Scientific Computing with FPGAs
The Reconfigurable Computing Cluster Project

Ron Sass
http://www.rcs.uncc.edu/~rsass
University of North Carolina at Charlotte
September 8, 2009
Fate of the Beowulf

computing for the masses
... slightly different audience than the National Labs and largest Universities
Computational Science

- in addition to experimental and theoretical branches of science, computational science is now crucial to nearly every discipline
in addition to experimental and theoretical branches of science, computational science is now crucial to nearly every discipline.

however...
in addition to experimental and theoretical branches of science, **computational science** is now crucial to nearly every discipline

however...

- science is limited by the power of the instrument
- the *rate-of-discovery* is tightly coupled to the *rate-of-computation*
Beowulf-style parallel computing couples
- Commodity Off-The-Shelf (COTS) hardware
- Open Source software (GNU, Linux, MPI, etc.)

with help from Moore’s Law, this approach has come to dominate the high-end computing; consider TOP500 list
Beowulf-style parallel computing couples
  - Commodity Off-The-Shelf (COTS) hardware
  - Open Source software (GNU, Linux, MPI, etc.)
with help from Moore’s Law, this approach has come to dominate the high-end computing; consider TOP500 list
  - 80% are “clusters”
  - 57% use Gigabit Ethernet
  - 85% use Linux
Scaling Beowulf

- to improve the rate-of-computation
  - use faster nodes
  - buy more nodes
Scaling Beowulf

- to improve the rate-of-computation
  - use faster nodes
  - buy more nodes

- however, to make it on the list ...
  - 960 nodes (six 42U racks) Num. 474
  - 1200 processors (10 racks) Num. 486
Scaling Beowulf

- to improve the rate-of-computation
  - use faster nodes
  - buy more nodes

- however, to make it on the list ...
  - 960 nodes (six 42U racks) Num. 474
  - 1200 processors (10 racks) Num. 486

- multi-core/many-core to the rescue! except:

  \[
  \begin{array}{ccccccc}
  M & M & M & M & \cdots & M \\
  \hline
  P & P & P & P & \cdots & P \\
  \end{array}
  \]

(same with disk I/O bandwidth)
Fate of the Beowulf

Moore’s Law will march on, but the technology trends are not positive

- memory bandwidth, latency are not improving
  - it is a packaging problem (no more pins)
  - 70ns for 64 Mb SDRAM in 1994;
    30-60ns for 2Gb SDRAM in 2007
Fate of the Beowulf

Moore’s Law will march on, but the technology trends are not positive

- memory bandwidth, latency are not improving
  - it is a packaging problem (no more pins)
  - 70ns for 64 Mb SDRAM in 1994;
    30-60ns for 2Gb SDRAM in 2007

- power density
  - not every scientist has 5 MW in his/her machine room
  - 400-500W power supplies are common
  - 100W/sqft (no floor/ceiling air)
Fate of the Beowulf

Moore’s Law will march on, but the technology trends are not positive

- memory bandwidth, latency are not improving
  - it is a packaging problem (no more pins)
  - 70ns for 64 Mb SDRAM in 1994; 30-60ns for 2Gb SDRAM in 2007

- power density
  - not every scientist has 5 MW in his/her machine room
  - 400-500W power supplies are common
  - 100W/sqft (no floor/ceiling air)

- size/mass
  - building infrastructure
  - physical distance between switch and node
Concrete Evidence?

![Graph showing performance in MFLOPS over years with linear regression lines and equations]

- \( \text{avg}(1:10) \quad y = 0.13195x + 1.2371 \)
- \( \text{avg}(491:500) \quad y = 0.14119x - 0.48305 \)
Concrete Evidence?

Performance in MFLOPS vs Years

- avg(1: 10) $y = 0.13195x + 1.2371$
- avg(96: 105) $y = 0.12923x + 0.21129$
- avg(246:255) $y = 0.13539x - 0.18182$
- avg(491:500) $y = 0.14119x - 0.48305$
Concrete Evidence?
Outline
Hypothesis

**Hypothesis:** A network of Platform FPGA devices will scale to a PetaFLOP and be more cost-effective than Beowulf-style Commodity Clusters.

this is controversial...

- FPGAs consume more power than ASICs or custom ICs
- typically $10 \times$ slower clock frequency, $4 \times$ more area
- communication costs torpedo many applications
- programming model (850,000 programmers graduate each year versus 80,000 hardware engineers!)
Spirit: Reconfigurable Computing Cluster
to answer these questions, *Spirit*, a small-scale model was fabricated
to answer these questions, *Spirit*, a small-scale model was fabricated

- 64 commodity developer boards (Xilinx ML-410)
to answer these questions, *Spirit*, a small-scale model was fabricated

- 64 commodity developer boards (Xilinx ML-410)
- custom network board that with low-cost SATA connectors/cables
to answer these questions, *Spirit*, a small-scale model was fabricated:

- 64 commodity developer boards (Xilinx ML-410)
- custom network board that with low-cost SATA connectors/cables
- developed system software for remote access (power on/off, JTAG, etc.)
Organization
IBM CoreConnect (SoC) – Platform FPGA

![Diagram showing the IBM CoreConnect (SoC) Platform FPGA and its components including PHY, DDR2 Controller, NETWORK INTERFACE, PPC405, DISK CONTROLLER, SERIAL PORT, SYSTEM BUS, PERIPHERAL BUS, and BRIDGE. The diagram also includes an On-Chip Router with Port connections and Switch Ctrl. The diagram uses symbols to indicate 'hard' cores (cannot be changed) and 'soft' cores (flexible, can be changed).]
many processes in nature exhibit an exponential decay property

- molecular forces
- concentration gradients of protein in a gel
- other simulations
many processes in nature exhibit an exponential decay property
- molecular forces
- concentration gradients of protein in a gel
- other simulations

hence, computational scientists make extensive use of \( e^{-x} \) in computer simulations
many processes in nature exhibit an exponential decay property
  - molecular forces
  - concentration gradients of protein in a gel
  - other simulations

hence, computational scientists make extensive use of $e^{-x}$ in computer simulations

our goal: FPGA implementation of double-precision, IEEE 754 standard $e^{-x}$
(in FORTRAN this DEXP, hence the name)
Core Design

nd - New data (enable) signal
dn - Done
• about 29 $\mu$s for FPGA, 66 $\mu$s on modern processor
• $< 20 \text{ W}$ for FPGA system versus $\approx 350 \text{ W}$ (not measured)
BLAST

- BLAST is a bioinformatics application used by thousands (hundred-thousands?) biologists every day
  - hardware can be used to speed it up
  - but it quickly becomes I/O bound problem (primary and secondary storage)
BLAST is a bioinformatics application used by thousands (hundred-thousands?) biologists every day
- hardware can be used to speed it up
- but it quickly becomes I/O bound problem (primary and secondary storage)

our goal: scalable FPGA implementation scan (previously NtWordFinder) and secondary storage subsystem to scale I/O bandwidth
BLAST Performance
HWFS: Migrating Filesystem Operations into Logic

Application

Operating System

File System

Device Driver

Disk Drive Controller

Hard Disk

Software

Hardware

Sector

Track

Application

Operating System

File System

Device Driver

Disk Drive Controller

Hard Disk

Software

Hardware

Sector

Track
Efficiency

The graph shows the efficiency of latency for different block sizes (64B, 256B, 512B) for both read and write operations. The y-axis represents the efficiency ratio, and the x-axis shows the block size ranging from 1KB to 100KB.

- **Ideal** represents the theoretical best-case scenario.
- **Read Latency** and **Write Latency** graphs compare the actual latency against the ideal latency for each block size.

The data indicates that efficiency increases with larger block sizes, especially for read operations.
BLAS — Basic Linear Algebra Subroutines

• BLAS (and its descendents) is a library often used by scientists for dense matrix computations
  • matrix-matrix multiplication
  • matrix-vector multiplication
  • inner product

Our goal: show peak floating-point performance of our devices; benchmark with High-Performance Linpack

Experiments include single node tests and an MPI application; largest matrix size: $14336 \times 14336$
BLAS — Basic Linear Algebra Subroutines

BLAS (and its descendents) is a library often used by scientists for dense matrix computations

- matrix-matrix multiplication
- matrix-vector multiplication
- inner product

Our goal: show peak floating-point performance of our devices; benchmark with High-Performance Linpack

Experiments include single node tests and an MPI application; largest matrix size: $14336 \times 14336$
MAcc — Multiply/Accumulate Array
## Single Node MFLOPS for Various Matrix Sizes

**Theoretical Peak:** 6.4 GFLOPs

<table>
<thead>
<tr>
<th>Size</th>
<th>MFLOPS</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>$16 \times 16$</td>
<td>839.04</td>
<td>1</td>
</tr>
<tr>
<td>$32 \times 32$</td>
<td>1431.02</td>
<td>1.71</td>
</tr>
<tr>
<td>$64 \times 64$</td>
<td>2102.38</td>
<td>2.51</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>2726.64</td>
<td>3.25</td>
</tr>
<tr>
<td>$256 \times 256$</td>
<td>3197.60</td>
<td>3.81</td>
</tr>
<tr>
<td>$512 \times 512$</td>
<td>3498.54</td>
<td>4.17</td>
</tr>
<tr>
<td>$1024 \times 1024$</td>
<td>3670.87</td>
<td>4.38</td>
</tr>
<tr>
<td>$2048 \times 2048$</td>
<td>3763.60</td>
<td>4.49</td>
</tr>
<tr>
<td>$4096 \times 4096$</td>
<td>3811.77</td>
<td>4.54</td>
</tr>
</tbody>
</table>

**Measured Power:** 90 MFLOPS/Watt
Network Performance

- theoretical
  - 4 Gbps (error-free) per direction per link (lanes)
  - 8 links per FPGA — so 64 Gbps in/out of node
  - after 8B/10B encoding, 3.2 Gbps
  - $120 (real cost) per NIC (switch is free)

- measured hardware core-to-core
  - about 95% of theoretical bandwidth with 16 KB messages
  - 0.8 $\mu$s chip-to-chip latency
  - 0.08 $\mu$s on-chip latency (just the crossbar)

- measured (Linux) software process-to-process
  - about 56% of theoretical for 16 KB message
  - approaches 2.4–2.5 Gbps (80% of theoretical) for 1 MB message
  - 100 $\mu$s chip-to-chip latency
MPI Collective Communications

- migrate latency sensitive operations
- reduce interrupts and traversing OS/library interfaces
- hardware cores directly connected to network
Barrier: Point-to-Point Communication in hardware

4-ary 2-cube torus (subcube of whole cluster); (a) Full-Radix tree, (b) Linear tree, (c) Binary tree, (d) Star tree
software MPI_Barrier GigE versus hardware barrier core on Full-Radix (best) and Linear (worst) topology

hardware barrier core on Full-Radix, Binary tree and Star topology
Spirit Summary

- work-in-progress... but cards are falling right
- absolute comparisons are difficult right now — Spirit is a very small scale model
- power numbers are excellent
  - FPGA with MAcc array: 90 MFLOPS/Watt
  - Desktop CPU (Opteron): 27 MFLOPS/Watt
- network numbers are solid
- will it work for a range of applications???
Thanks to the People that Really Did the Work

- Andy Schmidt
- Will Kritikos
- Robin P.
- Shan Yuan Gao
- Ashwin Mendon
- Yamuna Rajasekhar
- Sidd Datta
Overview

Resilient

- Webster: recovering readily from adversity
- Presently: fault tolerance
- Longview: performance degradation as well as fault mitigation
  - OS noise / OS jitter
  - timing due to hardware RAS (hard drives, down clocking)
  - result of checkpointing/restart software

My aim with the remainder of this talk is spur questions: Where are there points of collaboration?
Trends Going Forward

- 65nm → 45nm → ⋯ 32nm
  - less tolerant, smaller target
  - more susceptible
- longer running simulations
- higher component count machines
- traditional techniques (TMR) not feasible
- commodity components will incorporate Reliability-Available-Serviceability (RAS)

Working Assumption: Every execution will have exceptional events.
Essential Question

*How to prepare for an unpredictable, unreliable future with today’s technology?*

- cycle-accurate simulators of parallel systems: impossible
- behavioral simulations lack fidelity
- real systems today (that exhibit exceptions) are rare and precious
Enter An FPGA Cluster

fully operational MPI solution with (re)programmable hardware offers interesting (inexpensive) possibilities

- targeted, reproducible fault injection
- variable grain disturbance (down to cycle-level)
- custom (exploratory) performance monitoring
- analytics to suggest an exceptional situation has (or will) occur

all (nearly) “Heisen-bug free” —

- implemented in hardware
- operating in parallel with functioning system
What is Needed?

Administratively, a testbed with...

- hardware fault (performance) injection based on a probability distribution function (or trace?)
- fault reproducibility (same physical bits flipped)
- behavior reproducibility (app fails at the same place)
- resilience middleware
- plan for credible experiments/exploration
  - What needs to be observed?
  - How to aggregate data into actionable decision?
Probes & Dials

- a set of adjustable, composable, interacting hardware components

- Probes
  - back-end components that sense specific events (interrupts, messages, bus activity)
  - front-end components that aggregate data (interrupts per second, sliding windows, trigger on extraordinary situation)

- Dials
  - perturb running system
  - adjustable at run-time (on/off, frequency, duration, etc.)
Back-End Probes

- PowerPC Trace Port (branches, system calls, etc.)
- PowerPC interrupts
- System bus activity
- Network packets (source, destination, size)
- Temperature
- Disk activity
Front-End Probes

- convert counts to rates (messages per second)
- collect sliding window of data
- convolution (e.g., edge detection)
- artificial neural network (trained to detect “healthy” node)
Dials

Performance
- system bus “cycle stealer”
- network bandwidth stealer
- increase DRAM latency
- false interrupt generator
- down clocking components

Fault
- DRAM bit flipper
- corrupt floating-point results
- corrupt data packets (after CRC)
Current Probes and Dials

- PLB (system) bus “cycle stealer”
  - master performs unnecessary reads or writes to a null slave
  - adjust frequency of interruption
  - adjust duration (repeated transaction) of interruption
- PowerPC Trace port — collects 64MB of data
- SDRAM bit flipper
  - selects a random word of off-chip memory and flips one random bit
  - frequency is adjustable
  - PRNG seed can be set at run-time
- Count Interrupts and Convert to rate — period is adjustable
- Sliding Window plus Edge Detector
Experiment with Cycle Stealer

- attach cycle stealer to degrade the performance of one node
- adds *contention* — does not always prevent processor
- run NAS Parallel Benchmark (IS) on one node
- Three questions
  - What are reasonable ranges for frequency and duration before node observes performance degradation?
  - What are the effects on a 16-node system if one node is degraded?
  - Will the *system* observe the performance degradation before the node?
Experimental Set-Up: One Nodes

- JTAG
- PowerPC
- Event Probe
- BRAM Controller
- DDR2 Controller
- CF Controller
- DDR Controller
- UART
- IIC
- MGT (wrapped with mgt_protector)
- IIC
- JTAG
- PLB Master Core
- Disturbance Unit
- FREE RUNNING PRNG
- XOR
- MEAN DELAY
- PRNG Mask
- COUNTER
- CONTROL
- BUS MASTER
- REG
- LENGTH
- CONTROL
- DDR2 Controller
- DDR Controller
- BRAM Controller
Experimental Set-Up: Sixteen Nodes
Increase in Execution Time while Varying Frequency

one node

sixteen nodes

percent increase in execution time of NAS Parallel Benchmark IS as $\mu_{freq}$ increases; $\mu_{dur}$ is fixed at 0.8$\mu$s, 328$\mu$s, and 84ms
Increase in Execution Time while Varying Duration

percent increase in execution time of NAS Parallel Benchmark IS as $\mu_{\text{dur}}$ increases; (a) one node and (b) 16 nodes
Interpretation

- PLB system bus (plus PowerPC cache) is very resilient!
  - bus reads are lower priority and no slow down was observed (results were achieved with writes)
  - even with near saturation (approximately 9:1) bus prevented starvation (only 20% slow down)
- duration has larger impact than frequency
- system was impacted by a single node failing
- in some cases, system was impacted sooner than a single node
Other Probes and Dials

- PowerPC Trace Port: we are gathering the data but not sure how to decode it — documentation is very thin
- Off-Chip RAM bit flipper
  - works... increasing rate of error generally decreases time to kernel panic/oops
  - even though the bit flipping is perfectly reproducible, Linux concurrency is not (so multiple runs and statistics are required)
Resiliency Work

- Rahul Sharma
- Nathan DeBardeleben
Discussion

I will be with Nathan the rest of day...
Why FPGAs?
1. Power

- every transistor (in the application-specific FPGA design) is contributing to the solution
  - minimizes static power
  - dynamic power is used for *useful* computation
1. Power

- every transistor (in the application-specific FPGA design) is contributing to the solution
  - minimizes static power
  - dynamic power is used for *useful* computation
- slower clock rates: a design win
2. System Integration

- highly-integrated systems:
  
  single Platform FPGA can be configured with processors, system bus, peripherals, network interface, disk controllers — all running Mainline Linux Kernel

- fewer discrete components:
  
  - lower power
  - size advantages
  - fewer points of failure
2. System Integration

- highly-integrated systems:
  
  single Platform FPGA can be configured with processors, system bus, peripherals, network interface, disk controllers — all running Mainline Linux Kernel

- fewer discrete components:
  
  - lower power
  - size advantages
  - fewer points of failure

- single memory hierarchy
2. System Integration

- highly-integrated systems:
  - single Platform FPGA can be configured with processors, system bus, peripherals, network interface, disk controllers — all running Mainline Linux Kernel
- fewer discrete components:
  - lower power
  - size advantages
  - fewer points of failure
- single memory hierarchy
- ability to use cheap, high-speed, custom networking
3. Resources Are Fungible

- we start with a super simple, bare bones design (processor, memory, Ethernet, serial console)
3. Resources Are Fungible

- we start with a super simple, bare bones design (processor, memory, Ethernet, serial console)
- depending on the application or domain, add special-purpose cores:
  - \( \text{dexp}(-x) \) — exponential decay is common in many computer simulations of natural phenomenae
  - \( \text{MACc} \) — 16 × 16 array of floating-point units for BLAS
  - \( \text{scan} \) — computationally intensive part of BLAST algorithm
  - On-Chip/Off-Chip Network \( \text{AIREN} \) — core-to-core communication DMA access to 64 Gbps custom network

PNN, FFT, Convolution, Barrier/Collectives, HWFS, Integer Sort
4. On-Chip Communication

- FPGAs already have a tested, high-bandwidth Network-On-Chip
- configurability allows for novel operations
  - computation-in-the-network
  - disk/network integration (think: multi-disk filesystem that spans cluster)