Merrimac
Supercomputing with Streams

Mattan Erez

Concurrent VLSI Architecture Group
Computer Systems Lab, Stanford University

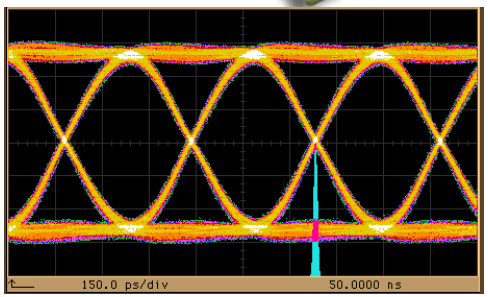
May 22, 2003

Streaming Scientific Computation Exploits the Capabilities of VLSI

- Modern VLSI technology makes arithmetic cheap
  - 100s of GFLOPS/chip today TFLOPS in 2010
- But bandwidth is expensive
- Streams change the ratio of arithmetic to bandwidth
  - By exposing producer-consumer locality
    - Not exploited by caches – no reuse, no spatial locality
- Streams also expose parallelism
  - To keep 100s of FPUs per processor busy
- High-radix networks reduce cost of bandwidth when it is needed
  - Simplifies programming

Computation is Inexpensive and Plentiful

nVidia GeForce4
~120 Gflops/sec
~1.2 Tops/sec

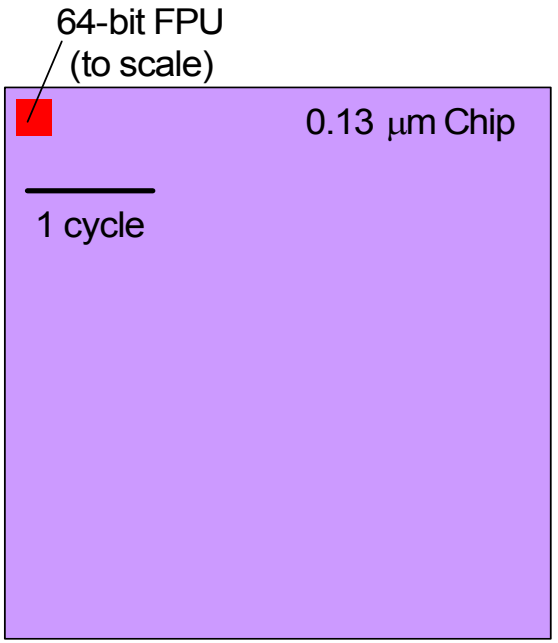


Velio VC3003
1Tb/s I/O BW

DRAM < $0.20/MB

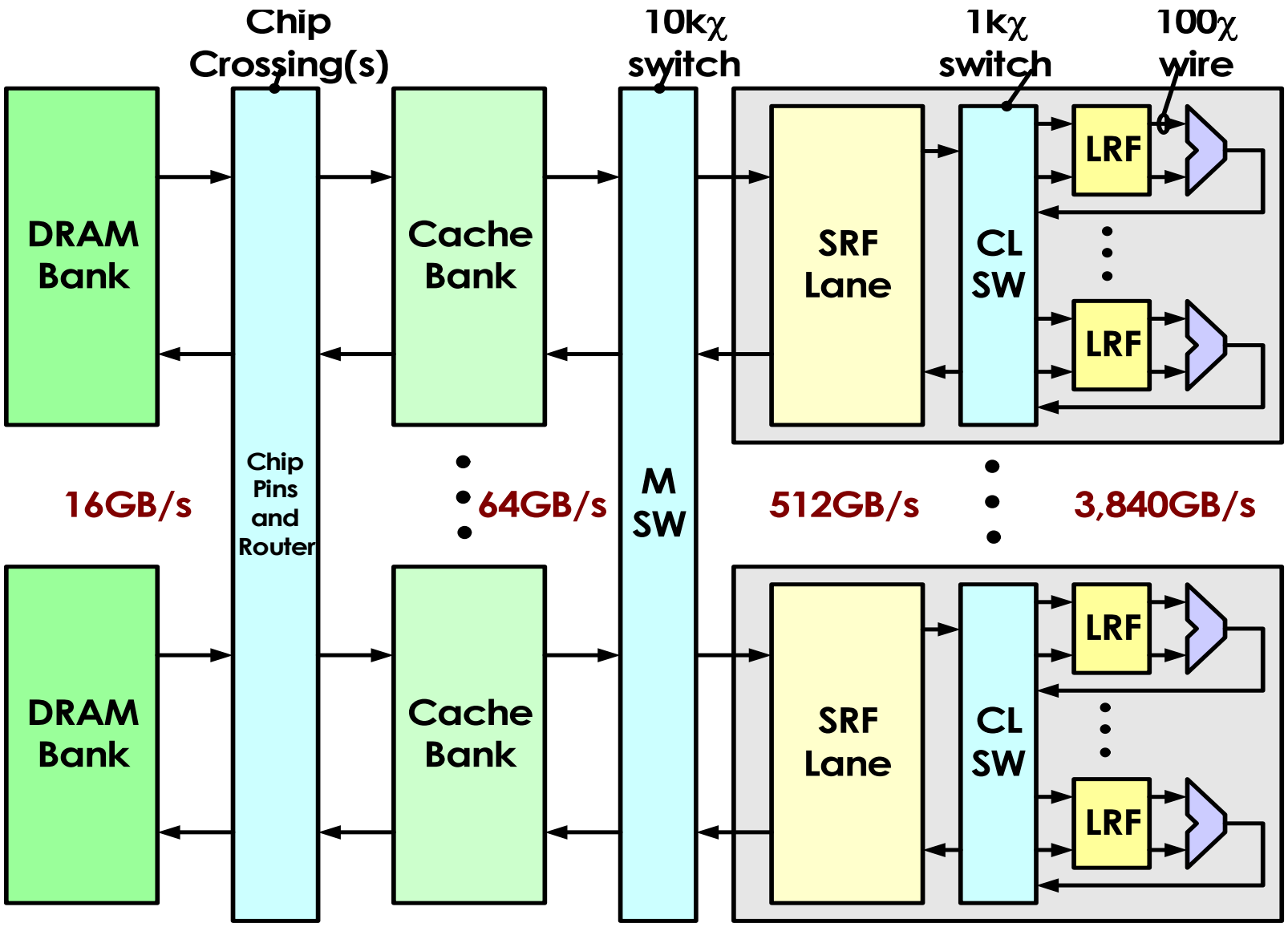
To Exploit VLSI Technology We Need:

- **Parallelism**
  - To keep 100s of ALUs per chip (thousands/board millions/system) busy
- **Latency tolerance**
  - To cover 500 cycle remote memory access time
- **Locality**
  - To match 20Tb/s ALU bandwidth to ~100Gb/s chip bandwidth
- Moore's Law
  - Growth of transistors, not performance

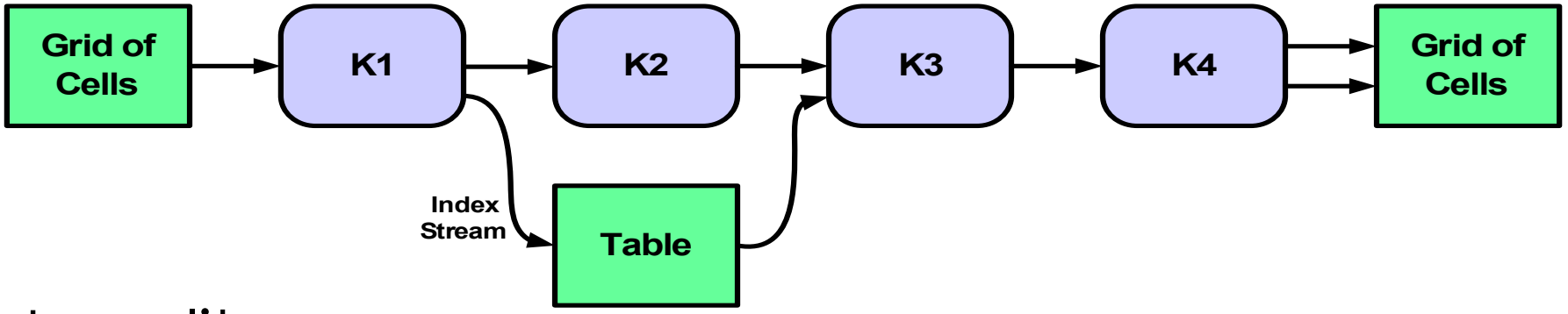


Arithmetic is cheap, global bandwidth is expensive
Local << global on-chip << off-chip << global system

Stream Architecture Makes Communication Explicit – Exploits Parallelism and Locality

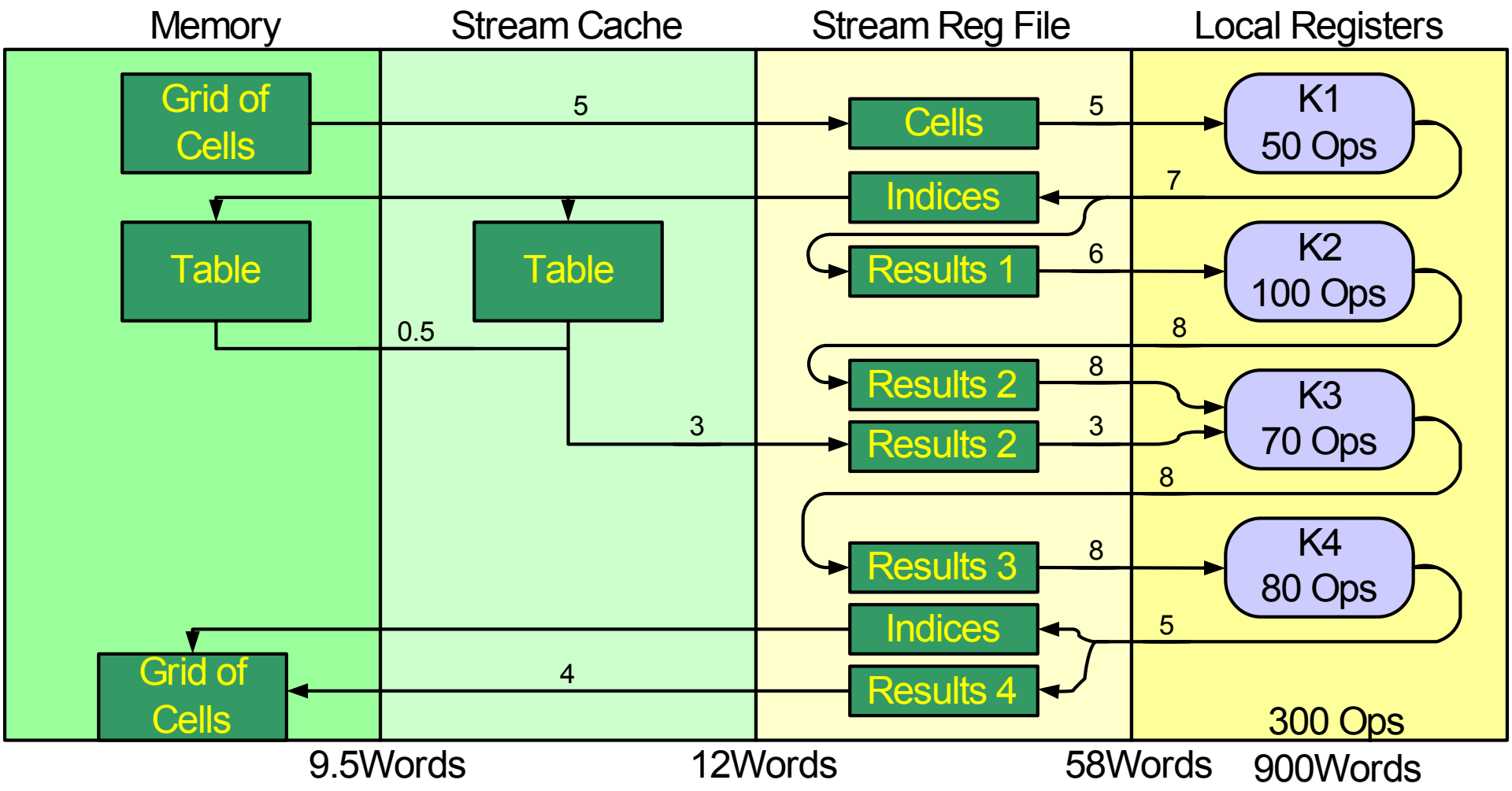


Stream Programming Exposes Parallelism and Locality



- Locality
  - Producer-consumer within a kernel (local register file)
  - Producer-consumer across kernels (on-chip SRF)
  - Stream locality (spatial)
- Parallelism
  - Data Level Parallelism across stream elements
  - Task Parallelism across kernels

Streamed Applications Exploit Producer-Consumer Locality to Exploit BW Hierarchy



Stream program matches application to BW Hierarchy 1:1:6:100

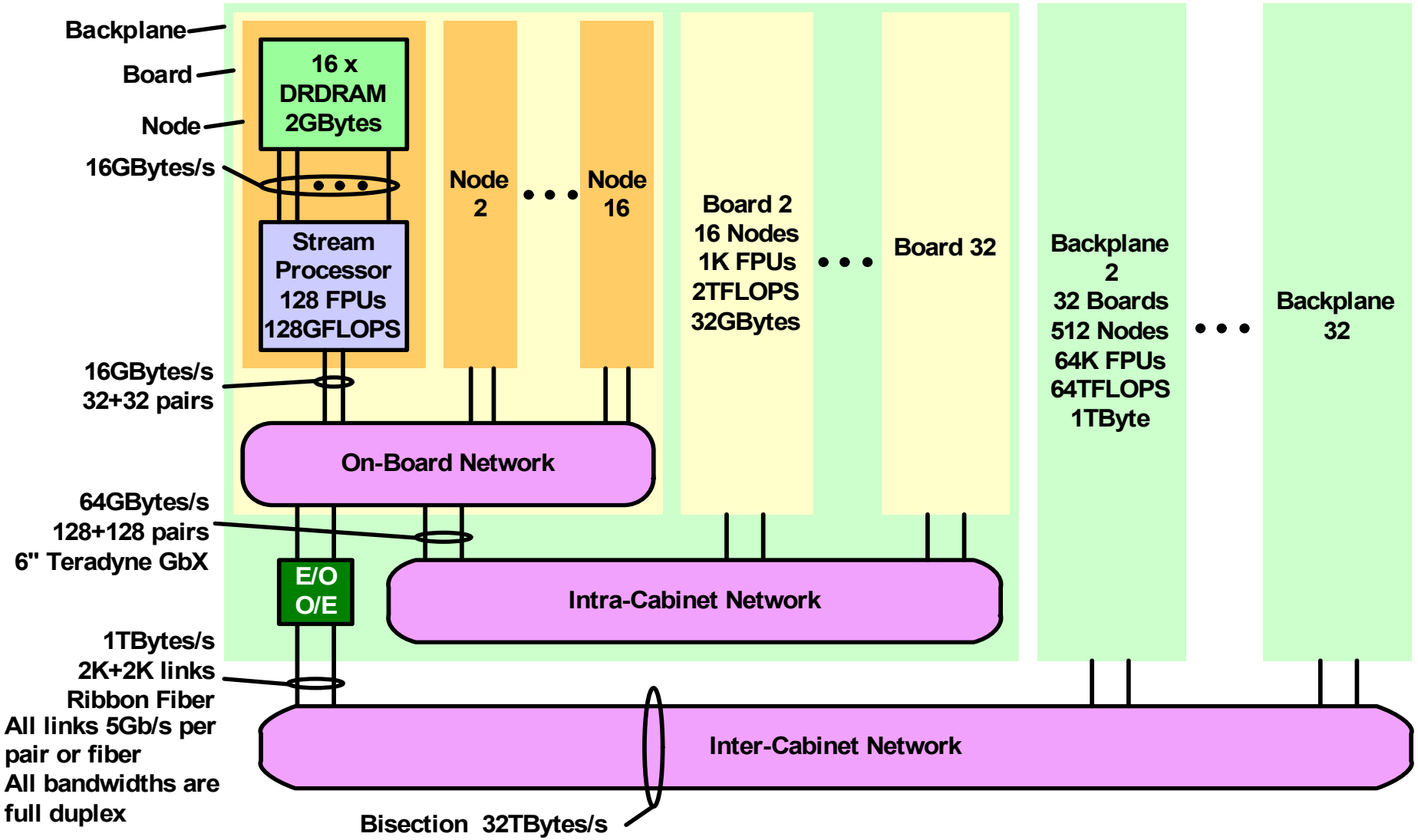
A streaming supercomputer exploits the arithmetic density of VLSI to realize an efficiency of $6/GFLOPS and $3/M-GUPS

Capability AND Capacity

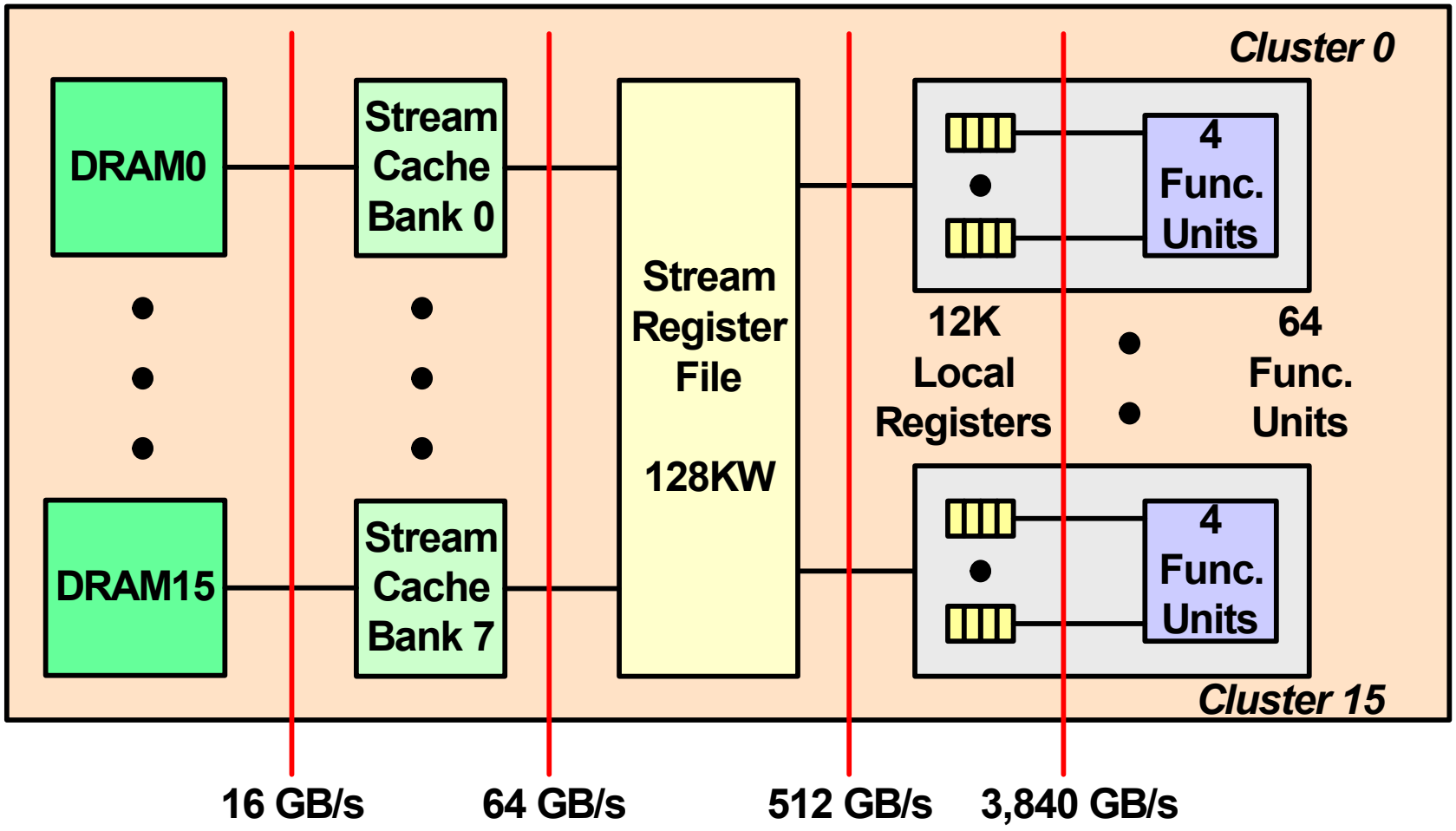
A PFLOPS machine with only 8,192 nodes (or less)

A TFLOPS workstation for $20,000 (parts cost)

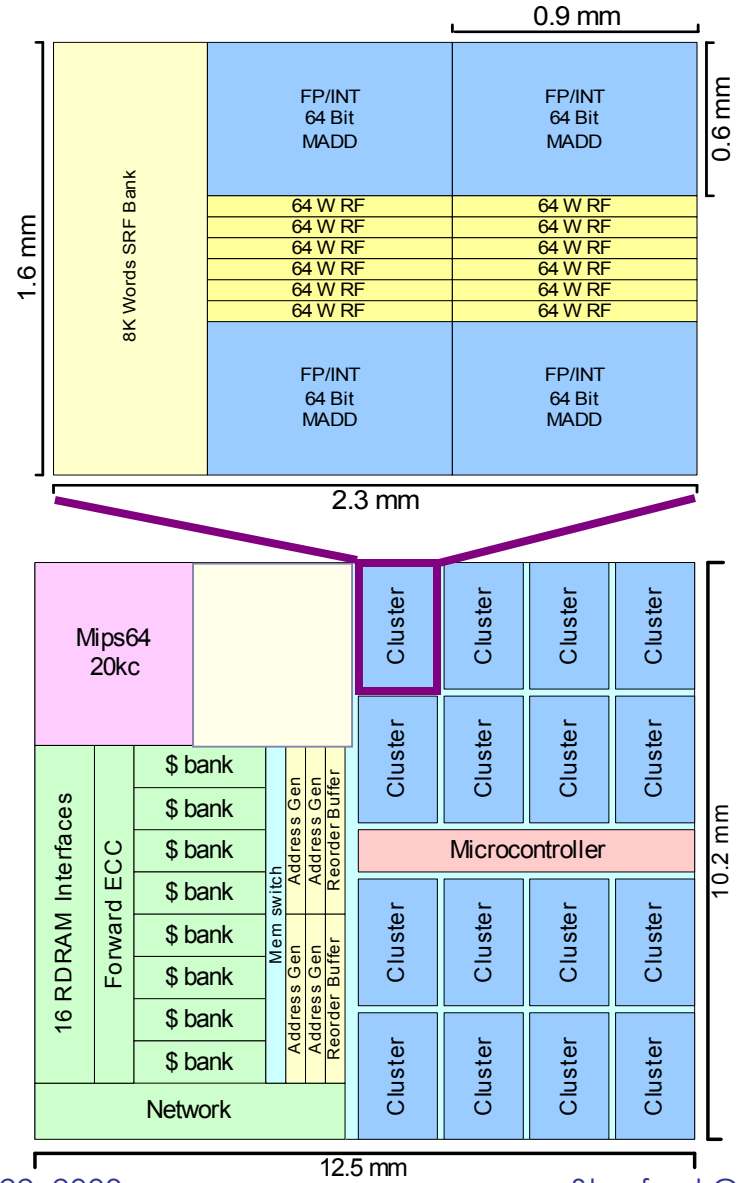
Merrimac - a Streaming Supercomputer



Merrimac Node

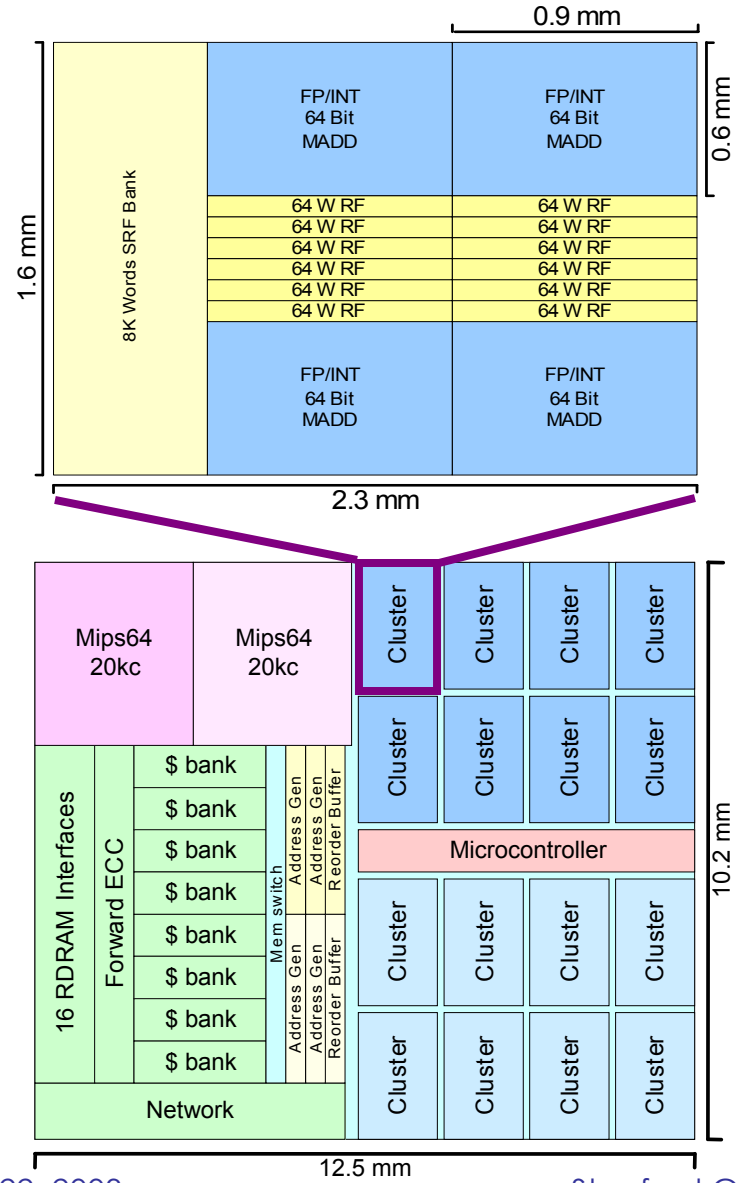


Merrimac Processor



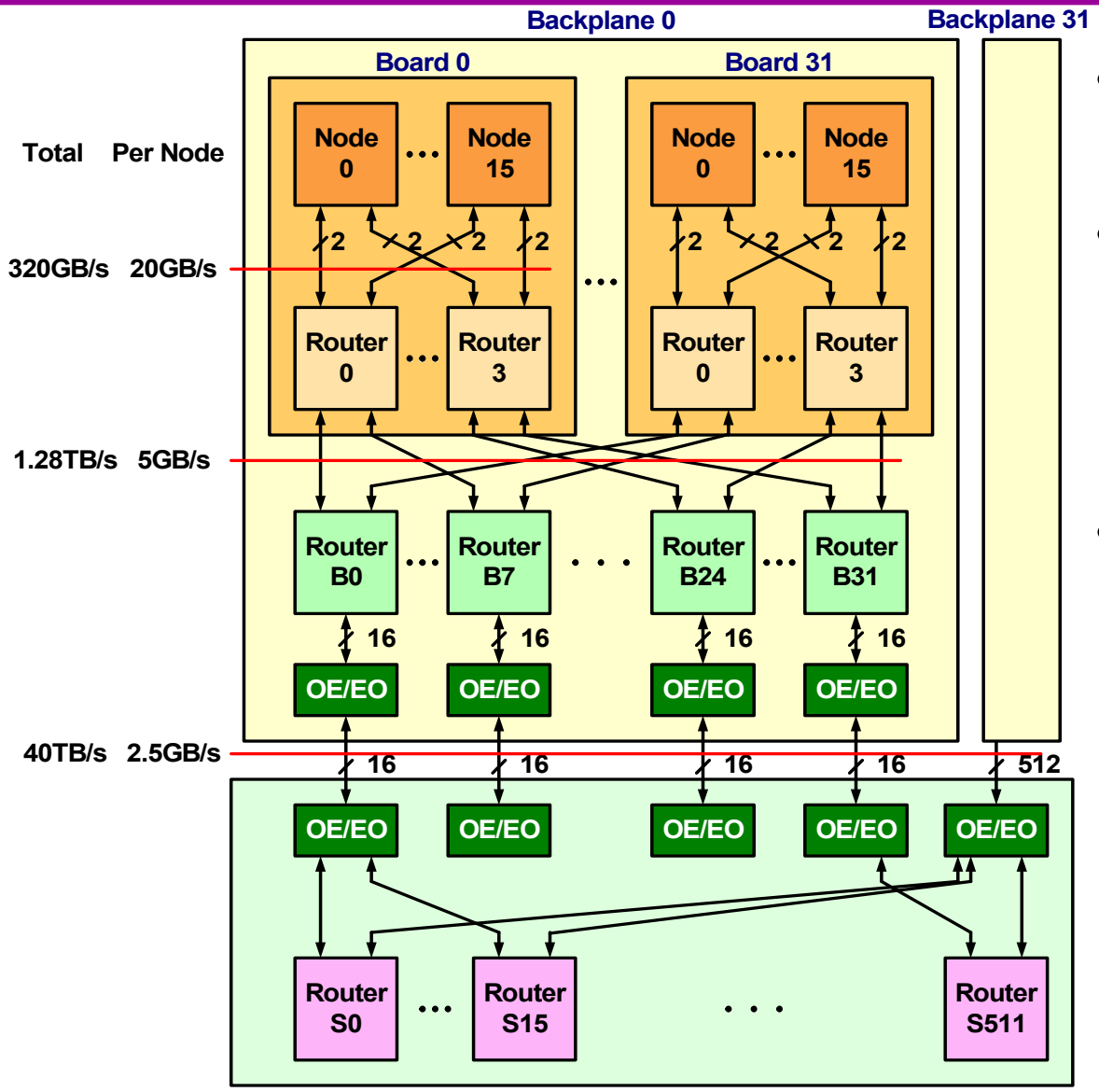
- 90nm tech (1 V)
- ASIC technology
- 1 GHz (37 FO4)
- 128 GFLOPs
- Inter-cluster switch between clusters
- 127.5 mm2 (small ~12x10)
  - Stanford Imagine is 16mm x 16mm
  - MIT Raw is 18mm x 18mm
- 32 Watts (P4 = 75 W)

Merrimac Processor



- 90nm tech (1 V)
- ASIC technology
- 1 GHz (37 FO4)
- 128 GOPs
- Inter-cluster switch between clusters
- 127.5 mm2 (small ~12x10)
  - Stanford Imagine is 16mm x 16mm
  - MIT Raw is 18mm x 18mm
- 32 Watts (P4 = 75 W)

High Radix Routers Enable Economical Global Memory



- Flat memory bandwidth within a 16-node board
- 4:1 Concentration within a 32-node backplane, 8:1 across a 32 backplane system
- Routers with bandwidth B=640Gb/s route messages with length L=128b
  - Requires high radix to exploit

Bandwidth Hierarchy Enabled by Streams

Level	Bandwidth per Node (GB/s)
Cluster registers	3,840
Stream register file	512
Stream cache	64
Local Memory	16
Board Memory (16 Nodes)	16
Cabinet Memory (1K Nodes)	4
Global Memory (16K Nodes)	2

Streaming

Network

Rough Per-Node Budget

Item	Cost	Per Node
Processor chip	200	**200**
Router chip	200	69
Memory chip	20	**320**
Board	1000	63
Router Board	1000	2
Backplane	5000	10
Global Router Board	5000	5
Power	1	50
Per-Node Cost		**718**
$/GFLOPS (64/node)		**6**
$/M-GUPS (250/node)		**3**

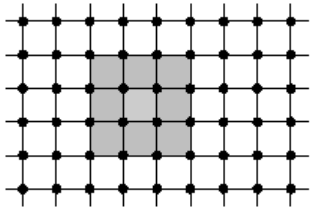
Preliminary numbers, parts cost only, no I/O included

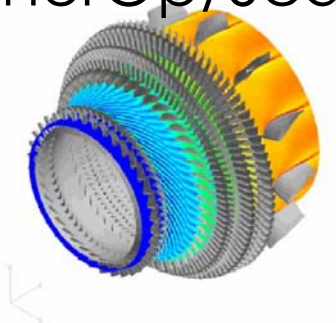
Gentle Slope to Port Applications to Merrimac

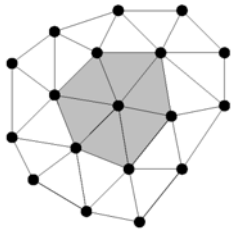
like OpenMP

Brook and Brooktran – Stream Extensions with Multiple Dimensions and Irregular Grids

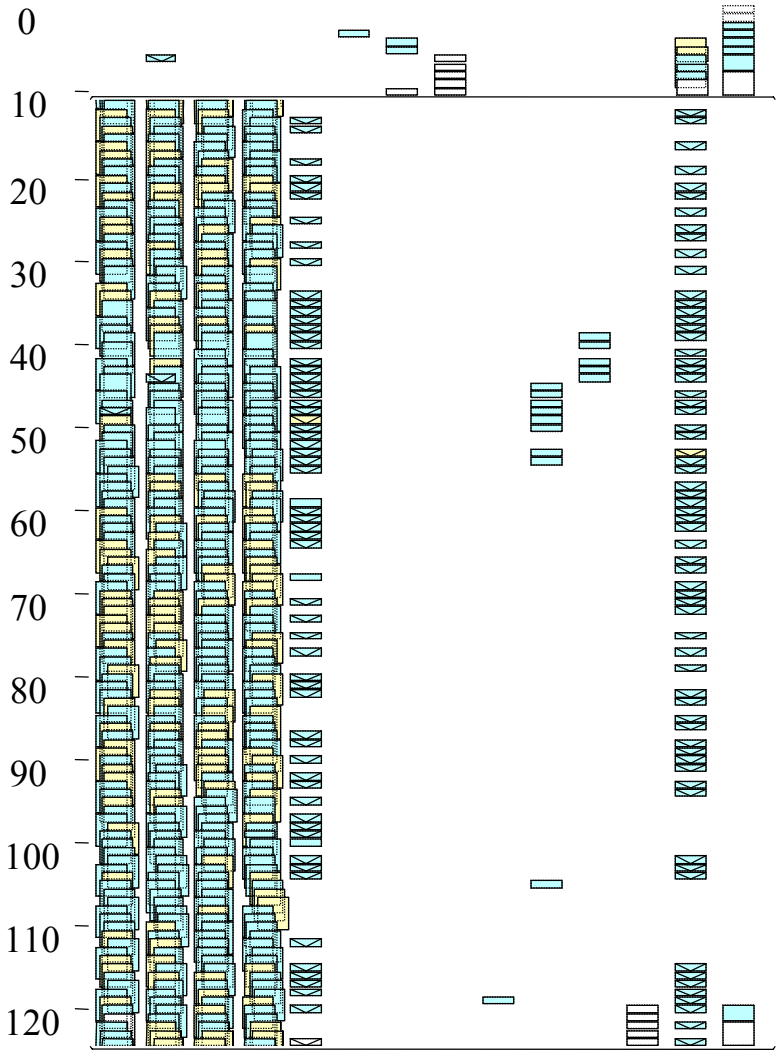
- Stream code intermixed with "scalar" code
  - **C** – Brook **Fortran** – Brooktran
  - Easy migration - only port the time-consuming kernels
- Locality and Parallelism
  - Producer-consumer locality
  - No global memory references within a kernel
  - No retained state in kernels (reductions allowed)
  - Stream elements are processed independently
- Stream operators for defining neighbors
- Gather/Scatter and GatherOp/ScatterOp



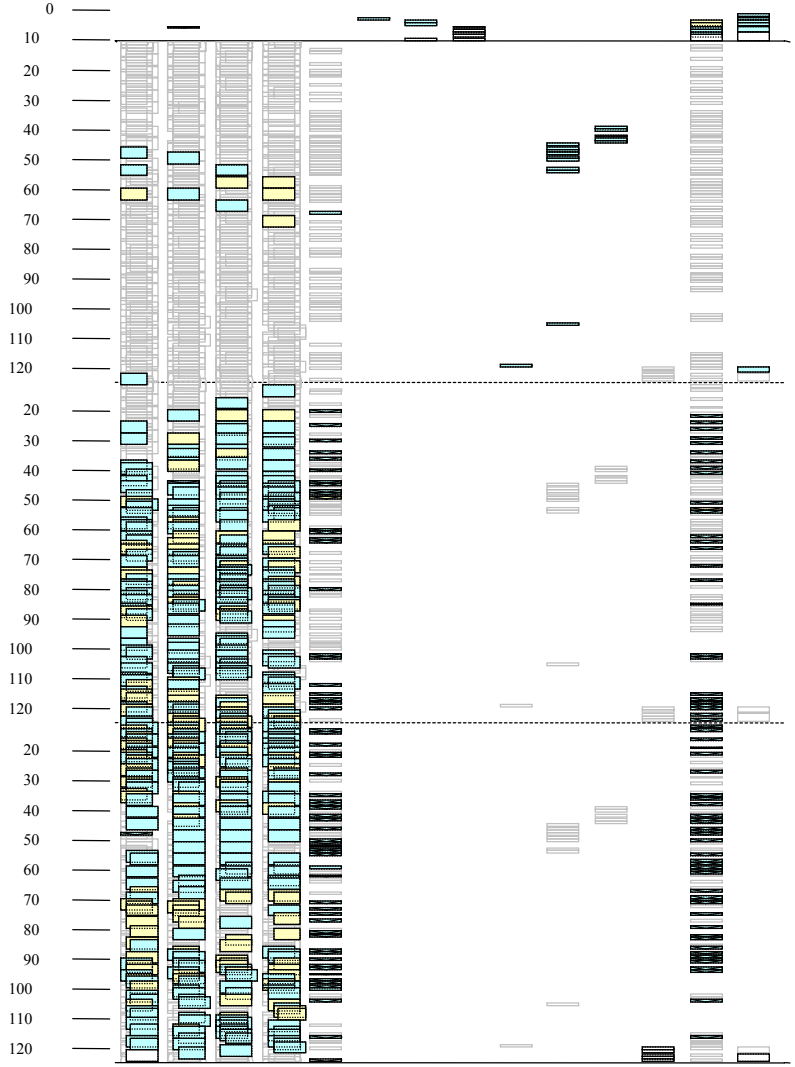




Stream Compiler Achieves Near Optimum Kernel Performance



single iteration

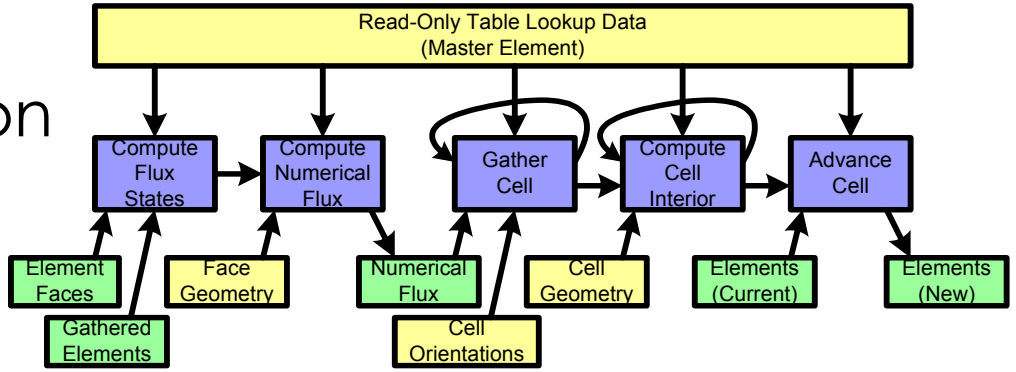


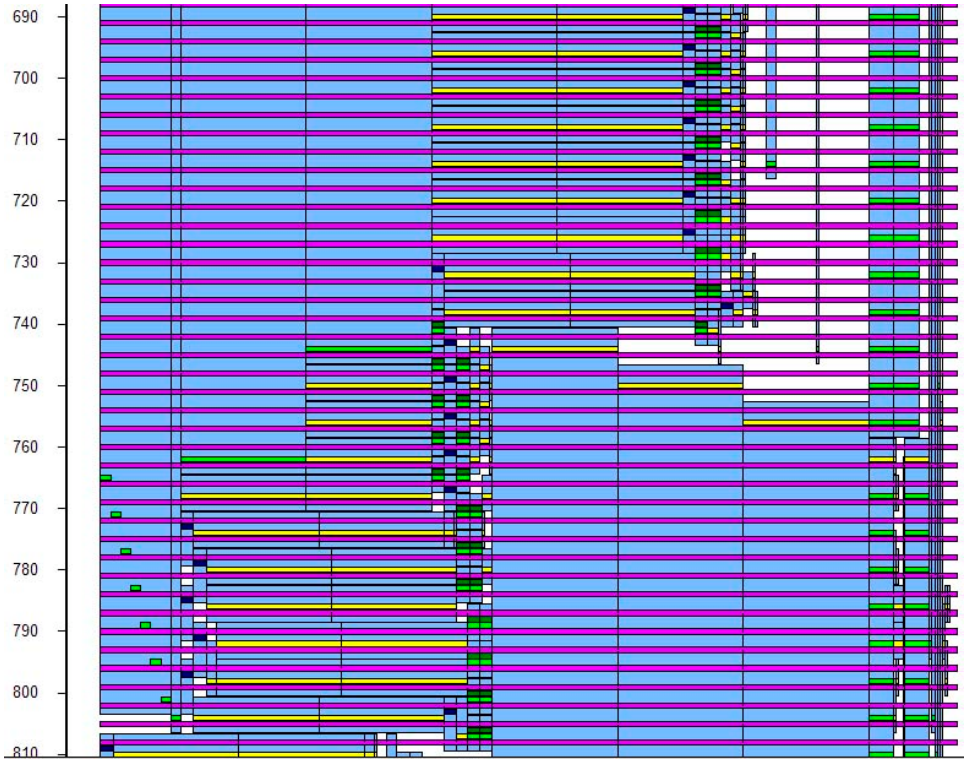
software pipeline shown

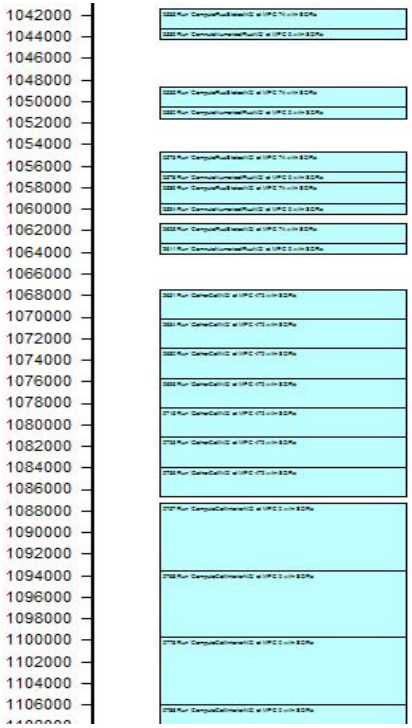
ComputeCellInterior Kernel from StreamFEM application

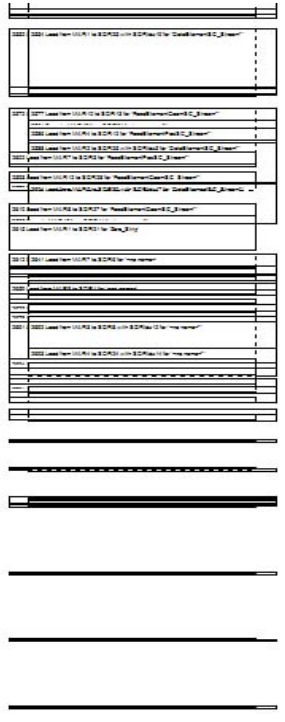
Stream Compiler Reduces Bandwidth Demand Compared to Caching

StreamFEM application









Capable Hardware Makes Software Easier

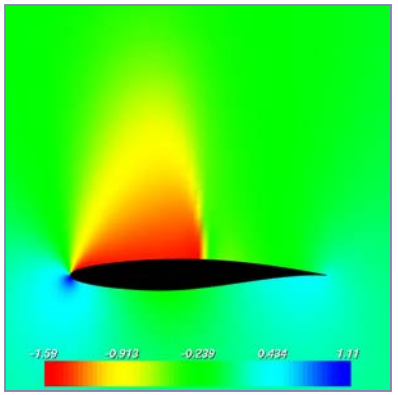
- Nearly flat global bandwidth (10:1)
  - Supports non-local applications
  - Eliminates need to partition and place application
- Fine grain remote access
  - Single word gather/scatter
  - Eliminates need to restructure application for spatial locality
- Leaves the programmer to focus on
  - Writing a correct program
  - Expressing parallelism

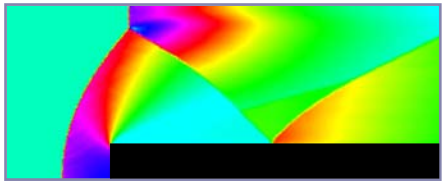
Several scientific applications have been demonstrated on a stream processor simulator

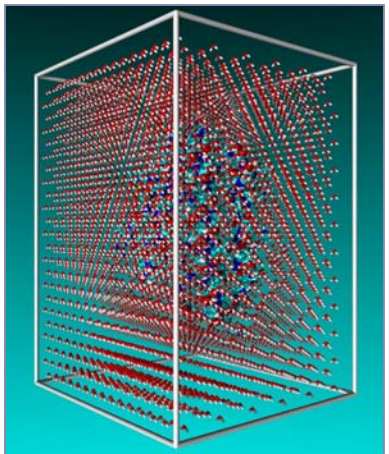
They all exhibit stream locality and achieve a high fraction of peak performance

Scientific Streamed Applications

- **StreamFLO** is a streaming version of FLO82, [Jameson], for the solution of the inviscid flow around an airfoil
  - Uses a cell centered finite volume formulation with a **multi-grid** acceleration to solve the 2D Euler equations
- **StreamFEM** implementation of the Discontinuous Galerkin (DG) Finite Element Method (FEM) (Tim Barth, NASA)
  - 2D or 3D triangulated domains
  - Increasingly complex PDEs
    - Scalar advection (1 PDE), Euler (4 PDEs), Magnetohydrodynamics (6 PDEs)
  - Piecewise polynomial function
    - Constant (1 dof), Linear (3 dof), Quadratic (6 dof), Cubic (10 dof)
- **StreamMD** molecular dynamics simulation
  - Box of water molecules
  - Electrostatic and Van der Waals interactions
  - Gridded to accelerate approximate force calculation







Scientific Applications Stream Well

Application	Sustained GFLOPS[1]	FP Ops / Mem Ref	LRF Refs	SRF Refs	Mem Refs
StreamFEM2D (Euler, quadratic)	32.2	23.5	169,505,648 (93.6%)	10,299,776 (5.7%)	1,354,448 (0.7%)
StreamFEM2D (MHD, cubic)	33.5	50.6	733,294,080 (94.0%)	43,762,752 (5.6%)	3,165,280 (0.4%)
StreamMD	23.3[2]	14.3	427,743,216 (96.5%)	9,505,099 (2.1%)	5,978,848 (1.4%)
StreamFLO[3]		50	(96%)	(2%)	(2%)

1. Simulated on a machine with 64GFLOPS peak performance
2. Expect to get closer to 45 GFLOPS once a false-dependency is removed
3. Based on estimates of key application kernels

Scientific stream applications match BW Hierarchy >90% local, <2% off-chip memory, ~50% of peak performance

Conclusion

- The problem is bandwidth – arithmetic is cheap
  - Forget GFLOPS,
  - forget traditional P:M:C ratios – balance by incremental return
- Streams expose and exploit locality and concurrency
  - Makes communication explicit
  - Keeps most (>90%) of data operations local (>1TB/s, 10pJ)
  - Increases *arithmetic intensity* (arithmetic rate/bandwidth)
  - Enables compiler optimization at a larger scale than scalar processing
  - $15/GFLOPS sustained on 'local' parts of applications
- A capable network provides high global bandwidth
  - 20GBytes/sec on board (16 processors)
  - 2.5GBytes/sec globally (16K processors)
  - 25% of machine cost
  - $4/M-GUPS on 'non-local' parts of applications
- Must be scalable to be economically feasible
  - Scalable – 2 TFLOPS board for $40K to 32-backplane 2 PFLOPS for $40M