Supercomputing in a Box

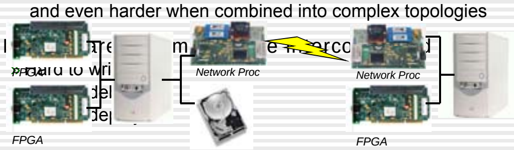
Roger D. Chamberlain

Dept. of Computer Science and Engineering
Washington University in St. Louis

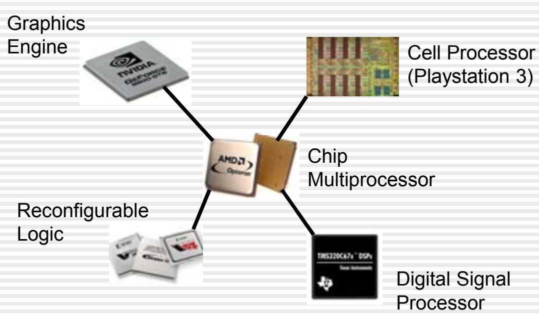
This work is supported by the National Science Foundation,
the National Institutes of Health, and Exegy, Inc.

Problem domain

- High-performance streaming applications
  - » Large streams of high-throughput data
    - – Networking and communications
    - – Scientific computing (offline AND online)
    - – Media creation and playback
    - – Data mining (e.g., bioinformatics, security)
- Special-purpose devices (FPGAs, CMPs, NPs, GPUs, DSPs)
  - » Often *much* faster (10-100x not uncommon)
  - » These are hard to program by themselves… and even harder when combined into complex topologies



Diverse Architectures



Chip Multiprocessors

Gaming Chips, Networking Chips

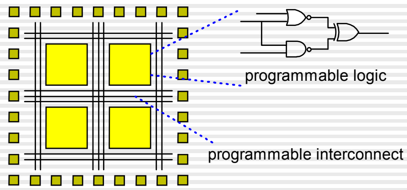
Examples include:
- Intel IXP
- ClearSpeed CSX600
- Tensilica Xtensa
- IBM Cell processor

Graphics Processors

Today's results are measured using an Nvidia GTX 260

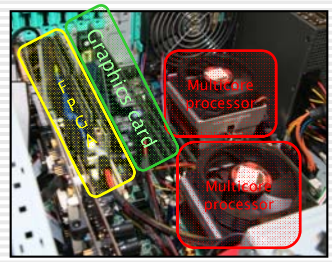
We're now using a Tesla C1060

Reconfigurable Logic



- Field Programmable Gate Arrays (FPGAs) provide custom logic function capability
- Operate at hardware speeds
- Can be altered (reconfigured) in the field to meet specific application needs

Supercomputer in a Box



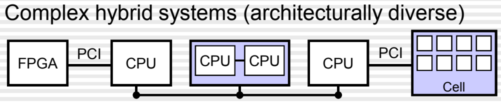
Auto-Pipe

Auto-Pipe is a development environment…

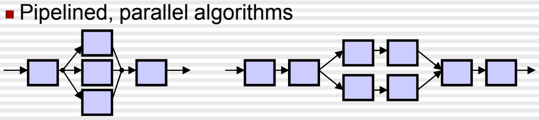
- a set of tools used to **create**, **test**, **build, and deploy** distributed applications

Auto-Pipe is made for…

- Complex hybrid systems (architecturally diverse)

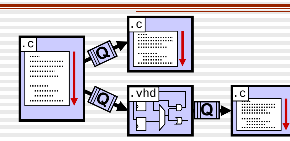


- Pipelined, parallel algorithms

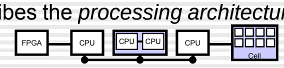


The Auto-Pipe Approach

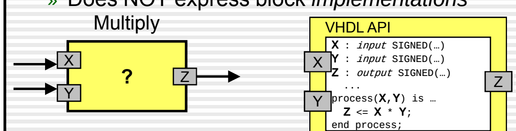
- The X language
  - » Dataflow semantics
  - » Queues



  - » Also describes the *processing architecture*



  - » Does NOT express block *implementations*

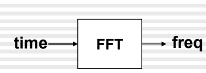


The Auto-Pipe Approach

- Auto-Pipe toolset:
  - » ***X-Com***: *X language compiler* [IPDPS 2006]
  - » **X-Sim**: A federated simulation environment [PADS 2007]
    - – Orchestrates simulators, emulators, and models to get various qualities of performance results
  - » **X-Dep**: Application deployment tool [SC 2007]
    - – Executes application on real hardware, including:
    - – Multicore [STMCS 2008] plus FPGA [NIM 2008]
- Development activities
  - » Functional simulation (test for correctness)
  - » Performance simulation
  - » Deployment to real hardware

Given (what this talk is not about)

Native implementation of modules for each compute resource




```
fft () {
  while(1) {
    rcve(array time[N]);
    perform_fft(time,freq);
    send(array freq[N]);
  }
}
```

```
entity fft is
    time: in std_logic_vector(...);
    freq: out std_logic_vector(...);
end fft;

architecture foo of fft is
begin
    ...
end foo;
```

Module design constraints

Module designs must conform to common interface specification

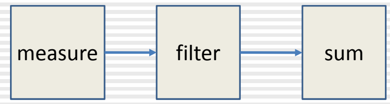
time → FFT → freq

```
fft () {
  while(1) {
    rcve(array time[N]);
    perform_fft(time,freq);
    send(array freq[N]);
  }
}
```

```
entity fft is
   time: in std_logic_vector(...);
   freq: out std_logic_vector(...);
end fft;

architecture foo of fft is
begin
   ...
end foo;
```


Specify Application Topology

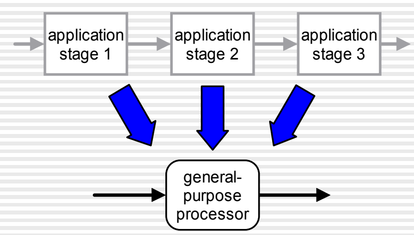


- Generate formal interconnect specification
- MPI-like semantics between modules

Design Tool

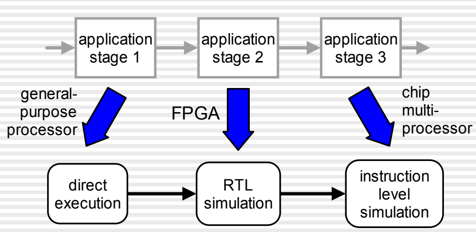
- Inputs:
  - » Individual module designs
  - » Interconnect specification
  - » Mapping of modules to compute resources
- Outputs:
  - » Software-only, pipelined "golden model"
  - » Instrumented functional RTL simulation
  - » Parameterizable discrete-event simulation model
  - » Deployable implementation

Software Golden Model



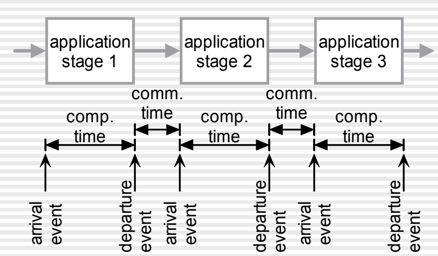
Also usable as a software pipelined implementation, if multiple general-purpose processors available.

Instrumented Functional Model



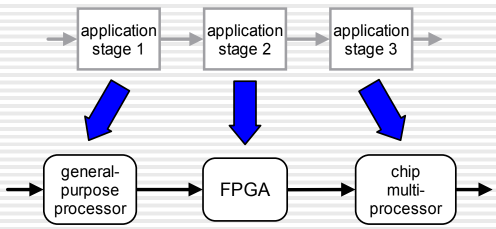
Collect performance data on:
- compute times
- communications volume

Discrete-Event Simulation Model



Parameterized from RTL-level simulation and/or deployed implementation.

Deploy on Actual Hardware



Features of Auto-Pipe

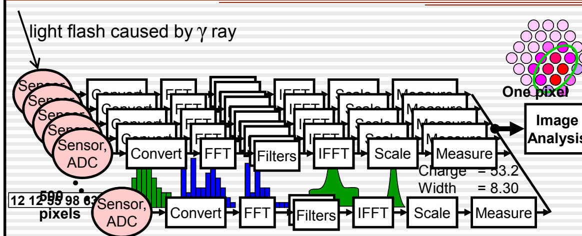
- Supports Architecturally Diverse Systems
  - » Targeting development on ALL types of compute and interconnect resources
- Uses Coordination Language
  - » Combine traditional implementation languages with the flexibility of high-level dataflow organization
- Deployable to REAL hardware
- Emphasis on Delivering Performance
  - » Early performance evaluation in design cycle
  - » Easing deployment enables greater exploration of design space
- Expandable
  - » New resource types and implementation languages

Example Application #1

Astrophysics

- Gamma-ray event parameterization
  - » *Active sources*: galactic nuclei, pulsars
  - » *Transient sources*: hypernovae, ...
- Lots of data: 20TB/year
  - » Want to process as fast as possible
  - » Process whole DB for rare events

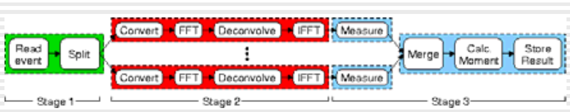
Gamma Ray Parameterization



- Gamma ray event parameterization
  - » Lots of signal processing
  - » Big! Wide *and* pipelined parallelism

Exploiting FPGAs

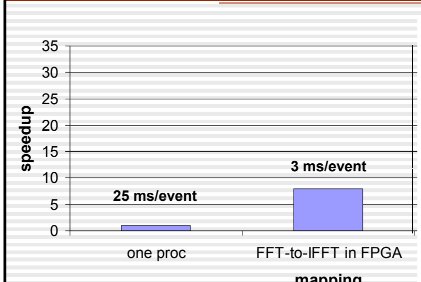
VERITAS signal processing



Mapping A, 1 proc

Mapping B, 2 procs and 1 FPGA: proc - FPGA - proc

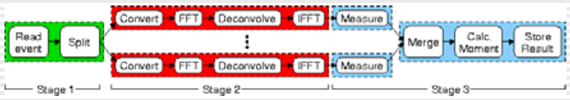
Pipeline Performance



Current bottleneck is delivery of data between CPUs and FPGA

Alternative Mapping

VERITAS signal processing

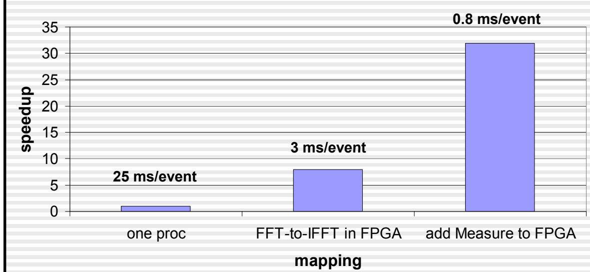


Mapping A, 1 proc

Mapping B, 2 procs and 1 FPGA: proc - FPGA - proc

Mapping C, 2 procs and FPGA: move Measure to FPGA

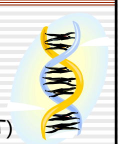
Performance of Complete Pipeline



This throughput is sufficient to support real-time operation in a single box (vs. ≥8 four-node boxes)

Example Application #2

Computational Biology

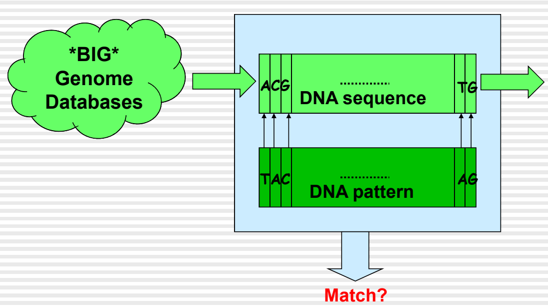


- Genome maps being expanded daily
  - » 80,000 genes, 3 billion base pairs (*A*,*C*,*G*,T)
- Look for matches
  - » Identify function
  - » Disease: understand, diagnose, detect, therapy
  - » Biofuels, warfare, toxic waste
  - » Understand evolution
  - » Forensics, organ donors, authentication
  - » More effective crops, disease resistance

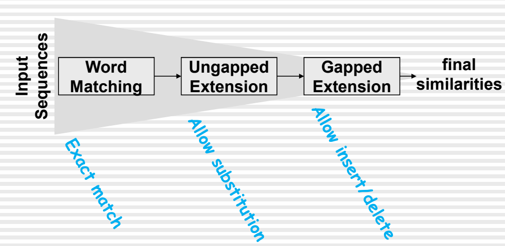
DNA String Matching

- Looking for *CACGTTAGT…TAGC*
- Interested in matches and near matches
- Search human genome, other gene oceans
  - » Need to search entire data sets

Bio Computation Problem



BLAST Biosequence Alignment



As we move down the pipeline:
- Data volume shrinks
- Compute requirements grow

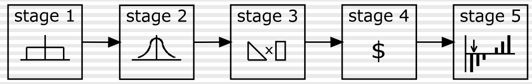
Performance Results

- DNA sequences
  - » Human chromosome 22 vs. entire mouse genome
  - » Baseline: executes in 3 hrs. 38 min. on 3 GHz Pentium D
  - » We execute it in 19 min., more than 10 times faster
- Protein sequences
  - » *Y. Pestis* (plague bacteria) vs. Genbank NR database
  - » Baseline: 40 hrs. 30 min.
  - » We execute it in 2 hrs. 42 min.
  - » Performance gain of > 15x

Example Application #3

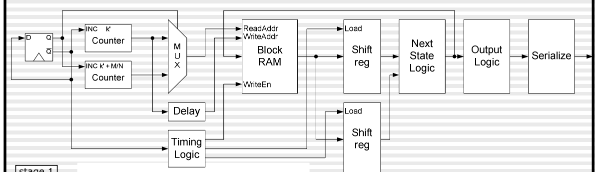
Computational Finance

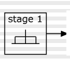
- Compute “value at risk” for a portfolio
  - » 1000 stocks
- Evaluate via Monte Carlo simulation
  - » Brownian motion random walk
  - » Execute 1 million trials and aggregate results



Stage 1: Mersenne Twister

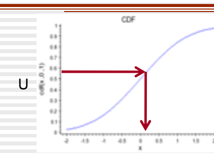
- Generates uniformly distributed pseudo random numbers with period $2^{19937} - 1$
- GPU version uses 1024 copies each with period $2^{607} - 1$
- FPGA version has following structure:



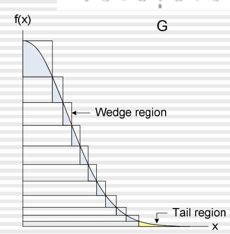


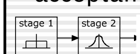
Stage 2: Transform into Gaussian

- CPU and GPU use piecewise polynomial approximation to inverse CDF (due to Acklam)



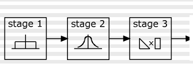
- FPGA uses ziggurat method, dividing area under PDF curve into equal area rectangles for frequent simple acceptance test





Stage 3: Correlate RNs

- Computation is matrix-vector multiply operation
- Vector is 1024 Gaussian RNs from stage 2
- Matrix is Cholesky factored correlation matrix
- Result is 1024 correlated Gaussian RNs
- Uses BLAS routines on CPU(s) and GPU
- Not yet implemented on FPGA (work in progress)



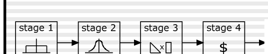
Stage 4: Random Walks

Use traditional Black-Scholes model:

$$S(T) = S(0)\exp\left((r - \sigma^2/2)T + \sigma\sqrt{T}z\right)$$

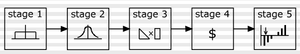
Models forward progress of price using underlying Brownian motion model

Same computation on CPU(s) and GPU

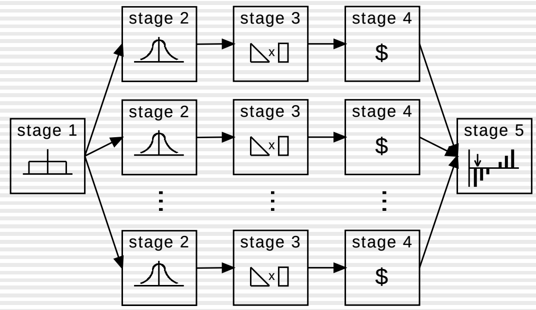


Stage 5: Compute Value at Risk

- Sort 1M P&L values from stage 4 trials
- 5^{th} percentile gives 95% VAR
- Computed on CPU (using qsort)

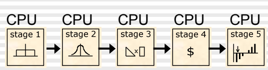


Parallel Pipelines

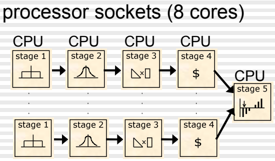


Experiments

- Baseline [C]
  - » 2.2 GHz AMD Opteron processor

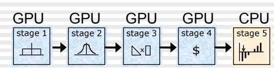


- Multiple CPUs [8C]
  - » 4 dual-core processor sockets (8 cores)

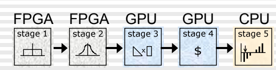


Experiments

- Graphics engine + processor [G-C]
  - » Stages 1 to 4 deployed on graphics engine

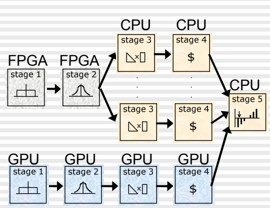


- FPGA + graphics engine + processor [F-G-C]
  - » Stages 1 and 2 deployed on Xilinx Virtex-4 LX80 FPGA



Experiments

- FPGA + graphics engine + processors [FG-CG-C]
  - » Stages 1 and 2 deployed on FPGA and graphics engine
  - » Stages 3 and 4 deployed on CPU and graphics engine



Performance

- Baseline [C]
  - » 450 thousand walks/second
  - » 37 minutes to execute 1 billion walks
- Multiple processors [8C]
  - » 1.1 million walks/second
  - » 15 minutes to execute 1 billion walks
  - » Disappointing speedup, issue is in BLAS acceleration
- Graphics engine + processor [G-C]
  - » 80 million walks/second $\Rightarrow$ 12 seconds for 1 billion walks
  - » Speedup of 178x
  - » $\approx$ 15% of GPU time is spent in stages 1 and 2

Performance

- FPGA + graphics engine + processor [F-G-C]
  - » 60 million walks/second ⇒ 17 seconds for 1 billion walks
  - » Slower ☹
  - » Due to no concurrent computation and communication on graphics engine (we are working with Nvidia to correct this)
- FPGA + graphics engine + processor [FG-CG-C]
  - » 81 million walks/second ⇒ 12 seconds for 1 billion walks
  - » Speedup of 180x
  - » Best overall performance
  - » FPGA lightly utilized (≈ 10%), performance will improve with stage 3 also deployed on FPGA
- First use of all three compute components

Performance Summary

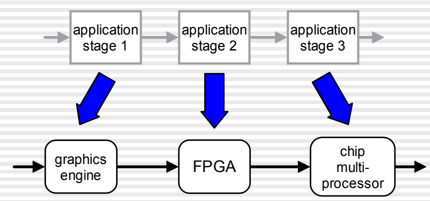
- Parallel BLAS routines have issues
  - » Simple nested for loop scales linearly with number of cores
  - » Possibly due to memory BW issues, but this is unconfirmed
- Expected improvement due to FPGA not realized
  - » Due to blocking communication/computation on GPU
- Very strong performance improvement associated with graphics processor
  - » Vector-matrix multiply is well suited to GPU
- Best performance achieved with complete set of computing resources
  - » Speedup of 180x over single processor core

Applications

- Operational
  - » Exact and approximate text search [ICCD 2004] [ISCA 2006]
  - » VERITAS gamma ray astronomy [STMCS 2008] [NIM 2008]
  - » Biosequence analysis for DNA [JVSP 2007] and protein [FCCM 2007] [TRETS 2008]
  - » Monte Carlo simulation for risk analysis [WHPCF 2008]
- In Progress
  - » More biosequence apps (HMMer [ICS 2007], PhyloNet)
  - » RNA folding [ASAP 2008]
  - » Options pricing in finance
  - » Monte Carlo solution to Laplace's eqn.

Summary

- Easing the deployment task is key



- Including
  - » loading executable images, and
  - » managing communication between stages
- Our development environment can do all this

Acknowledgements

- **Storage-based Supercomputing group**

  Michela Becchi, Justin Brown, Jim Buckley, Jeremy Buhler, Roger Chamberlain (PI), Patrick Crowley, Mark Franklin (PI), Saurabh Gayen, Michael Hall, Eric Tyson

- **High Performance Computational Biology group**

  Jeremy Buhler (PI), Roger Chamberlain, Mark Franklin, Brandon Harris, Arpith Jacob, Joseph Lancaster

- **Gamma ray application** VERITAS collaboration
- **National Science Foundation**
- **National Institutes of Health**
- **Exegy, Inc.**

More info is available at

http://sbs.wustl.edu
and
http://hpcb.wustl.edu

Questions?