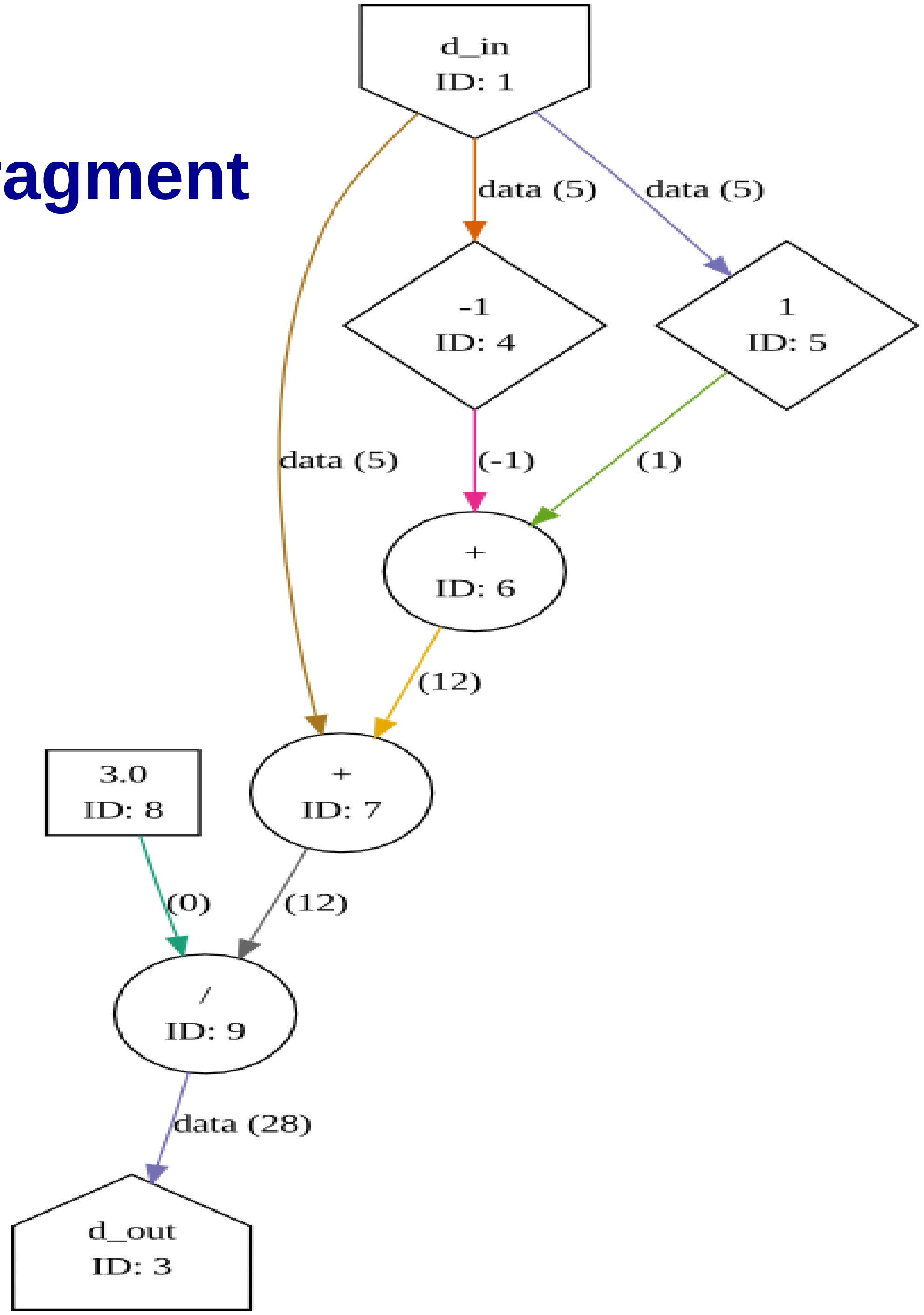
- Create “static” form of source code
- Create dataflow graph of static source code
- Compile dataflow graph into synchronized dataflow machine; suitable for data streaming
- Iterate on DF machine to optimize use of I/O pins and silicon (usage of elements)
- Simulate; then place and route

Speedup with the Cylindrical Model

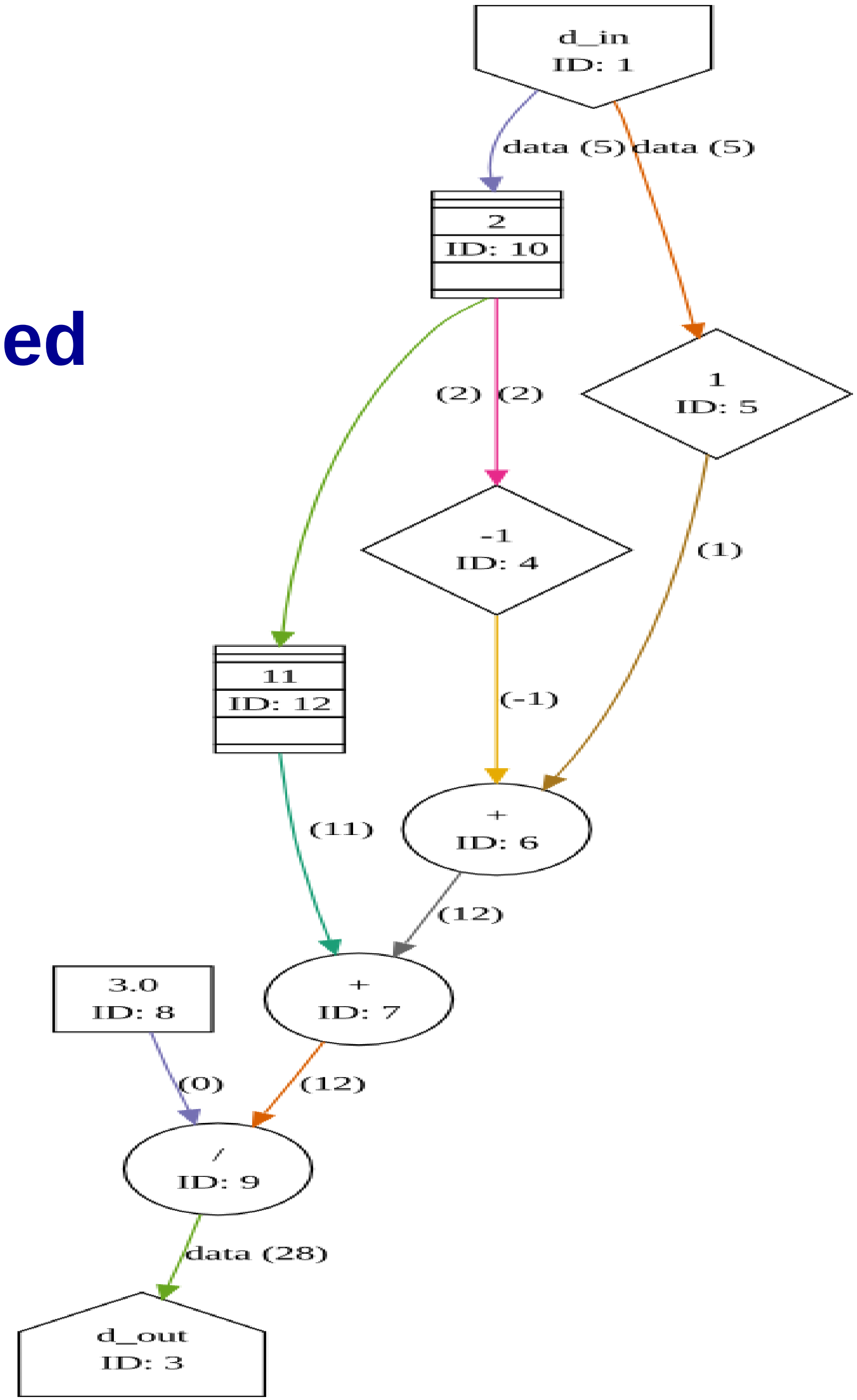
- Transform application to execute multiple simultaneous DFMs using DRAM "pipes"
- Stream computations through each pipe using memory choreography
- DFM size limited by FPGA area and DRAM (and FPGA pin) bandwidth
  - Application specific data precision
    - multiplies FPGA area
    - multiplies DRAM bandwidth

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Data flow graph fragment

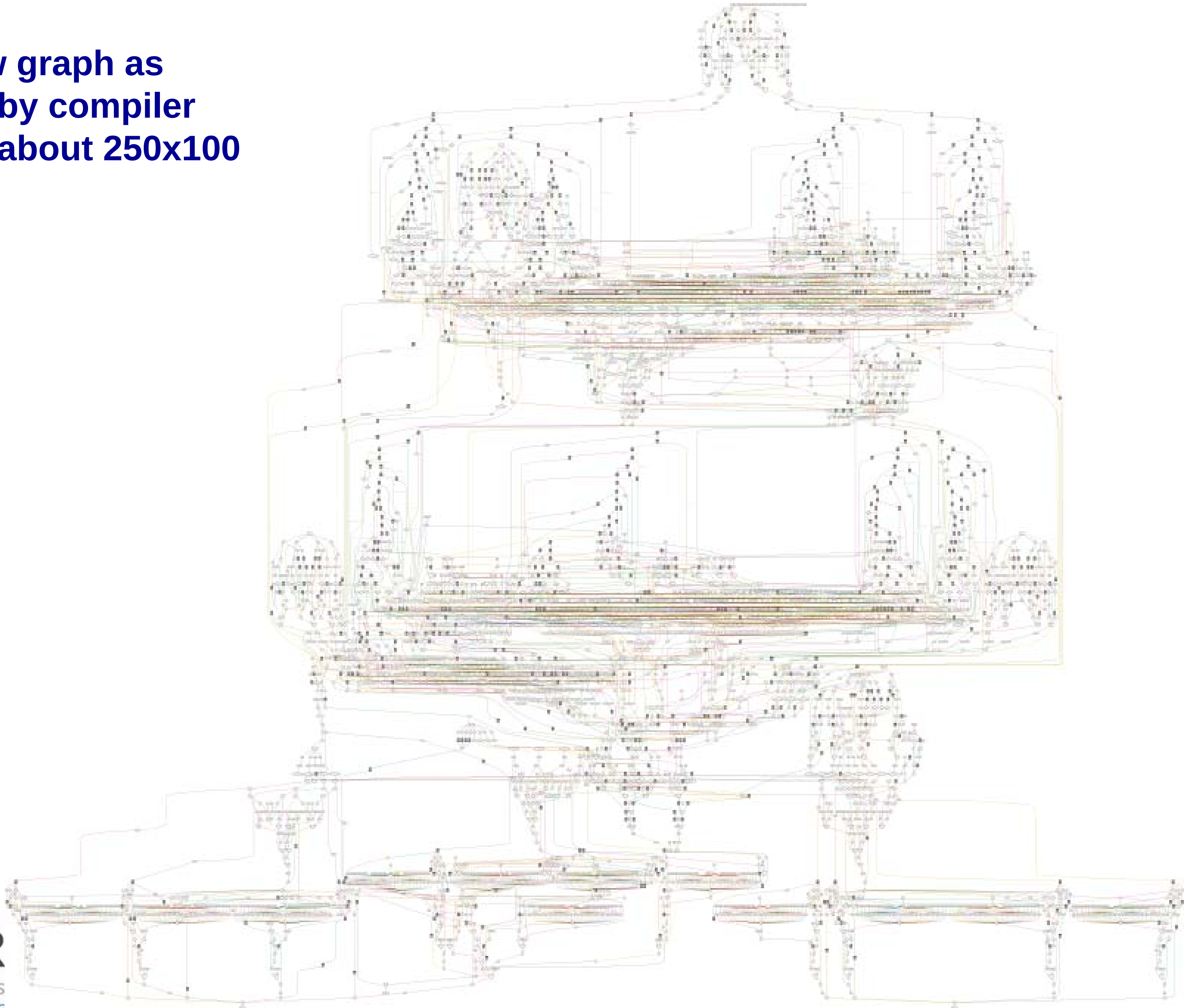


Same fragment after compilation now buffer synchronized



MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

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



MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

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

Automating the Process: 1

- Tools we now have:
  - Profiler identifies “essential” kernel
  - Compiler creates DFG
  - Compiler creates DFM from DFG
  - OS, drivers and source data “streams” enable memory choreography

Automating the Process: 2

- Tools and methodology under development
  - Tools to assist the rewrite source code into “static” form with data streams
  - Compiler optimized for pipeline BW, not for minimum critical path length
  - Identifying algorithmic tradeoffs
  - Managing the DFG: reshaping to use computational volume, optimizing for pin BW
  - Optimizing data structures

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Some application areas with published results

- Finite Difference Modelling
- Reverse Time Migration
- Common Refection Surface stacking
- Sparse Matrix Solving
- Credit Derivatives Pricing (Monte Carlo simulation)

Example: Seismic Data Processing

- For Oil & Gas 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 seconds

⇩

Order of terabytes of data each day

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

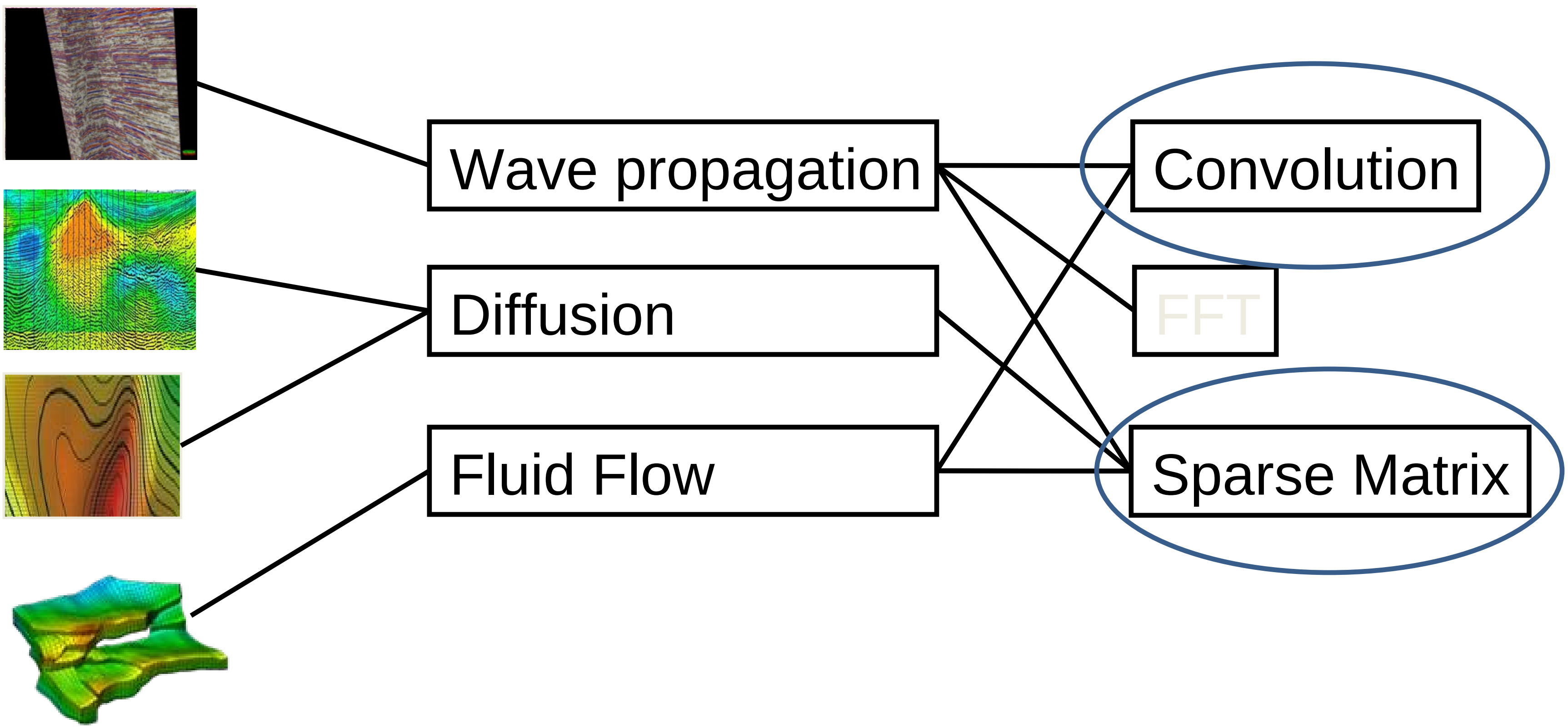
Seismic Data Processing: A Lot of Data to be Processed

- 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

That's Only Part of the Story; Much More Computational Capacity is Required

- Better physics
- More robust mathematical models
- More data and higher resolution

Oil and Gas Computational Kernels



MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

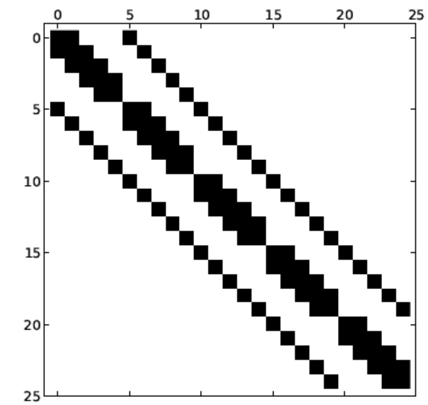
Sparse Matrix Solving

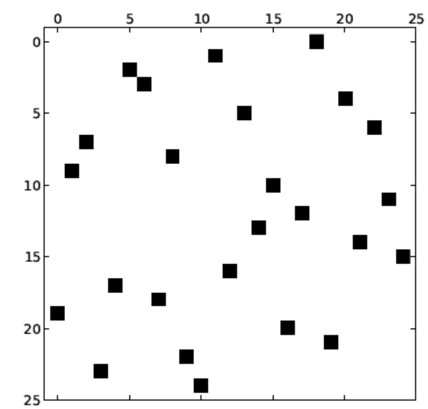
O. Lindtjorn et al, 2010

- Sparse matrices are used in a variety of important applications
- Matrix solving. Given matrix **A**, vector *b*, find vector *x* in:

$$\mathbf{A}x = b$$

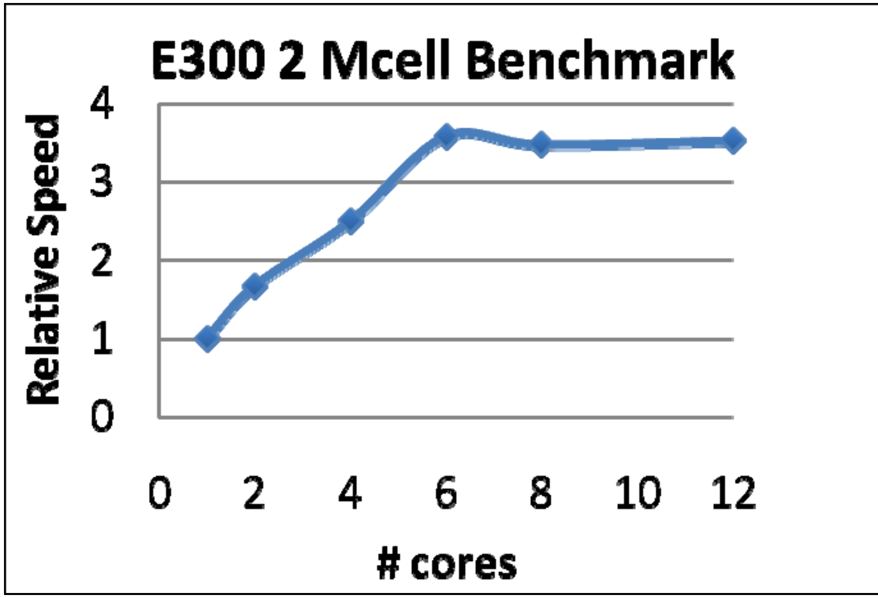
- Direct or iterative solver
- Structured vs. unstructured matrices

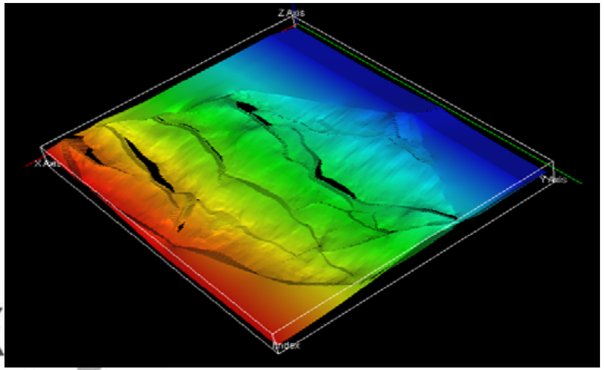




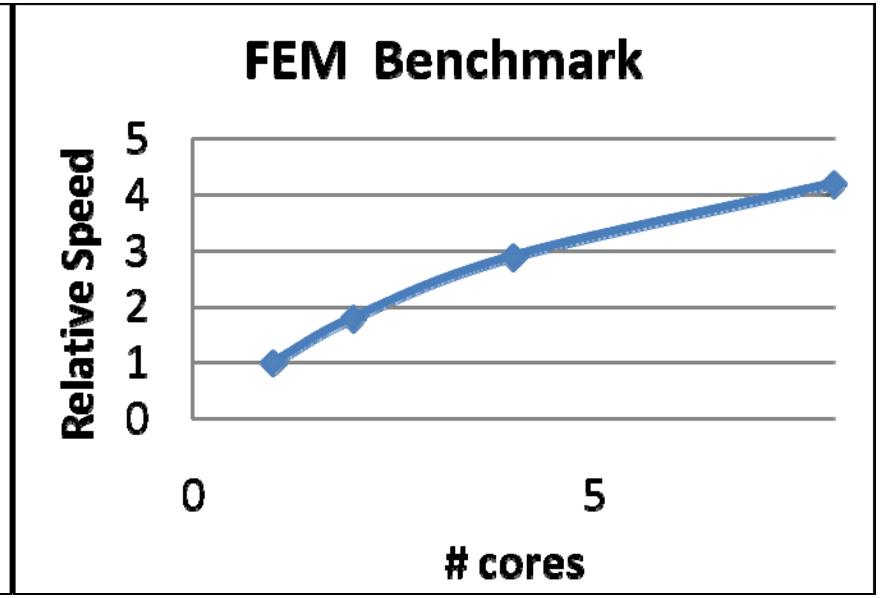
Limited Multicore Scalability of SLB Sparse Matrix Applications

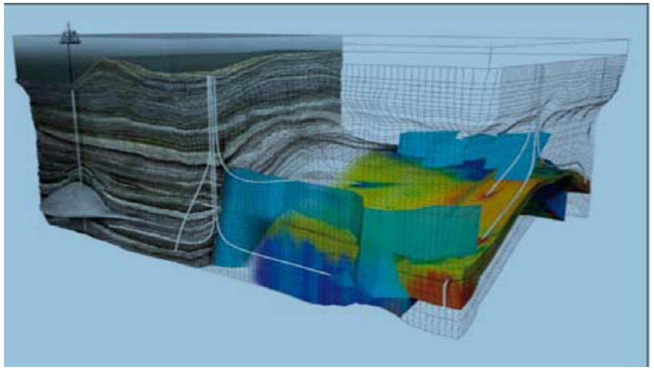
Eclipse Benchmark
(2 node Westmere 3.06 GHz)



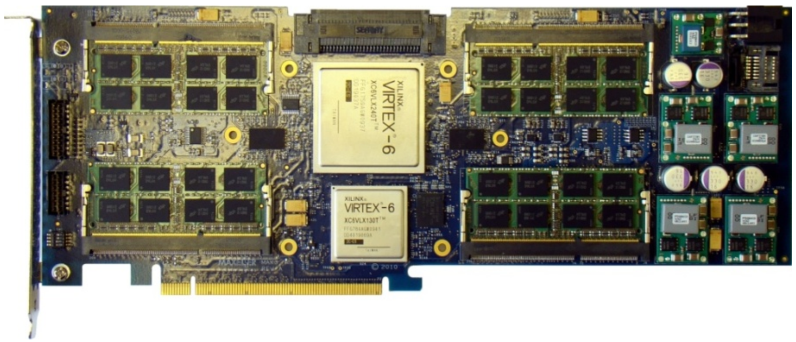


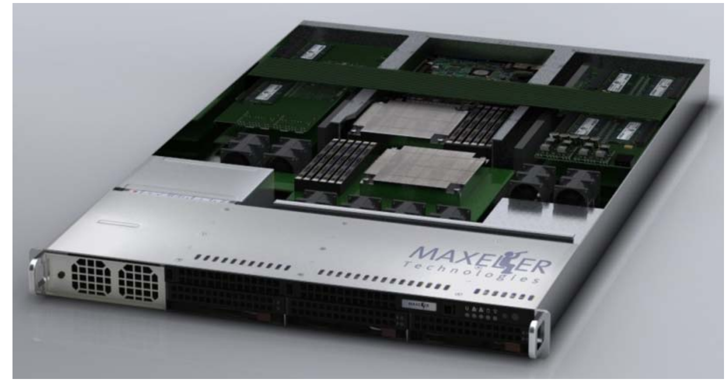
Visage – Geomechanics
(2 node Nehalem 2.93 GHz)

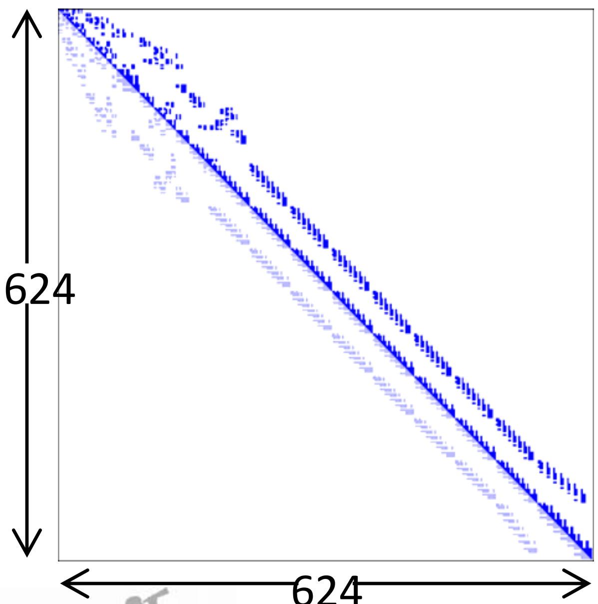


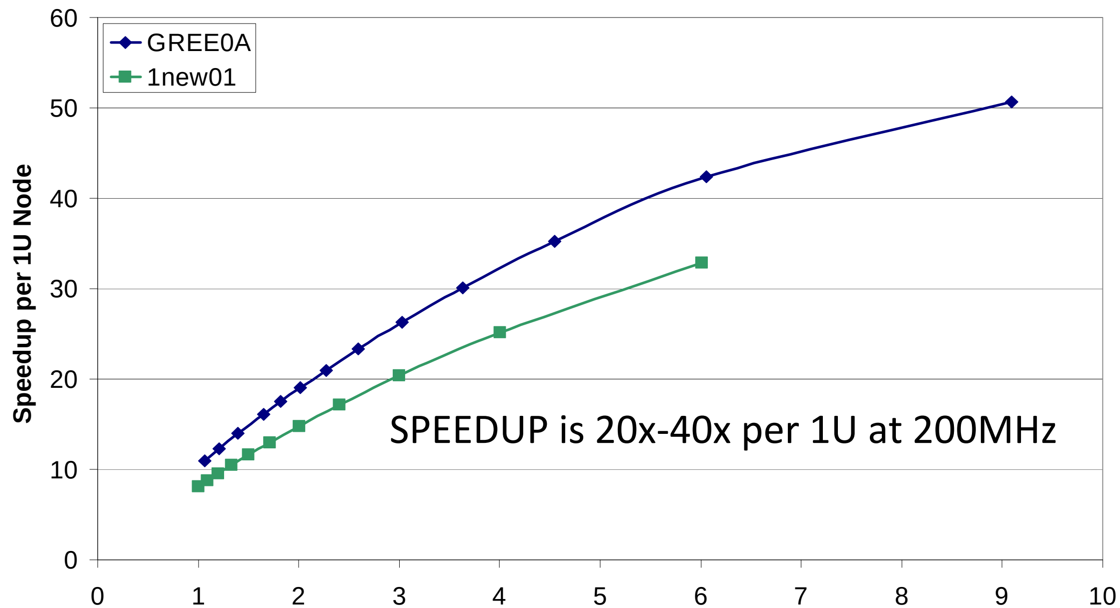


Sparse Matrix on FPGA









Maxeler Domain Specific Address and Data Encoding

3D Finite Difference Modeling

T. Nemeth et al, 2008

- Geophysical Model
  - 3D acoustic wave equation

$$\frac{\partial^2 p}{\partial t^2} = K\ \vec{\nabla} \cdot \left(\frac{1}{\rho} \vec{\nabla} p \right) + S(t)$$

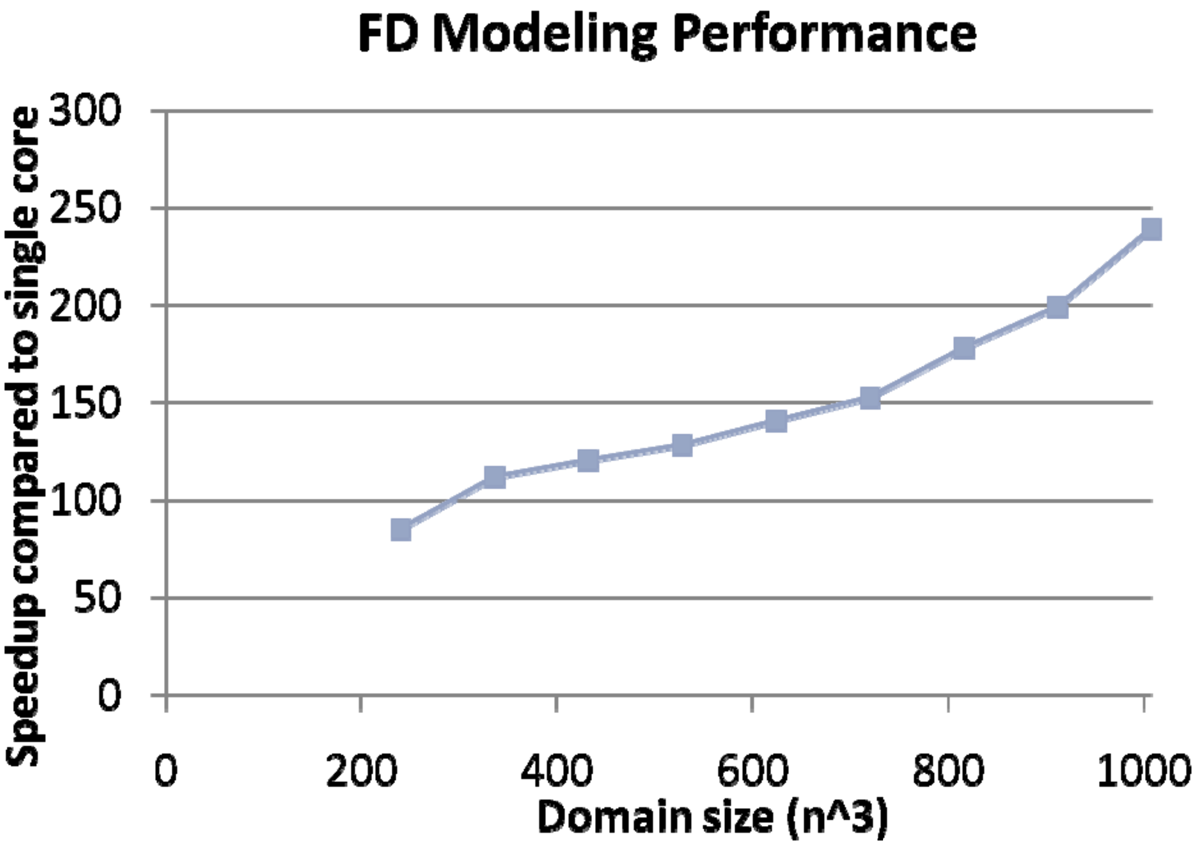
  - Variable velocity and density
  - Isotropic medium
- Numerical Model
  - Finite differences (12th order convolution)
  - 4th order in time
  - Point source, absorbing boundary conditions

MAXELER Technologies MAXIMUM PERFORMANCE COMPUTING

Chevron

Modelling Results

- Up to 240x speedup for 1 MAX2 card compared to single CPU core
- Speedup increases with cube size
- 1 billion point modelling domain using single FPGA card



Computations per Output Point on Intel Xeon (Convolution)

	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

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

Computations

- On average about a data cache miss per 10 floating point operations
- 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

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.

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

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 speedup possible)
- Operate at one-twentieth the frequency; reduce power and space

Achieved speedups (published results)

Problem	*Sponsor*	*Reference*	*Speedup per core*	*Speedup per server*
Conjugate Gradient Optimization	ENI-AGIP (seismic trace)	EDGE 2010	218X	-
Convolution	Chevron Schlumberger	SEG 2008 Hot Chips 2010 IEEE Micro 3/2011	250x	73 X
Sparse Matrix	Schlumberger	Hot Chips 2010		40 X
Monte Carlo simulation (credit derivative pricing)	JP Morgan	Derivative & Risk Mgmt Conf. (Paris, May 10)		79 x

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

So How Can Emulation (FPGA) Be Better Than The x86 Processor(s)?

- Multi core approach lacks robustness in streaming hardware (spanning area, time, power)
- Multi core lacks robust parallel software methodology and tools
- FPGAs form an unlikely basis for acceleration
- Success comes about from their flexibility in matching the DFG with a synchronous DFM and streaming data through and shear size > 1 million cells
- Effort and support tools provide significant application speedup

Conclusions 1

- Many applications are starved for computation
- The success of FPGA acceleration points to the weakness of evolutionary approaches to parallel processing: hardware (multi core) and software (C++, etc.), at least for some applications
- The automation of acceleration is still early on; still required: tools, methodology for writing apps., analysis methodology and (maybe) a new hardware basis

Conclusions 2

- In acceleration (and parallel processing): to find success, start with the problem not the solution
- Effort (sweat and tools) provides speedup, not silver bullets