Disruptive Technologies: Field Programmable Gate Arrays

Maya Gokhale

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

June 14, 2007

FPGA architectures

Heterogeneous System on Chip

•Configurable Logic Blocks, hierarchically aggregated

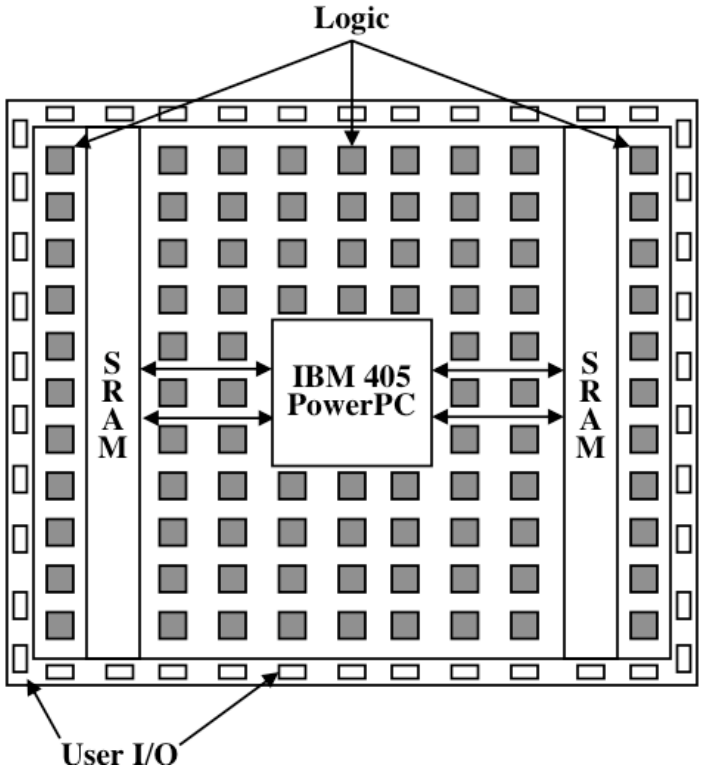
•Embedded memory blocks, configurable in width, number of ports, depth

•On-chip ALUs such as multipliers and other DSP blocks; on-chip processors

•On-chip external I/O interfaces - Gigabit transceivers, PCI core

•Extensive routing fabric for flexible interconnection

•Used as glue logic, fast prototyping, logic emulation, embedded systems, … reconfigurable computing



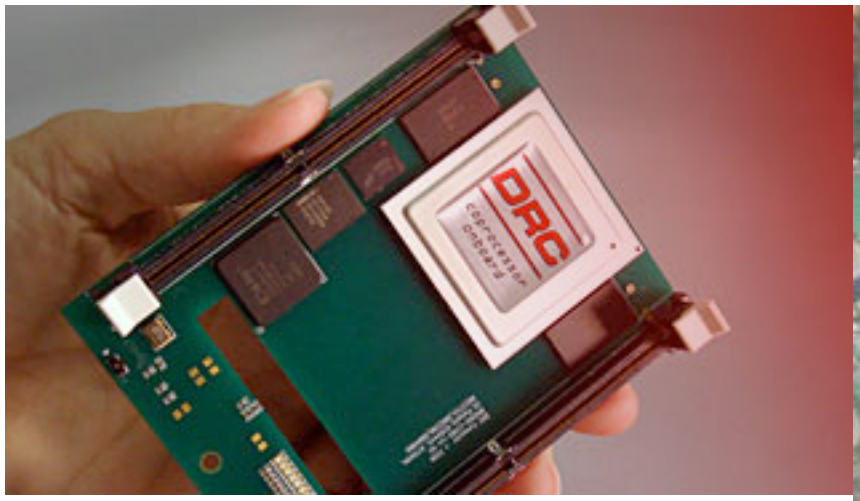
Computation Directorate

Reconfigurable Computing Systems

- Acceleration engine on I/O bus of PC, workstation
- Closely coupled co-processor - dual socket FPGA/Opteron
- Cluster with FPGA engine on each node
- Reconfigurable supercomputer - high performance interconnect between nodes

Computation Directorate

FPGA co-processors



- Versatile architecture
- Part of Cray “Adaptive Supercomputer” roadmap
- Possible use in future “appliance” architectures (Exegy)

Reconfigurable computing pros/cons

Pro

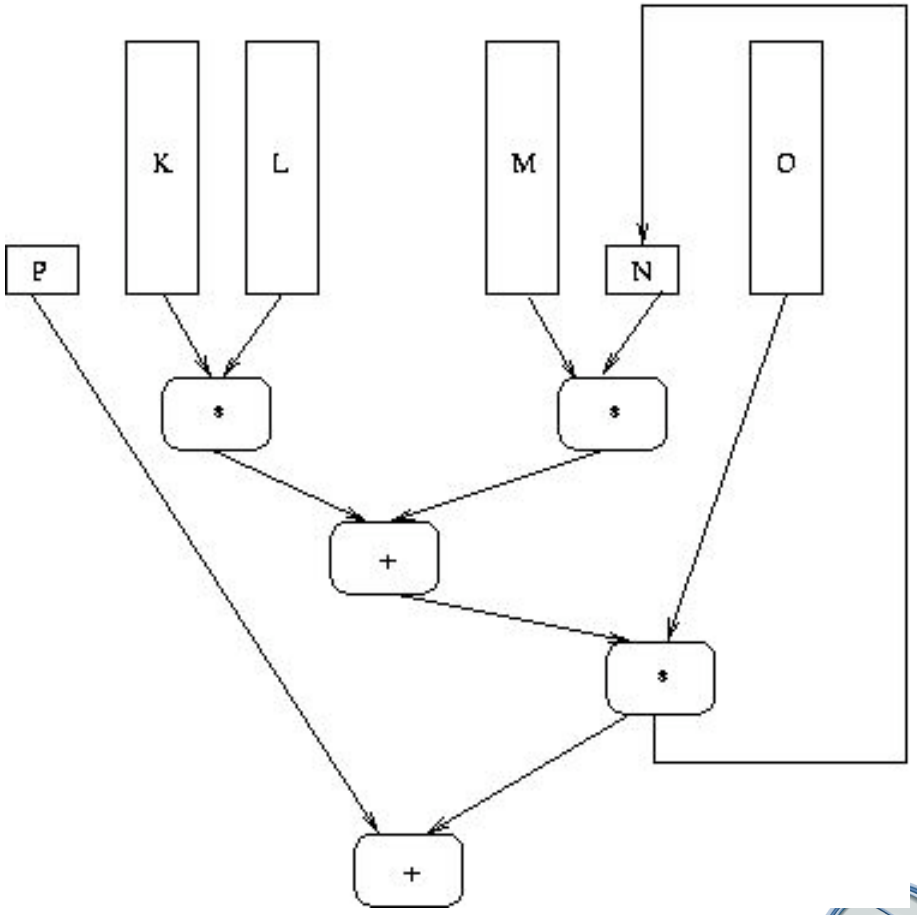
- Performance - order of magnitude on the right kernels
- Footprint - small form factor for the delivered performance
- Power/energy - 25W vs. 100W
- Re-programmable - not just a single application kernel

Con

- Cost
  - Hardware cost is many times COTS hardware
  - Design tools are expensive
- Limited application domains
  - Well suited to small integer, DSP computation, but not suited to 64-bit floating point
  - Amdahl's Law: Co-processor mode is inherently challenging
- Algorithm development difficulty
  - FPGA architecture is difficult target: Very large search space for placing and routing a design

Example

```
n=0
p=0
for i
    n = (K[i]*L[i]+M[i]*n)*O[i]
    p = n+p
```



- Many choices for instruction level parallelism
  - multiplies/adds in parallel or sequential?
    - area/speed tradeoff
    - affects loop-level pipelining
  - Number of clock cycles to compute n & p
    - affects clock speed

Pipeline loop alternatives

- Pipelined: One memory for all arrays

Initiate a new loop iteration every 4 clock cycles.

8 stage pipeline

Stage 0: Update array pointers; issue read of M

Stage 1: Increment i; issue read of K; save M in register

Stage 2: temp1 = M*n; issue read of L; save K in register

Stage 3: issue read of O; save L in register

Stage 4: temp2 = K*L; save O in register

Stage 5: temp3 = temp1 + temp2

Stage 6: n = temp3*O

Stage 7: p = p+n

- Pipelined: Four memories

Initiate a new loop iteration every 2 clock cycles

6 stage pipeline

Stage 0: Issue reads of K, M, L, O

Stage 1: Increment i; save M, L, O in registers

Stage 2: temp1 = K*L; temp2 = M*n

Stage 3:temp3 = temp1 + temp2

Stage 4: n = temp3*O

Stage 5: p = p+n

- Combinational: Four memories

Initiate a new loop iteration every clock cycle

1 stage pipeline

Stage 0: Perform all multiplies and add, store results in registers

For XC2V2000-6

153/10752 slices

5/56 multipliers

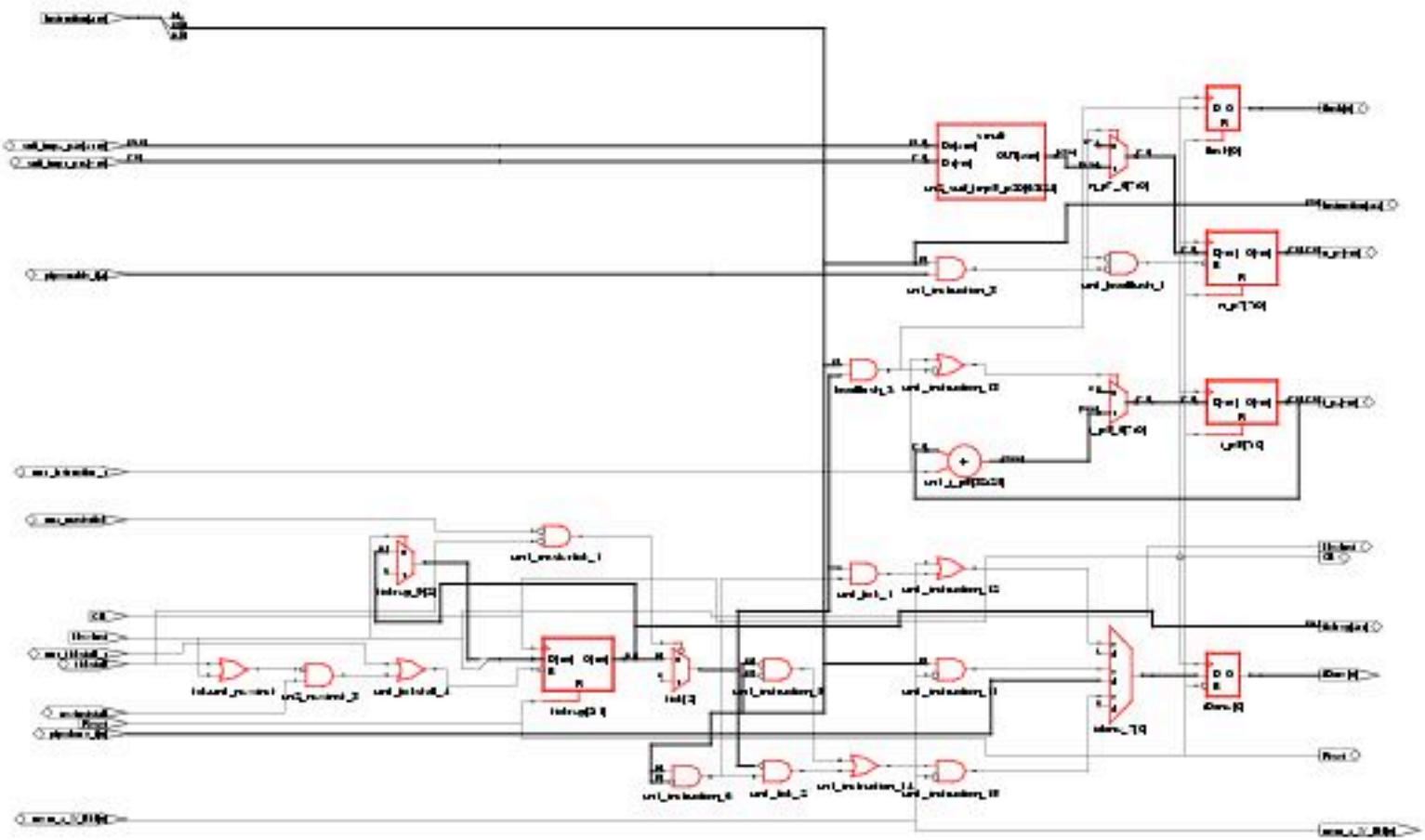
66 MHz

131/10752 slices

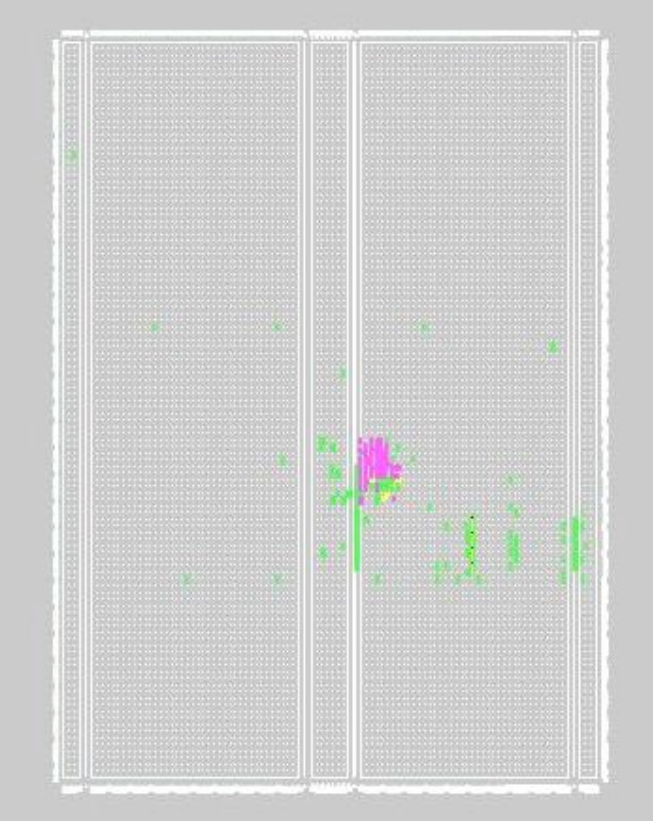
3/56 multipliers

219 MHz

Schematic view from synthesis



Schematic view from Place&Route



Pipeline controller

Datapath: adders/multipliers/registers

Sequence controller

Device speed data version: PRODUCTION 1.118 2004-03-12.
Device utilization summary:

Number of External IOBs	442 out of 624	70%
Number of LOCed External IOBs	0 out of 442	0%
Number of MULT18X18s	3 out of 56	5%
Number of SLICEs	131 out of 10752	1%
Number of BUFGMUXs	1 out of 16	6%

The AVERAGE CONNECTION DELAY for this design is: 1.217
The MAXIMUM PIN DELAY IS: 4.558
The AVERAGE CONNECTION DELAY on the 10 WORST NETS is:4.005

Computation Directorate

Reconfigurable supercomputing applications

- Simulations
  - physical phenomena
  - physics-based codes
  - behavior of physical entities in space and time
  - run single large problem with high interaction between parts
- Analysis
  - Bioinformatics - BLAST, genomic expression
  - Financial predictions - Monte Carlo methods to (eg.) price derivatives
  - Text mining
  - Database search
  - Run large number of independent problems on compute/data intensive backend processors
  - Gather results at front end

Computation Directorate

Assessing FPGAs as simulation co-processors

- Study execution profile
  - oprofile, PAPI, TAU
  - quantify time spent at loop or even line granularity
  - find representative data sets
    - execution profile may vary greatly depending on data set
  - want 90% time in a small kernel, preferably library function
- Study code of likely acceleration candidates
  - data type - integer, single precision FP, double FP
  - types of operations - divides, transcendental functions
  - numbers of operations - how many FP units are needed
- Study data profile
  - data consumed and produced in a region must be communicated between global microprocessor memory and FPGA board memory
  - need to know amount of data transferred (per loop iteration)
  - need to know if communication and computation can be overlapped
- ERSA 2005, RSSI 2006

Computation Directorate

Scientific simulation profiles

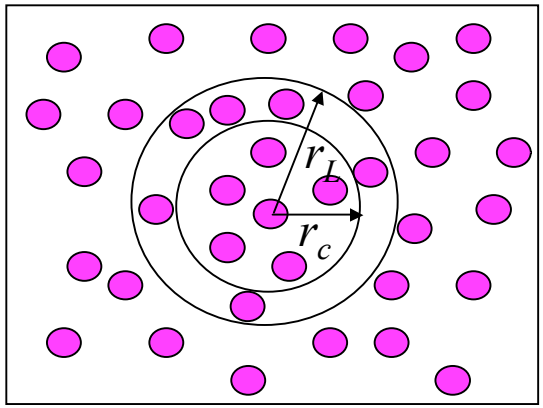
Code	Compute Kernel %	Comment
LINPACK	74%	DGEMM
MILC	55%	Collection of matrix algebra routines
GAMESS	40%	EDIMER Large, complex routine
POP	10%	Conjugate gradient solver
GROMACS	55%	In 5 routines, already optimized in sse

Speedup of 2X-3X at the very most

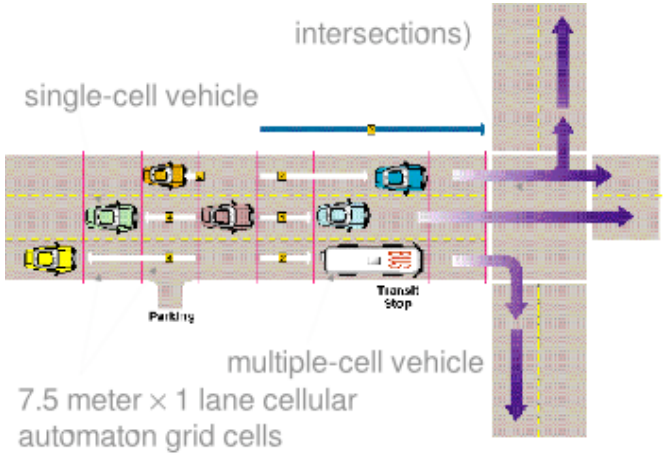
Computation Directorate

Scientific simulations on FPGAs

- Molecular dynamics
- Non-bonded force calculation
- Specialized floating point format (BU) **5X** over Pentium on Annapolis Micro PCI card
- Single precision floating point format (USC) **2X** over Pentium on SRC node



- Road traffic simulation
- Massively parallel cellular automaton
- **2X** on Cray XD1 node



90-10 is hard to find in scientific simulations

- Acceleration using only library routines will be negligible for scientific codes.
  - Even Linpack needs at least 5X DGEMM acceleration, which has not yet been demonstrated for double precision FP.
- Acceleration of compute kernels is problematic.
  - Long, complex double precision code sequences: not a good fit for FPGA
  - Collection of little hot spots whose data structures are enmeshed in surrounding serial code
- The Amdahl's Law limitation also applies to other co-processors.

Computation Directorate

FPGAs for analysis problems

- Signal and image processing
  - Integer and single precision FP
  - Amenable to streaming, pipelining
  - Compute within the data acquisition pipeline
  - Lots of working, real world implementations, eg. Cibola Flight Experiment with 9 Virtex 1000's for signal processing on-board a satellite, launched March 2007
- Graph algorithms
  - Point to point shortest path on road network (a la mapquest): **Zach Baker** at LANL
  - All-to-all shortest path of very large sparse semantic graphs: **Scott Kohn** and **Andy Yoo** at LLNL

Example: Point to point shortest path

- Used for route planning in TranSIMs, simulating road traffic on road networks in large metropolitan areas
- Opportunity for parallel execution of route planning
  - 100K - 1,000K routes to compute
  - Can complete in nearly arbitrary order
  - Lat/long of all road nodes provided
- Latency dominated computation
  - Mitigate through application-specific multi-threaded approach
- Implemented on Cray XD1 node
  - Dual Opteron + Xilinx Virtex2Pro50 FPGA for every
  - RapidArray HT connection to FPGA

Point-to-point shortest path

- A* algorithm
  - Uses distance to destination to decide which possible paths to pursue
  - Hardware-friendly priority queue implementation needed
  - Bandwidth to road network graph critical to performance

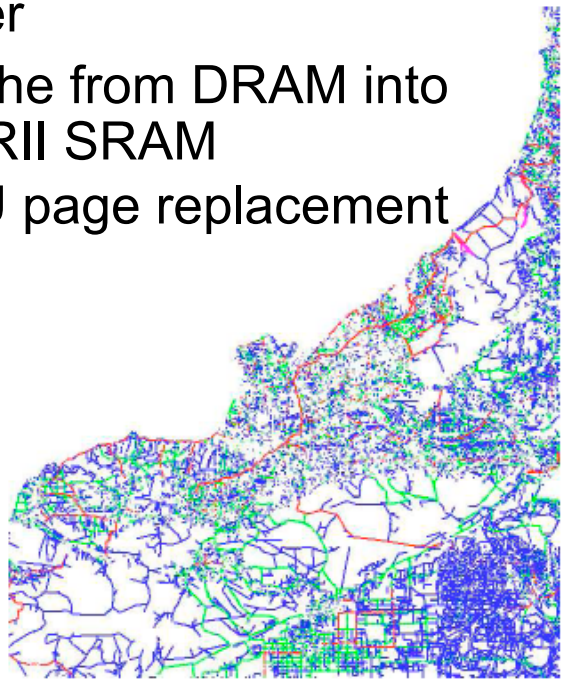
```
add_queue(start, 0)
while(queue != empty)
  u = extract_min(queue)
  if(u.explored == true)
    continue
  elsif(u == dest)
    return path
  else
    foreach(edge (u,v) out from u)
      d(v) = distance(v, dest)
      add_queue(v, path_dist + d(v))
      previous[v] = u
    u.explored = true
```

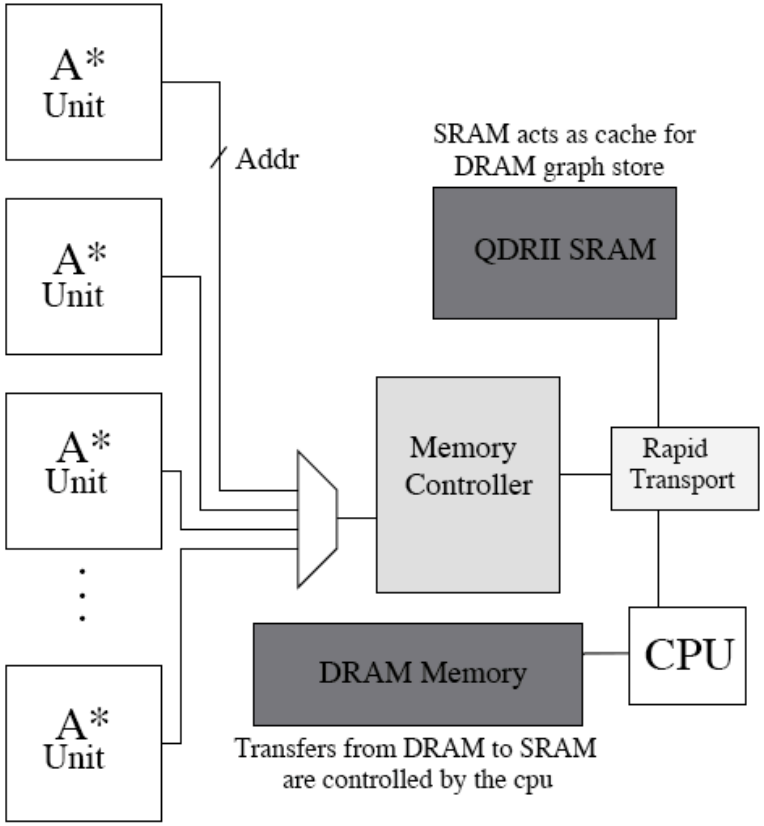
Adaptations for hardware: priority queue

- Software A* uses Fibonacci Heap
  - O(log(n)) average performance
  - Complicated data structure
  - Heap can grow without bound during execution
- Hardware approach uses bubble sort!
  - Needs only a single memory port
  - Sort speed not as important: parallel A* units
  - Buffer limited to 32 entries, determined by analysis of data set

Parallel A* units

- Each unit contains
  - Distance calculation block
  - bubble sort hardware block to sort the queue
- Customized cache in memory controller
  - Cache from DRAM into QDRII SRAM
  - LRU page replacement





Computation Directorate

Performance

Num Units	Area (slices)	Area (%)	Mult (of 232)	BRAM (of 232)
24	18165	76	48	112

Implementation	Time (sec)	Loads	Rate/sec	Speedup
CPU	35	2024294 edges	86376	1
XD-1 (DRAM)	0.68	778 blocks	2925280	50

- Compare Opteron only to Opteron + FPGA
- FPGA algorithm uses Opteron's DRAM to load SRAM cache
- Access to 2024294 edges, with 778 page loads

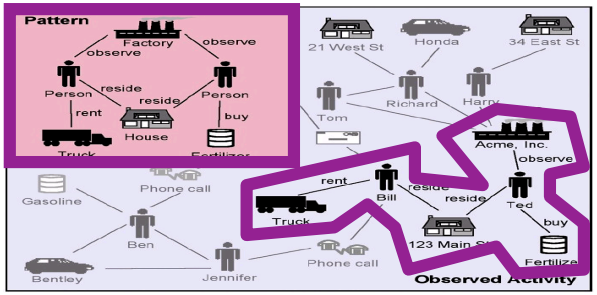
Discussion

- A **50X** speedup for a latency-driven random-access algorithm truly demonstrates a disruptive technology
- BUT …
- Hardware implementation in VHDL by experienced hardware/software designer
- Six months to build/debug hardware
- Needs considerable expertise with FPGA, CAD tools, board level architecture, system level architecture, software algorithms and their implication for hardware
- Needs coordination between software and hardware

Example: Path finding in semantic graphs

Semantic graphs are used to analyze relationships in large data sets coming from heterogeneous data sources

How is A related to B?
Is a certain activity pattern in the data?

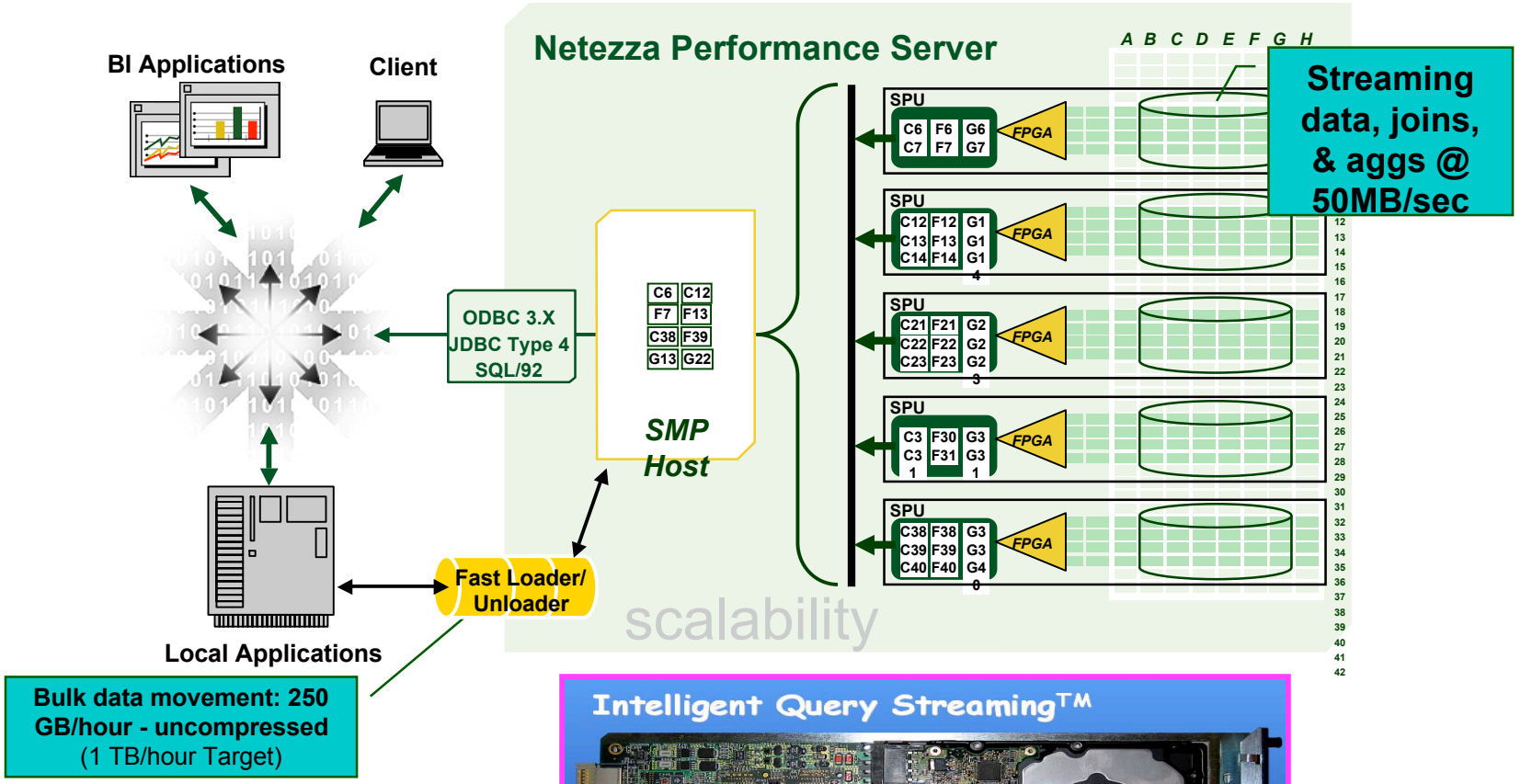


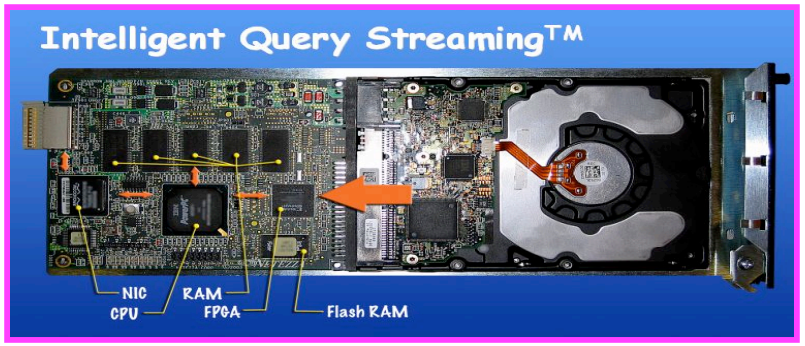
T. Coffman, S. Greenblatt, S. Marcus, *Graph-based technologies for intelligence analysis*, CACM, 2004

M. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev., 2003

We need to analyze graphs that are orders of magnitude larger than those processed using current technology, 10^{12} nodes, faster - in minutes instead of days

Database storage appliance





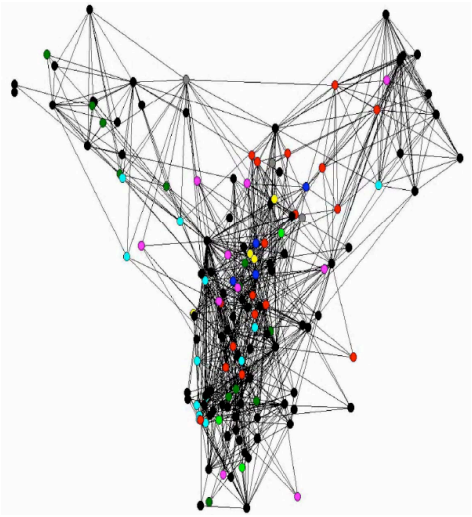
Methodology

- Represent graph as SQL table where each row represents an edge
- Pose the shortest path problem as SQL query
- Query is optimized on host
- Sub-queries dispatched to all the FPGA/PPC processors who read their part of the database and filter the table rows
- Rows matching the query are returned back to host
- User sees SQL interface only
- SC 2007

FPGA-accelerated storage server vs. BG/L

	IBM BG/L (2005)	Netezza 8650 (2006)
Graph Edges	30 billion	300 billion
Graph Description	Random **(academic problem)**	Scale-Free **("real world" problem)**
Processors	65, 538	648
Avg. Search Time	1.4 sec	218 sec
Level of Effort	6 months, 2000 lines of C code	2 week, 100 lines of SQL code

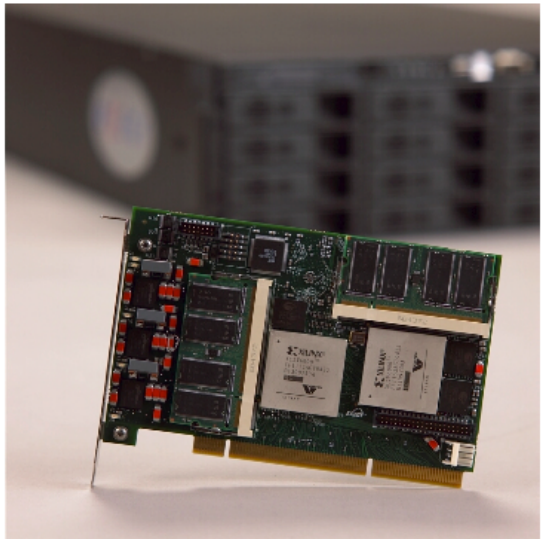
Bi-directional Breadth-First Graph Search Algorithm

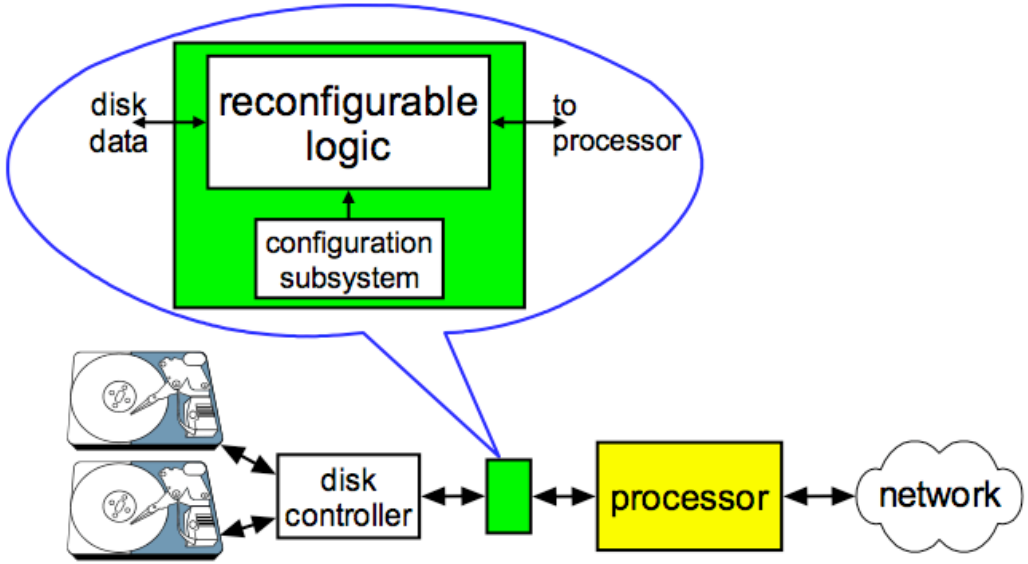


- **10X More Edges**
- **12X Productivity Improvement**
- **300 Billion Edge Problem Not Achievable on BG/L**

Computation Directorate

Exegy FPGA appliance





•Applications include financial analysis and text search.

•Flow data from disk through FPGA pipeline at 300MB/s - 600MB/s rate.

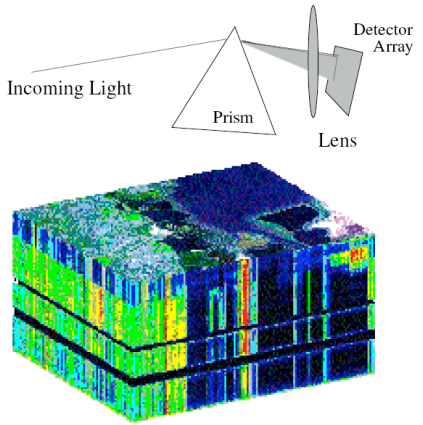
Well suited to streaming problems.

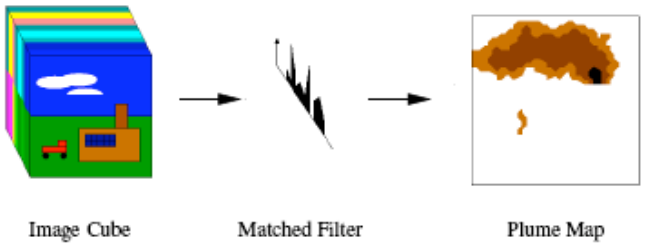
Discussion

- Pros
  - Appliance approach hides existence of FPGA from user
  - Familiar software-oriented interface
  - No need to do synthesis, place, and route
- Cons
  - **Cost**
  - Limited application classes
  - Closed source, not extensible

Comparison of various co-processors

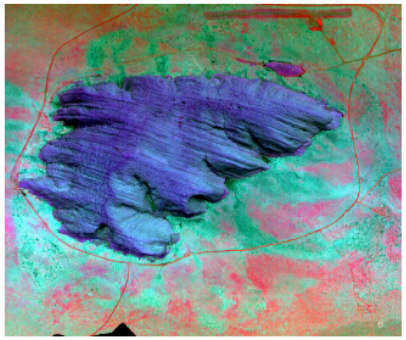
- Matched filter over hyperspectral imagery
  - Locate geographic, atmospheric features
  - Wide spectral content, divided into 100's of bands
  - Large data cubes (eg. 240 x 240 x 1000) collected in real time
- Compare Cell, FPGA, and GPU
- **Justin Tripp**, **Zach Baker** of LANL (FCCM, 2007)



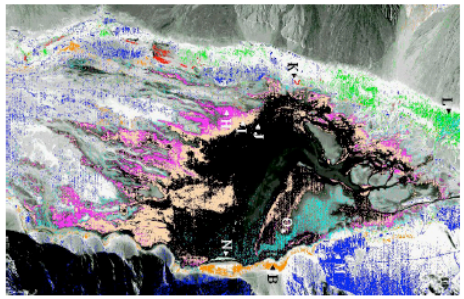


Hyperspectral imagery applications

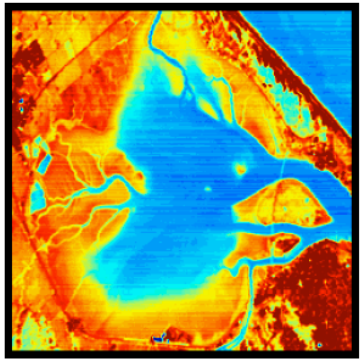
Geology



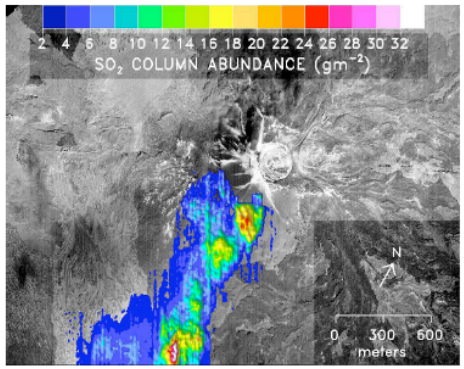
Mineral Detection



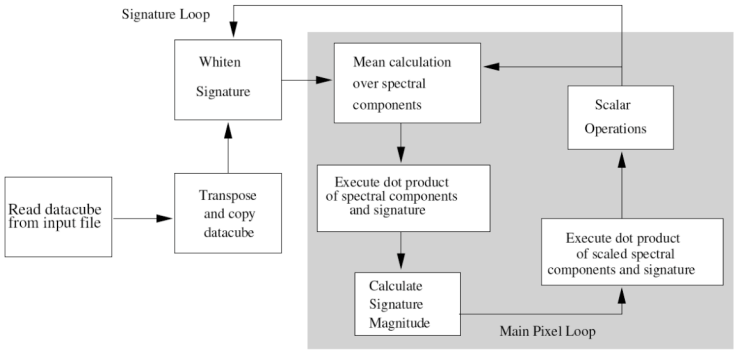
Wetland Studies



Atmospherics



Matched filter algorithm



Time break down for each task (seconds)			
Load File	4.32	Dot Product	.60
Copy Data	11.75	Signature Magnitude	.60
Whiten Signature	.15	Scaled Dot Product	.65
Mean Subtraction	1.30	Scalar Ops	.20

Processing of each signature takes 3.05 seconds on 3.2GHz Xeon.

Impl.	Read File	Trans-pose	Mean Calc.	Mean Sub.	Dot Product	Scalar Ops.
CPU	X	X	X	X	X	X
GPU			X	X	X	
CELL			X	X	X	X
FPGA			X	X	X	

Implementation	Speedup over CPU	Time per Signature	Speedup per $k	Speedup per kWatt
FPGA (V2Pro50)	3.91	0.78 sec	.39	11.2
GPU (nv 7900)	3.1	1.0 sec	1.24	8.86
Cell (3GHz)	8.0	0.38 sec	1.0	25.4
CPU (3GHz Xeon)	1	3.05 sec	.5	3.33

Summary

- FPGA co-processors have shown orders of magnitude speed up on certain problems.
- FPGAs are not a panacea and are not best used as general purpose processors.
- Adapting algorithms to FPGA continues to be a challenge, though “appliance” approach is useful for specific application classes.
- Cost of FPGAs and associated algorithm development tools is very high.
- Multi-core processors are competitive to FPGAs, especially for floating point dominated kernels.