Architecture Working Group

Larry Bergman (chair)
Loring Craymer (co-Chair)

October 25, 2007

Membership

Larry Bergman (JPL)
Maciej Brodowicz (LSU)
Nick Carter (Intel)
Loring Craymer (USC ISI)
John Gustafson (SUN)
Robert Lindsay (DOE)
Rich Murphy (Sandia)
Mike Niemier (Notre Dame)
Theodore Omtzigt (Stillwater SC)
Steve Poole (ORNL)
Karu Sandaralingam (Univ Wisconsin)
Steve Scott (Cray)
Karl-Heinz Winkler (LANL)

Top Architectural Challenges

- **Power**
  - Do we know how to design with other device technologies, eg reversible logic
- **Ease of programming vs efficiency**
  - Have to exploit locality without burdening programmer (>EF)
  - May have to introduce new programming/execution models (>EF)
- **Reliability/Resilience**
  - MTBF too short at EF scale for number of components used
  - Failure should be expected
  - Graceful degradation
  - Application continues to make progress in presence of failure
- How to help problems that don't have locality
  - E.g., graph problems, AI
  - Leave no code behind
- Getting the arithmetic right
  - Dealing with variable numerical precision needs
- Concurrency
- Size / Bulk
- Design / Build
  - First Principles Analysis and Design
  - Design time and manufacturing time
  - Selected / well chosen design points
- Supporting a global address space with greater than 2^64 bits
- Information security
- Cost
  - Commercial product alignment?
  - Use of consumer components?
- Number of interconnects & technology
- Packaging
  - cross cuts power, locality, size/bulk

Way Forward

- What do we need to do to realize this future
- What are the most promising courses of action
- How can we connect this future to our present

ExaFlops Target (c.f. S. Scott)

- 5-10 TF/chip and 100-200K chips

- Power
  - Assume 2014 FPU is ~10-20 pJ/flop, then 1 EF = 10-20MW
  - Memory bandwidth:
    - Eref/sec * 10% miss * 100 bits * 3pJ/bit = 30 MW for memory bandwidth

- Processor microarchitecture to exploit locality
  - Resolving the tension between programmability and efficiency
  - Need a new microarchitecture and execution model
    - Much lower control overhead relative to computation
    - Much more aggressive exploitation of locality (explicit control of data movement)
  - Co-design hardware with compiler/runtime software
  - Do *not* burden the programmer with this!
    - Need to be able to write in a portable HLL and compile to the microarchitecture

Potential Packaging Technologies

- 3D Stacking
  - extends CMOS and other technologies (Hafnium ??)
    - Processors
    - Memory (resilient, potential lower power)
    - Wafer-Scale opportunities
    - Feature size agnostic
  - Not just stacking (Full Crystal Growth, future)
  - Various intra-connects
    - Cu, Optical, C-Nano…
  - Hybrid
    - Mixed technologies on stack
      - CAM , FPGA, Optics (GAS, CMOS,…)
  - Potential for
    - FPGA
    - CAM
    - Flash
    - Quantum
- Better Memory Technologies
  - More efficient
    - MUCH Closer, MUCH Faster (3D paths)
- PIM or MIP or SPD's, SOC
  - Processors + Memory + Interconnect + Accelerator

Potential Packaging Technologies

- 3D Stacking
  - Stacked Optics
- LANL funded studies
  - Stacked CAM
  - Stacked FPGA
  - Stacked 40Gb/s packet analyzer
  - 8051 3D with IEEE + Crypto Functions
  - Stacked Cell processors
    - 128-256GB, 1TF processor
    - GAS layer (external optics)
    - 4-8TB/s on-stack memory BW
- Current potential vendors
  - Tezzaron
  - IBM
  - TSMC
  - Micron
  - Talked to several Graphics Vendors
- Potential enormous Application Impact
  - Minimizes latencies
  - Maximizes BW
  - Scatter / Gather easy to implement

Nominal Exascale Packaging Stats 2018 (probably sooner) Another View Based on Very Small Low Power Cores

3D MCM Module based on

- ~ 2000 cores (very low power), 2GF per core
- 15 layers
- 22 nm
- 16 TB / 4 TF per MCM stack
- P = 1-2 kW per MCM stack
- N = 250,000 modules (500M cores)
- P (system) = 250MW – 500MW (no power mgmt)
- P (system) = 25MW – 50MW (with rev logic, power mgmt)
- 8 MCM/blade
- 35,000 blades
- 1000 racks (30 x 30)

Complete system simulation project

- Government funded multi-diciplinary
  - Labs , Universities , Agencies , Commercial Partners
  - Multi year funding
- GPL, available to all.
- Allow for full system simulation
- Publish the API.
  - Allow for proprietary plug-ins
- Open Source / Open Development is non-negotiable
- Parallel design from conception
  - Must be able to be run on single node systems
  - Must be able to run on local as well “net” clusters (SETI)
- Design and produce POPs manual first
- Commercial agnostic (even if they pay)
- Element agnostic
  - Network technologies
  - Process technologies
  - Disk / Storage technologies…

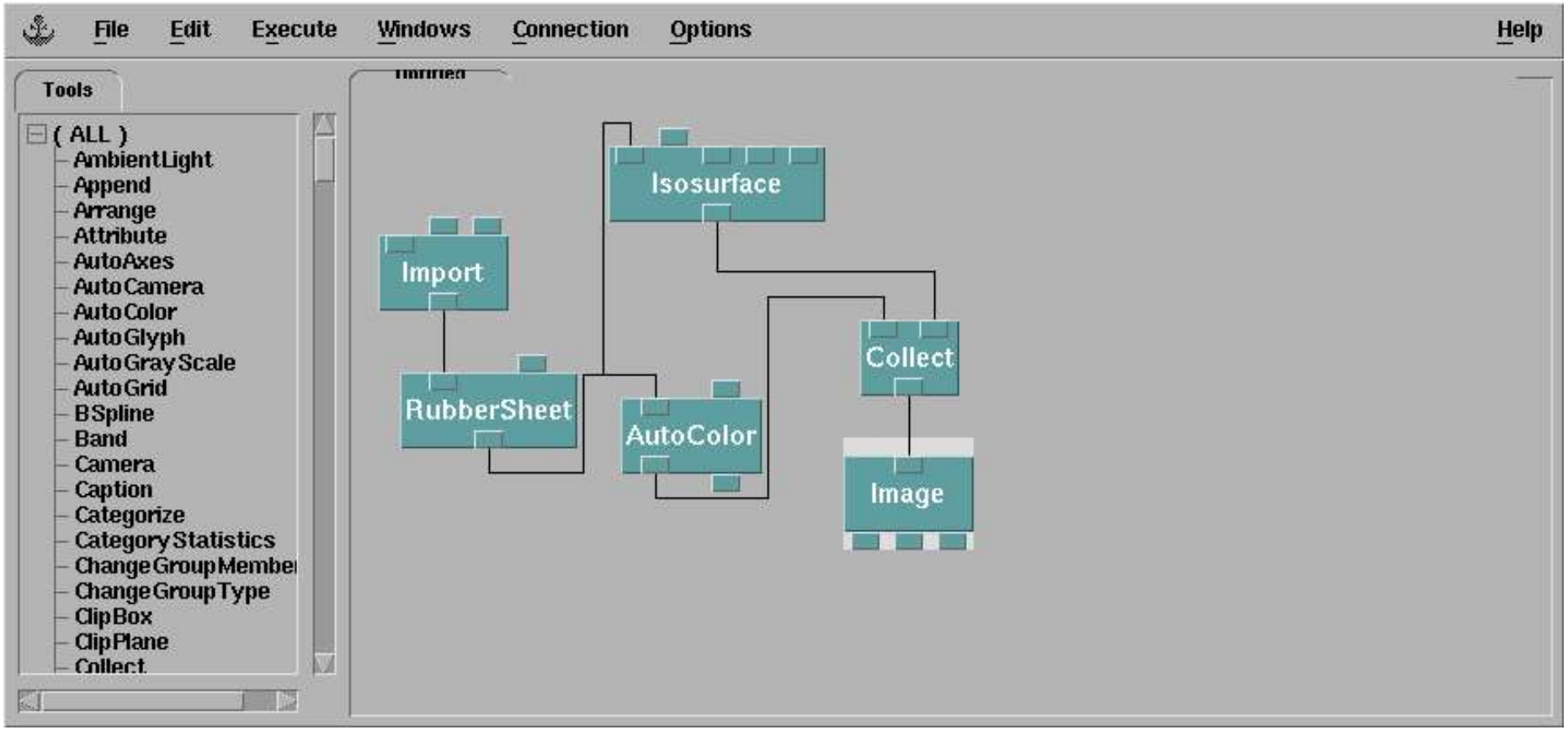
Complete system simulation project

- Full system component simulation
  - Function Accurate
  - Cycle accurate
  - Processor(s)
    - Vectors
    - Cache / no cache
    - Acceleration technoligies…
  - Intra-chip interconnect
    - Cu , Optical, C-nano…
  - Inter-chip interconnect
    - Cu, Optical, … ?
  - Memory hierarchies
  - Packaging agnostic
    - 2D, 2.5D,3D, ??
  - Process design rules agnostic
    - Should be able to work with all as well as potentially some not yet
  - Power aware modeling
  - Graph as well as graphical input

Complete system simulation project

- Full system component simulation
  - Must be able to replace SW with HW modules
    - FPGA
    - ASIC
    - Disk subsystem
    - Full systems
      - Ala RAMP
      - Ms. Clops (examples)
  - Replace SW model with physical prototype(s)
  - Must also model
    - Disk , I/O subsystem(s)
    - Flash, MRAM,…
- Numerous potential starting points
  - ASIM
  - NetSim
  - SST
  - BigSim
  - DiskSim (CMU)

Gui Interface (Example from OpenDX)

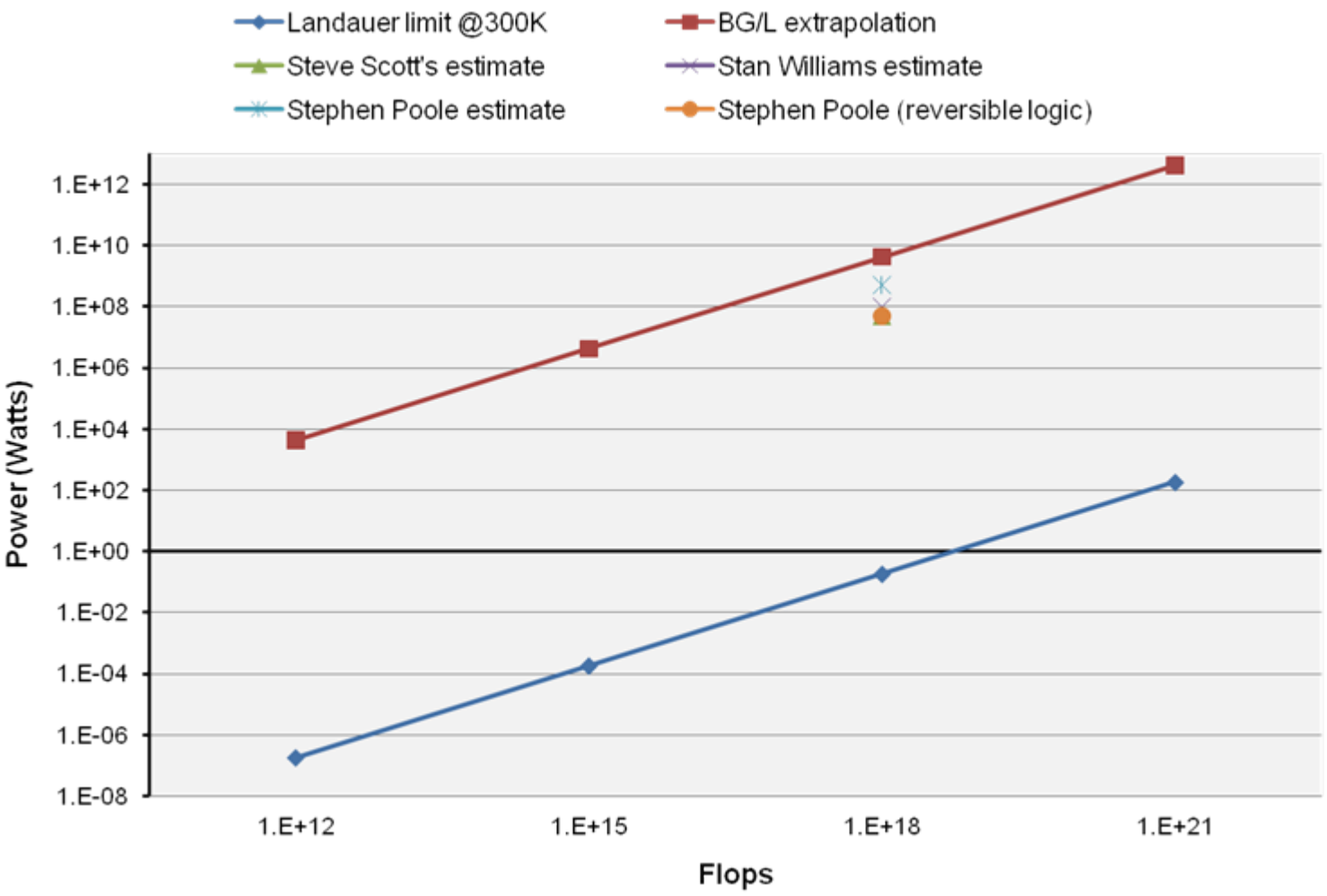


Links can be to either physical devices or logical devices
Modules can be virtual or physical
Define a standard API

Architecture Opportunities - Power

- Reducing Power problem
  - Reduce clock rate
  - Async logic
  - Operate on more data next to ALUs
  - Reversible logic
  - Variable precision (effects bw, storage, …)
  - Less speculation, less control (reduces overhead)
  - ….

Estimated Power Requirements



Resilience/Fault Tolerance

- Critical to offset aggregate error rates
- Implementation model
  - Hardware detection—minimal redundancy, minimal overhead
    - Need full Error Detection and Correction (EDAC) in microarchitecture to avoid full replication
    - EDAC on all communications
    - No single point of interrupt left uncovered
    - All failures detected and reported to system software
  - Checkpoint/rollback (or roll forward—avoid cascading rollback)
    - Non-volatile RAM for storage
    - May be staged to long-term storage for app scheduling
    - Could be implemented as shadowed RAM or virtual memory
  - Thread migration a feature of execution model
  - OS/runtime support for layered recovery
- Research needed
  - Algorithm-based fault tolerance (ABFT) for processor state machines, etc.
    - Compute invariants before/after and compare
    - Hardware design and experimentation
  - Fault coverage analysis integrated into hardware design tools
  - System-scale fault modeling tools
  - Programmable redundancy
  - Runtime Error Detection

Execution Model

- Based on SVP/Micro-threads research
  - Model vetted as part of EU AETHER effort
  - Some tools available now
- Uses Active-Object logical model
- Supports evolution across/from CMOS without programming changes
- Uses compilation support instead of hardware support to optimize locality
- Permits (but does not require) much simpler multi-core chip design.
- Support dynamic Aobject creation/migration

Execution Model – Active Objects

- Implemented in microthreads
- Creates Design/Programming Locality per instance that equates to data/operation locality per thread
- Architecture paradigm familiar to mpi programmers
- OO design concepts well understood
- Supports Aspects to handle hardware/system exceptions
- Application data expressed as active objects

On the Far Side

- Hybrid wet/dry strategies
- Biologically inspired digital systems design
  - MCM building blocks
  - Statically and dynamically reconfigurable
  - Possible stochastic methods for data integrity management
  - agents (for fault detection and recovery, programming redundancy, load balancing,

Recommendations

- Exascale (within decade)
  - Thread migration, locality exploitation, and micro architectures to support these features
  - Innovative resiliency methods required
  - Full system simulation an explore design space
  - Aggressive technology development:
    - 3D packaging
    - Optical interconnects at all levels (intrachip thru system)
    - Large amounts of non-volatile RAM close to the processor
- Zettascale (within 2 decades)
  - New device technology needed to reach last 10-100X