RAMP Blue: Implementation of a Manycore 1008 Processor System

RSSI 2008

July 8th, 2008

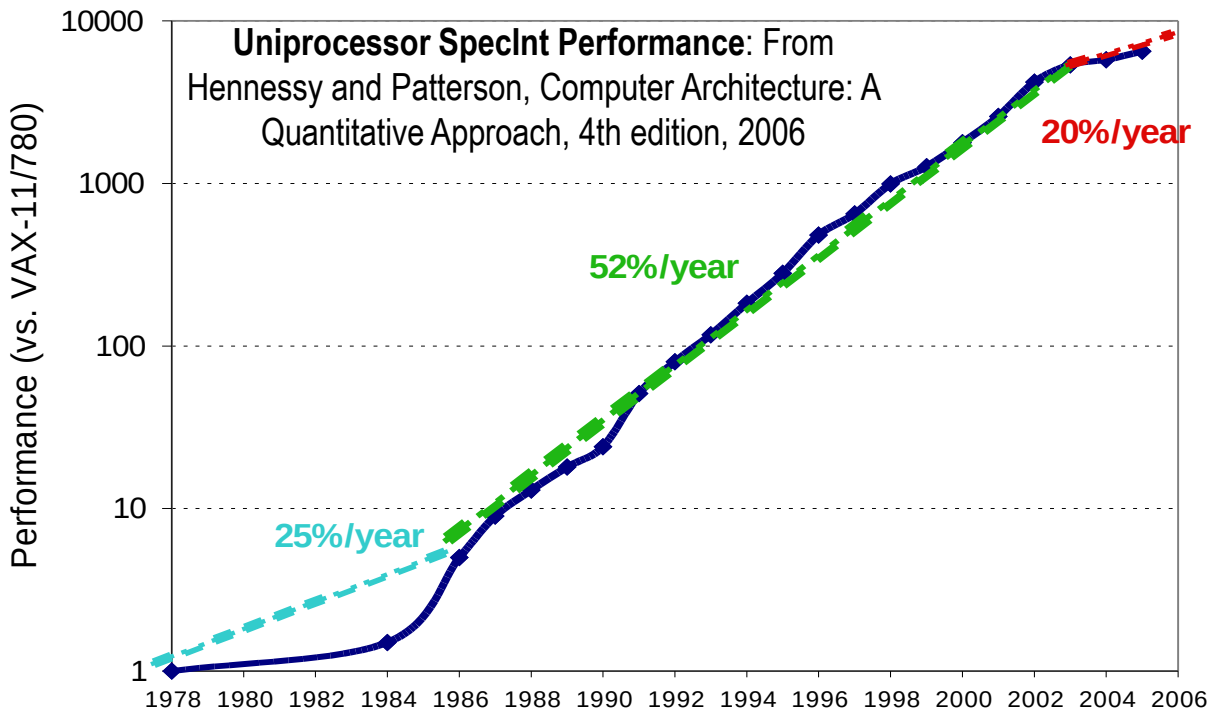
D. Burke, J. Wawrzynek, K. Asanović, A. Krasnov, A. Schultz, G. Gibeling, and P.-Y. Droz

© 2008 Regents University of California. All Rights Reserved

Talk Organization

- Overview, motivation, project organization
- RAMP Blue system details
- Version implementation
- Lessons learned and future directions
- Q&A

Motivation for RAMP



- Sea-change in chip design: multiple "cores"
  - More instances of simpler processors are more power efficient
  - Simple model for hardware scaling
- Moore's Law and many-cores: 2X CPUs per chip / ~ 2 years
- Multicores now (2, 4, 8, 16), Manycore soon (64, …)

RAMP Partnerships

- **Co-PIs:** Krste Asanovíc (UCB), Derek Chiou (UT Austin), Joel Emer (MIT/Intel), James Hoe (CMU), Christos Kozyrakis (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), David Patterson (Berkeley), and John Wawrzynek (Berkeley)
- HW development activity centered at Berkeley Wireless Research Center.
- Three year NSF grant (CNS-0551739) for staff (awarded 3/06).
- Hosted within GSRC Design Drivers Theme.
- Major continuing commitment from Xilinx.
- Collaboration with MSR (Chuck Thacker).
- Sun, IBM contributing processor designs, IBM faculty awards.

High-speed, high-confidence emulation is widely recognized as a necessary component of multiprocessor research and development. FPGA emulation is the only practical approach. RAMP is the only public project of this kind.

Research Accelerator for Multiple Processors

REVIEW

Invent an infrastructure to build cycle-accurate multi-core and many-core architecture emulators using FPGAs

- *Not FPGA computing*
- *Not a gate-level verification platform (Quickturn, Palladium)*

Why:

- **Rapid design space exploration** - A new set of architecture parameters can be tried each day leading to highly efficient (power, cost) designs.
- High confidence verification of design specification (conventional software simulators are either too slow or not trustworthy).
- An early platform for software development while waiting for machine to be built.

RAMP

Themes

- RAMP Blue is one of several initial design drivers for the **Research Accelerator for Multiple Processors** (RAMP) project.

- RAMP Blue is sibling to other RAMP design driver projects:
  - 1) **RAMP Red:** Port of existing transactional cache system to FPGA PowerPC cores
  - 2) **RAMP Blue:** Message passing distributed memory system using an existing FPGA optimized soft core
  - 3) **RAMP White:** Cache coherent multiprocessor system with full featured soft-core

Goal of building a class of large (500-1K processor core) distributed memory message passing many-core systems on a FPGAs

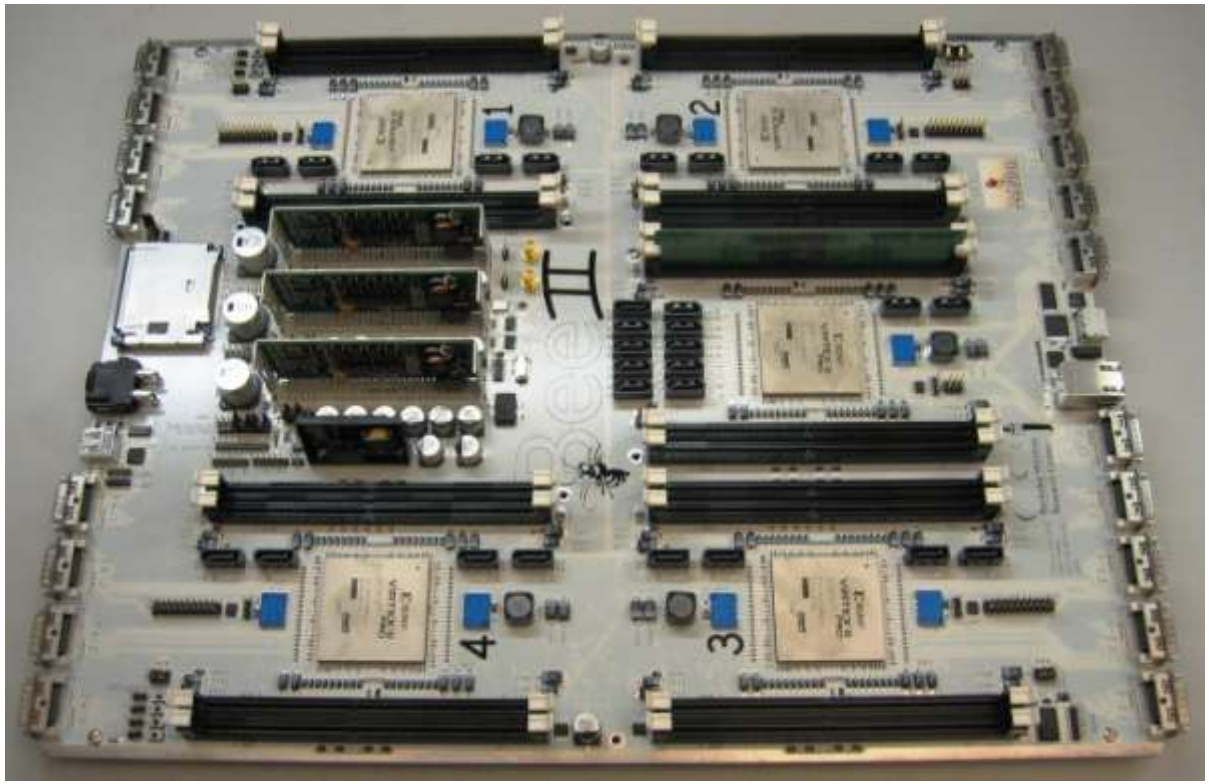
RAMP Blue Objectives

- Primary objective is to experiment and learn lessons on building large scale **manycore architectures on FPGA platforms**
  - Issues of FPGA implementation of processor cores
  - NOC implementation and emulation
  - Physical (power, cooling, packaging) of large FPGA arrays
  - Set directions for parameterization and instrumentation of manycore architectures
- **Proof of concept for RAMP** - *Technical imperative is to fit as many cores as possible in the system*.
- Starting point for **research in distributed memory** (old style "super computers") **and cluster style architectures** ("internet in a box")
- Develop and provide **gateware blocks** for other projects.
- **Application driven**: run off-the-shelf, message passing, scientific codes and benchmarks

RAMP Blue Hardware

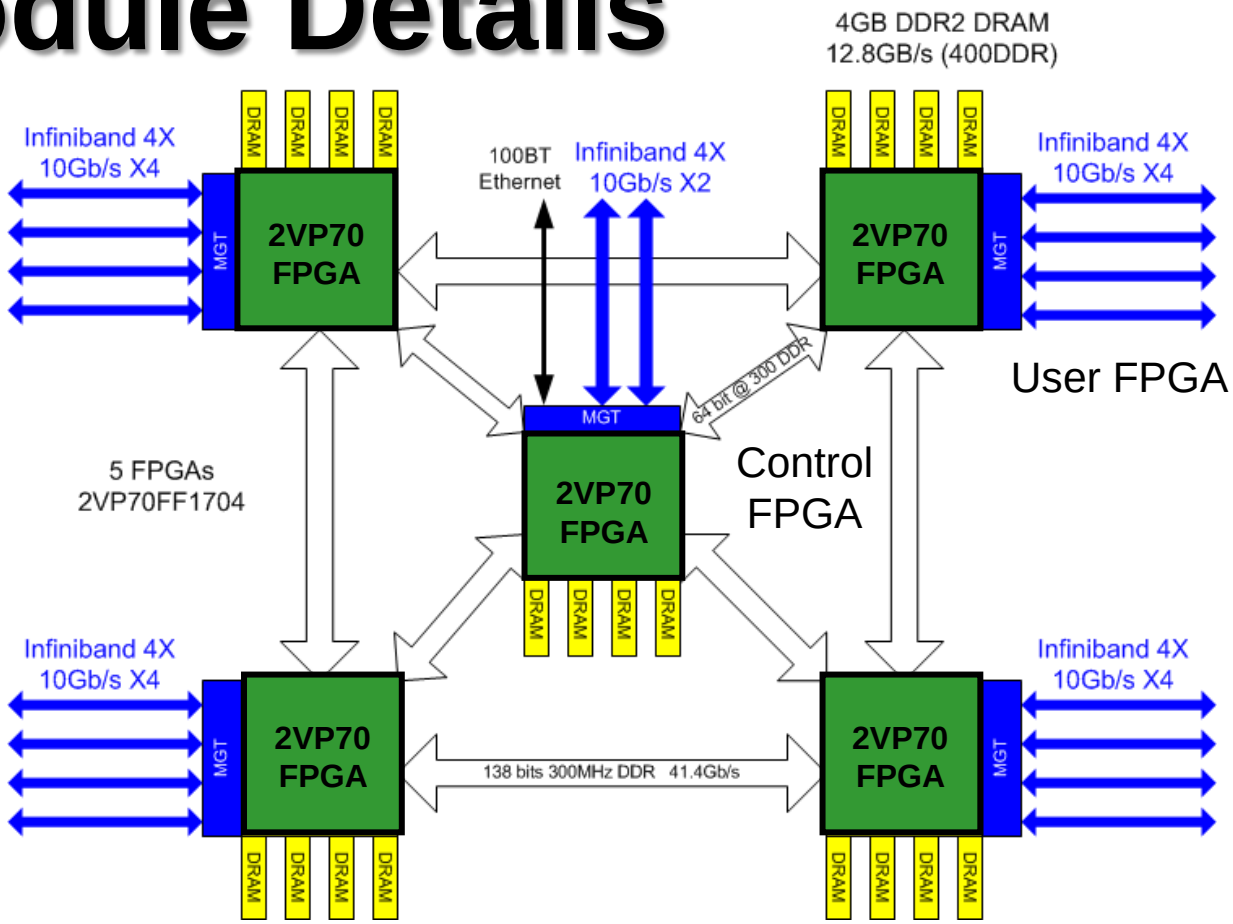
- Completed Dec. 2004 (14x17 inch 22-layer PCB)
- 5 Virtex II FPGAs, 20 banks DDR2-400 memory,
  18 10Gbps conn.
- Administration/ maintenance ports:
  - 10/100 Enet
  - HDMI/DVI
  - USB
- ~$6K (w/o FPGAs or DRAM or enclosure)

BEE2: Berkeley Emulation Engine 2



BEE2 Module Details

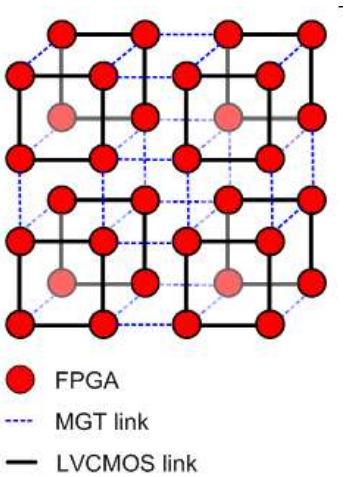
- Four independent 200MHz (400 DDR2) SDRAM interfaces per FPGA
- FPGA to FPGA connects using LVCMOS
  - Designed to run at 300MHz
  - Tested and routinely used at 200MHz (SDR) - single ended



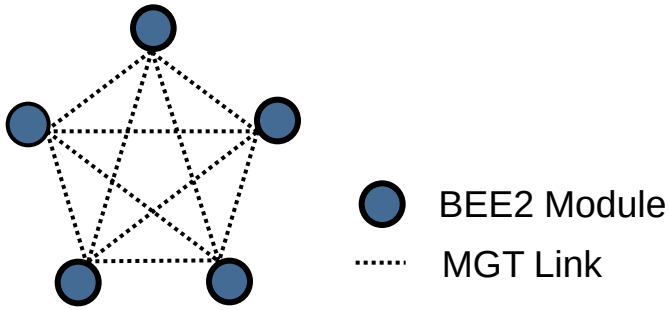
- "Infiniband" 10Gb/s links use "XAUI" (IEEE 802.3ae 10GbE specification) communications core for the physical layer interface.
- Hardware will support others, ex: Xilinx "Aurora" standard, or ad hoc interfaces.

Physical Network Topology

- The 10Gb/s off-module serial links (4 per user FPGA) permit a wide variety of network topologies.



- Example Topology: 3-D Mesh of FPGAs
- In RAMP Blue V1-V2 links are used to implement an **all-to-all module topology**



- FPGAs on-module in a ring
- Therefore each FPGA is at most 4 on-module links + 1 serial link connection away from any other in the system.
- + Minimizes dependence on use of serial links:
  - Latency in 10's of cycles versus 2-3 cycles for on-module links (ideal)
- - Scales only to 17 modules total, v3 used 3-D mesh

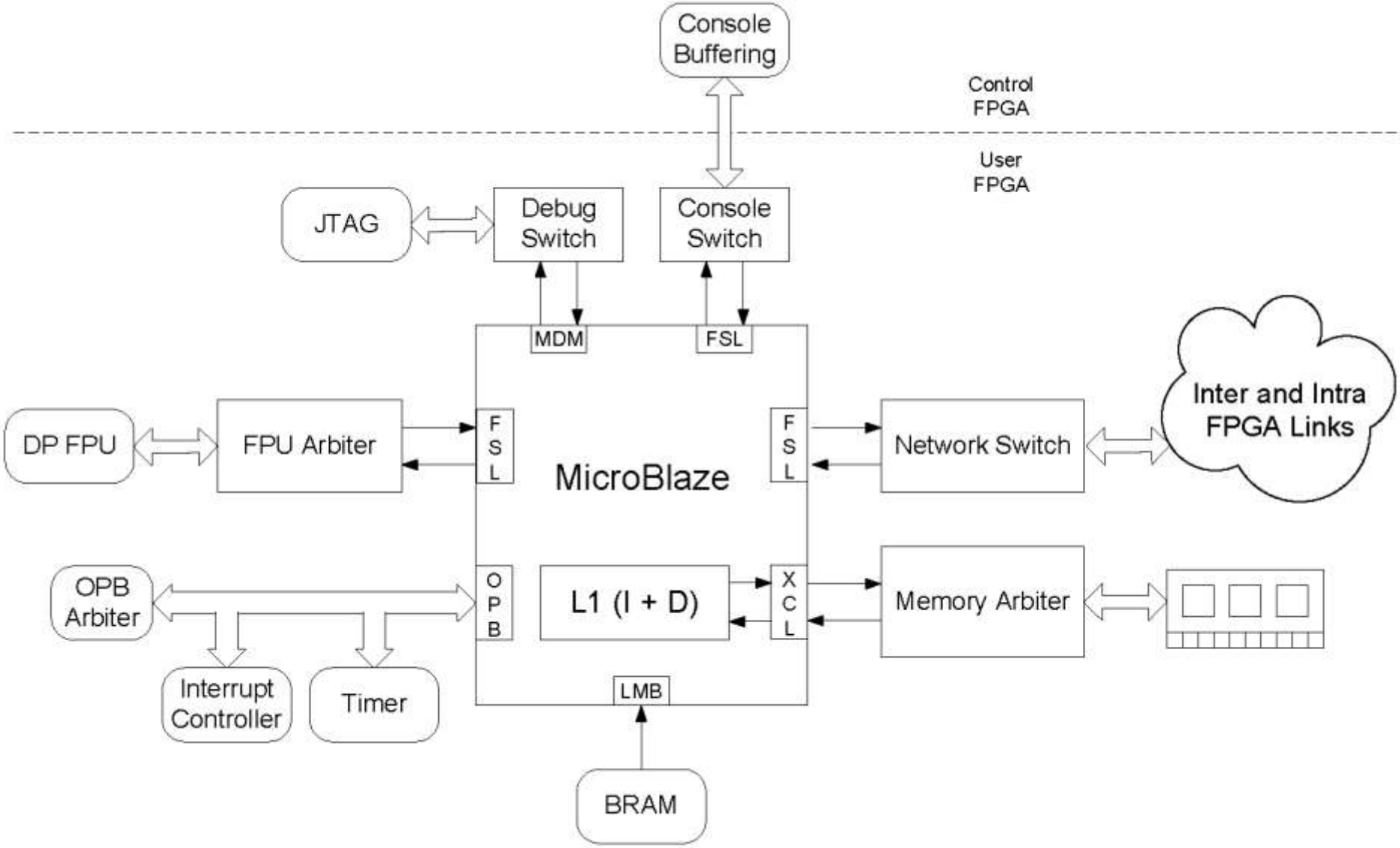
MicroBlaze V4 Characteristics

- 3-stage, RISC designed for implementation on FPGAs
  - Takes full advantage of FPGA unique features (e.g. fast carry chains) and addresses FPGA shortcomings (e.g. lack of CAMs in cache)
  - Short pipeline minimizes need for large multiplexers to implement bypass logic
- Maximum clock rate of 100 MHz (~0.5 MIPS/MHz) on Virtex-II Pro FPGAs
- Split I and D cache with configurable size, direct mapped (we use 2KB $I, 8KB $D)
- Optional single precision floating point unit (off).
- Up to 8 independent fast simplex links (FSLs) with ISA support
- Configurable hardware debugging support (watch/breakpoints) : MDM (Microprocessor Debug Module)
- GCC tool chain support and ability to run uClinux

Other RAMP Blue Requirements

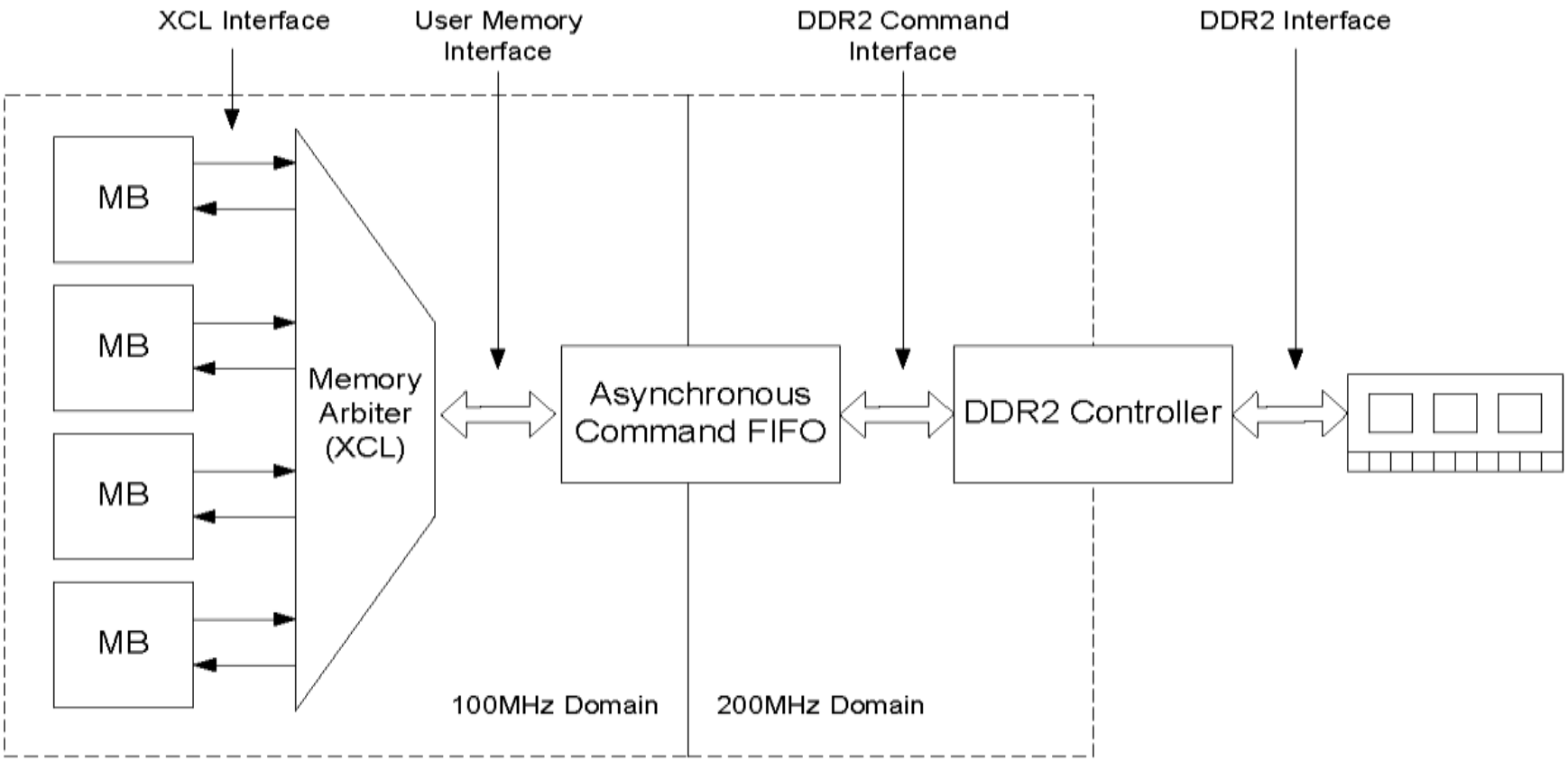
1. Gateware and software for **multiple MicroBlazes** with uClinux on BEE2 modules $\Rightarrow$
   - Sharing of DDR2 memory system
   - Shared debugging, control, and bootstrapping using the 5th "control" FPGA
2. "**On-chip network**" for MicroBlaze to MicroBlaze communication
   - Communication on-chip, FPGA to FPGA on board, and board to board
3. **Double precision floating point** unit for scientific codes

Node Architecture

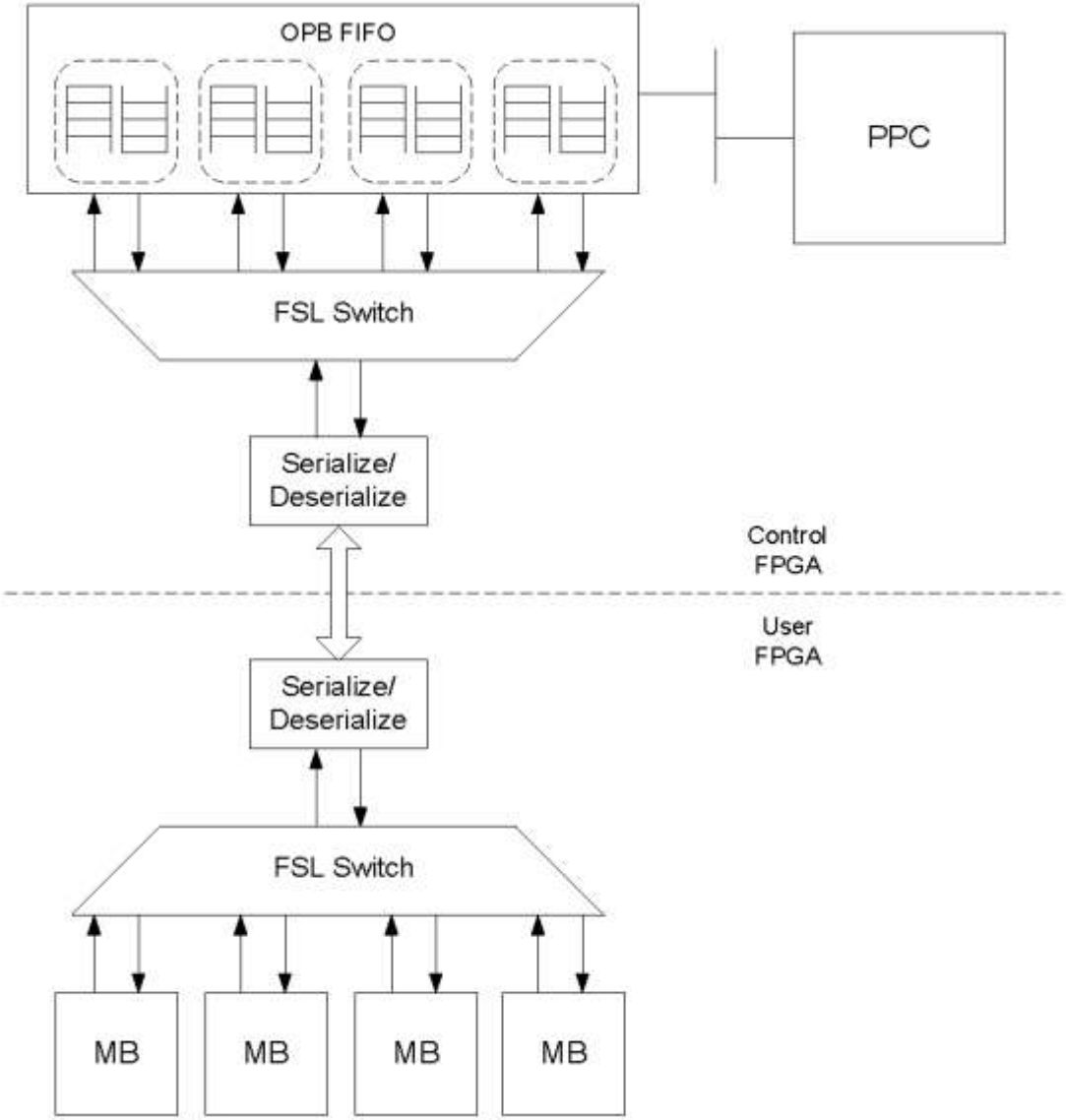


Memory System

- Requires sharing memory channel with multiple MicroBlaze cores
  - No coherence, each DIMM is partitioned, but bank management keeps cores from fighting with each other



Control Communication

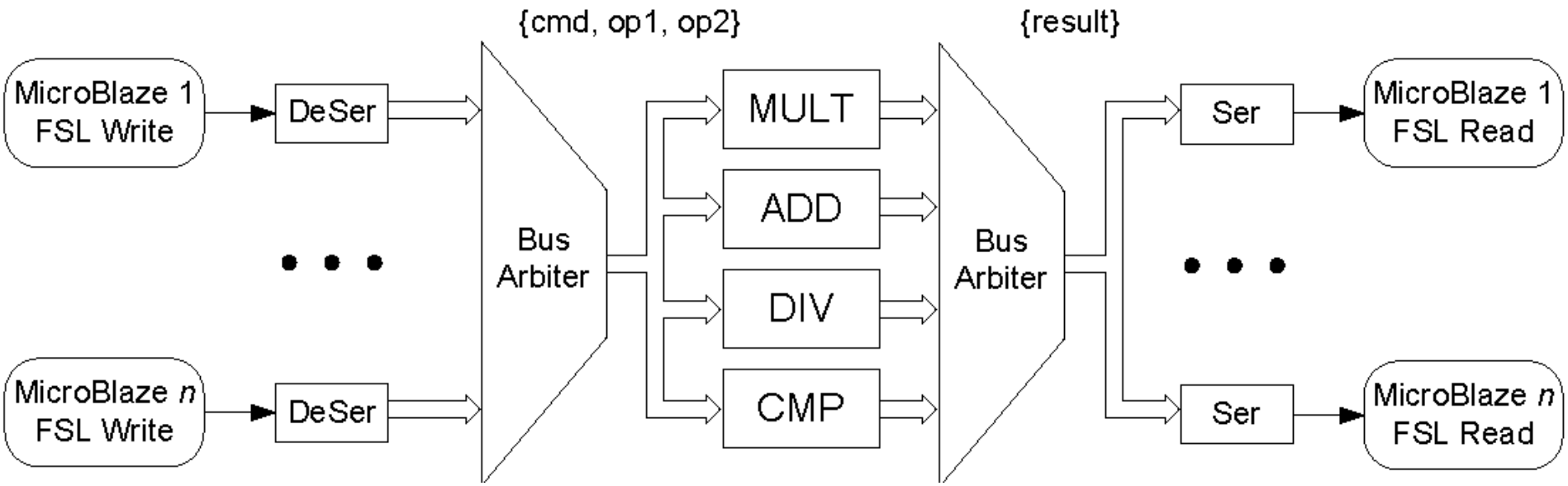


Communication channel from control PowerPC to individual MicroBlaze required for bootstrapping and debugging

- Gateware provides general purpose, low-speed network
- Software provides character and Ethernet abstraction on channel
- Linux kernel is sent over channel and NFS file systems can be mounted
- Linux console channel allows debugging messages and control

Double Precision FPU

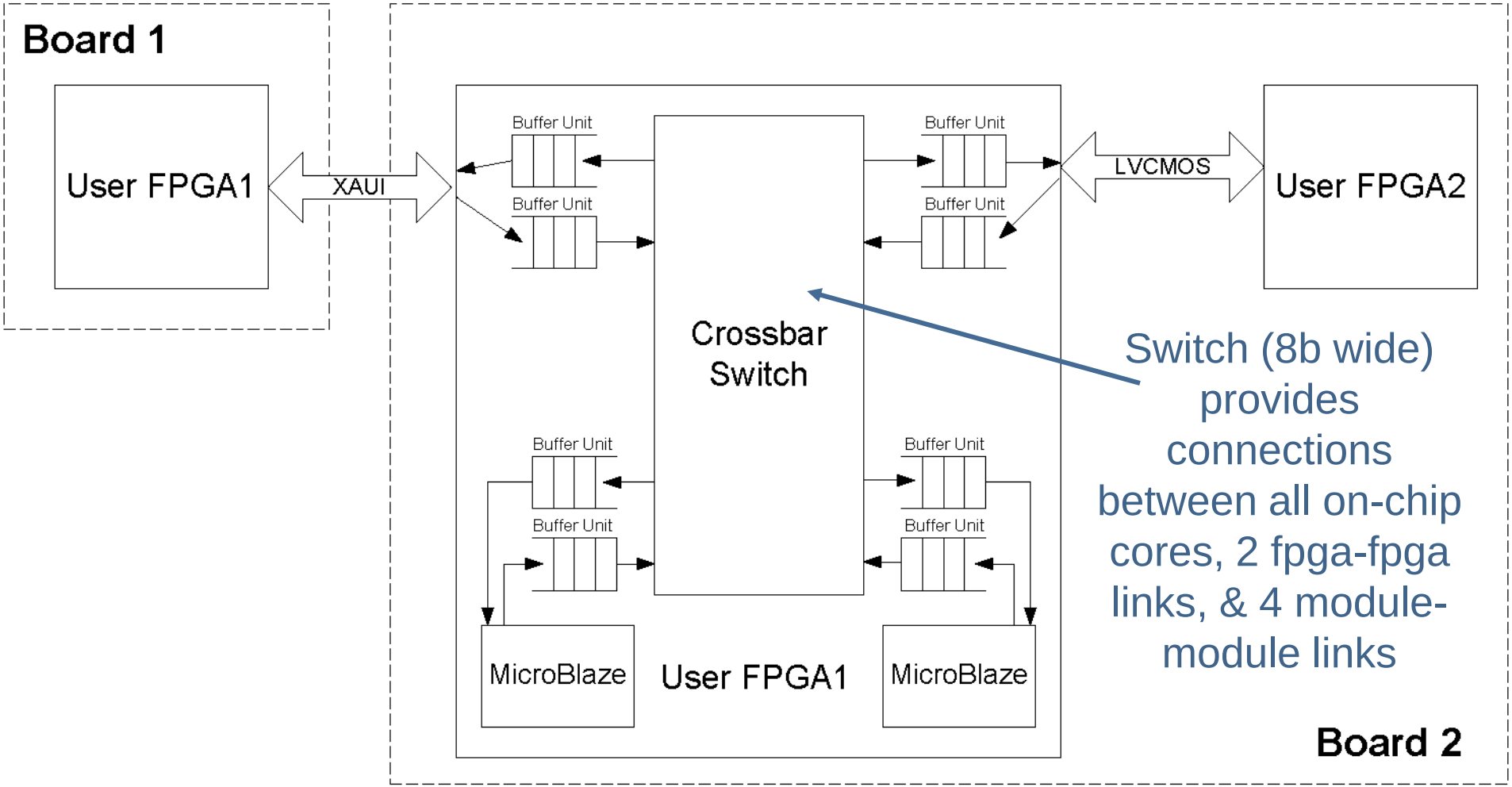
- Due to size of FPU, sharing is crucial to meeting resource budget
- Implemented with Xilinx CoreGen library FP components
- Shared FPU much like reservation stations in microarchitecture with MicroBlaze issuing instructions



Network Characteristics

- Processor/Network interface uses simple FSL channels, currently **programmed I/O,** but could/should be DMA
- *Source routing* between nodes (dimension-order style routing, non-adaptive $\Rightarrow$ link failure intolerant)
  - FPGA-FPGA and board-to-board link failure rare (1 error / 1M packets)
  - CRC checks for corruption, software (GASNet/UDP) uses acks/time-outs and retransmits for reliable transport.
- Topology of interconnect is full cross-bar on chip, ring on module, and all-to-all connection of module-to-module links (V1-2) and 3-D Mesh (V3)
- Encapsulated Ethernet packets with source routing information prepended
- Cut through flow control with virtual channels for deadlock avoidance

Network Implementation



Switch (8b wide) provides connections between all on-chip cores, 2 fpga-fpga links, & 4 module-module links

(In current versions, processor connection to network is programmed I/O limiting performance.)

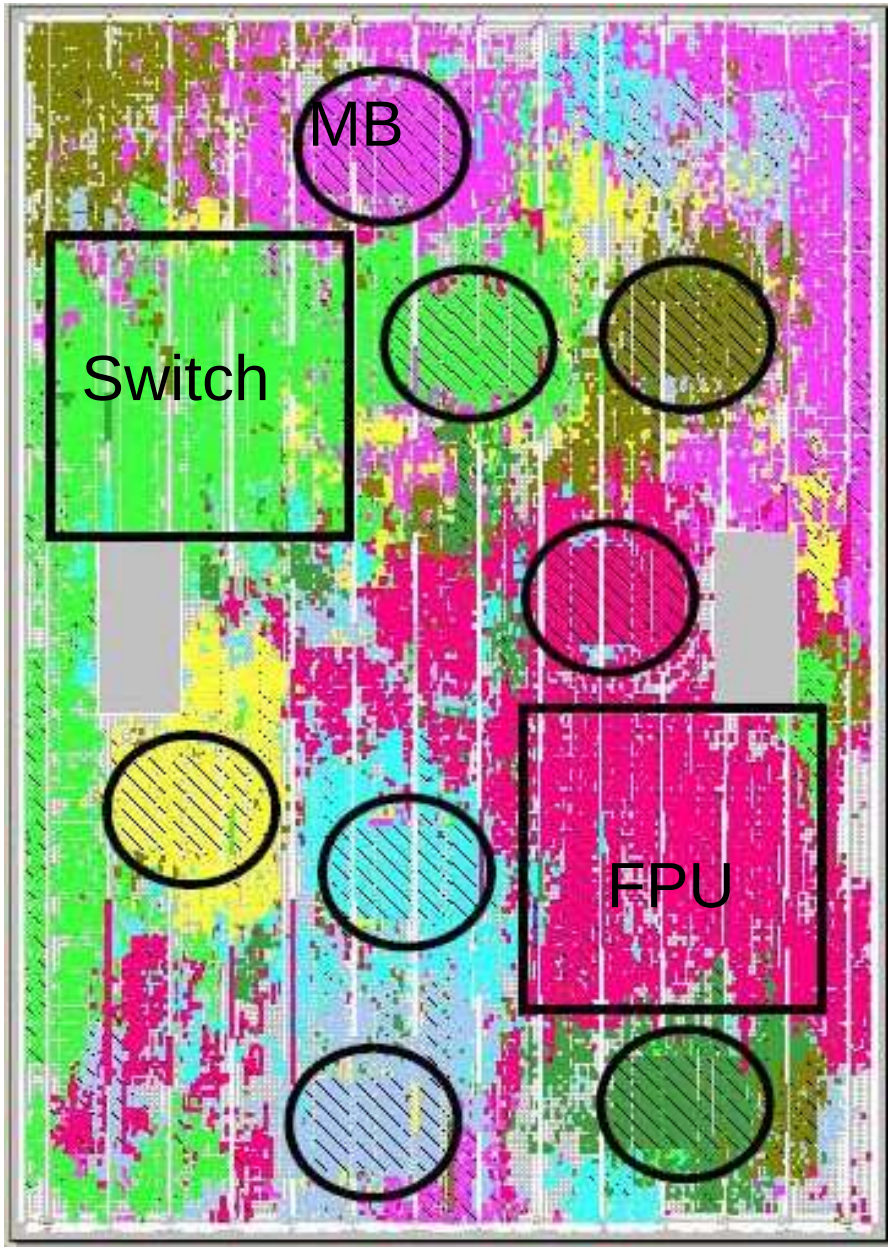
Software/Application

- Each node in the cluster boots its own copy of uClinux and mounts a file system from an external NFS file system (for applications)
- The Unified Parallel C (UPC) shared memory abstraction over messages framework was ported to uClinux
  - The main porting effort with UPC is adapting to its transport layer, GASnet, was eased by using the GASnet UDP transport layer
- Floating point integration is achieved via modification to the GCC SoftFPU backend to emit code to interact with FPU
- Runs UPC (Unified Parallel C) version of a subset of NAS (NASA Advanced Scientific) Parallel Benchmarks (all class S, to date)

CG *Conjugate Gradient, IS Integer Sort 512 cores*

EP *Embarrassingly Parallel, MG Multi-Grid on 512, 1008 cores*

FT *FFT <64 cores*

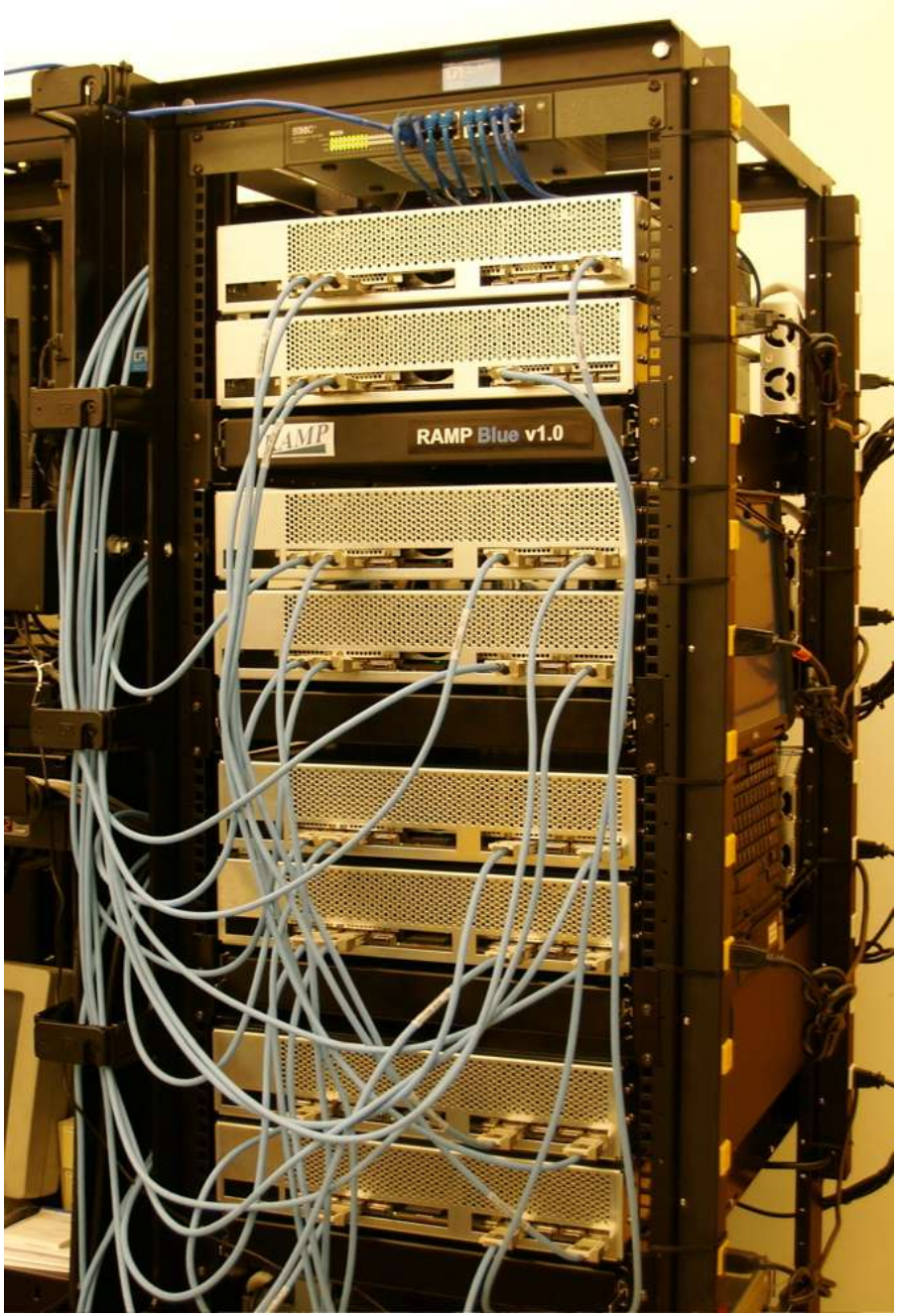


8 Core Layout

- FPU size/efficiency ⇒ shared block
- Scaling up to multiple cores per FPGA is primarily constrained by resources
  - This version implemented 8 cores/FPGA using roughly 86% of the slices (but only slightly more than half of the LUTs/FFs)
- Sixteen cores would fit on each FPGA but only without infrastructure (switch, FPU, etc)

Version Highlights

- **V1:**
  - Used 8 BEE2 modules
  - 4 user FPGAs of each module held 100MHz Xilinx MicroBlaze soft cores running uCLinux.
  - 8 cores per FPGA, **256 cores total**
  - Dec 06: 256 cores running benchmark suite of UPC NAS Parallel Benchmarks
- **V2:**
  - Used 16 BEE2 modules
  - 12 cores per FPGA, **768 cores total**
  - Cores running at 90 MHz
- **V3:**
  - Used 21 BEE2 modules
  - **1008 cores total**
- **V4**
  - Production release
  - Online system, **256 cores total**
- **Future versions**
  - Use newer BEE3 FPGA platform
  - Support for other processor cores.



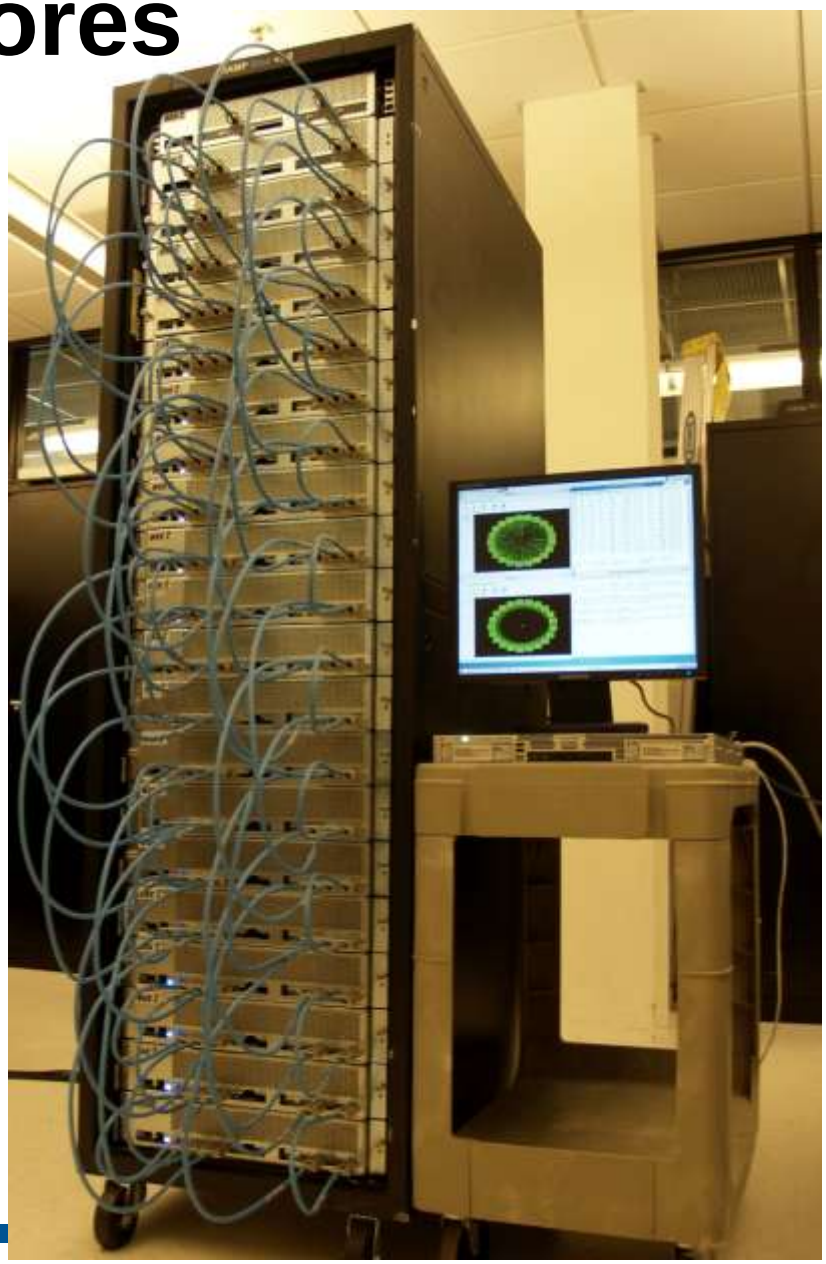
ISCA 2007

V2: 768 Core system on 16 BEE2 Modules

RAMP

V3: 21 BEE2, 1008 Cores

- 21 BEE2s, each with 48 cores, for a total of 1008 MicroBlaze cores.
- Picture also includes the NFS server, monitoring and debugging machine.



V4: RAMP Workshop and online system

Future Work / Opportunities

- Processor/network interface currently very inefficient
  - DMA support should replace programmed I/O approach
- Many of the features for a useful ***RAMP*** *(emulator)* currently missing
  - Time dilation (ex: change relative speed of network, processor, memory)
  - Extensive HW supported monitoring
  - Virtual memory, other CPU/ISA models
  - Other network topologies
- Good starting point for processor+HW-accelerator architectures
- We are very interested in working with others to extend this prototype.
  - Released version available in design repository at:

    http://repository.eecs.berkeley.edu

RAMP

Evolving RAMP Emulation Styles

- **Direct RTL**
  - Take existing RTL and synthesize to FPGA gates
  - Minor variant is to optimize RTL for FPGA synthesis
- **Virtualized RTL**
  - Improve FPGA utilization by using multiple FPGA clock cycles to model one target (emulated processor) clock cycle
- **Host Multithreading**
  - Improve emulation throughput by interleaving execution of multiple target units on one physical host engine
- **Split Functional/Timing models**
  - Faster experimentation - do not have to develop complete RTL
  - Can reuse functional model with different timing models
  - Reduces FPGA capacity requirements

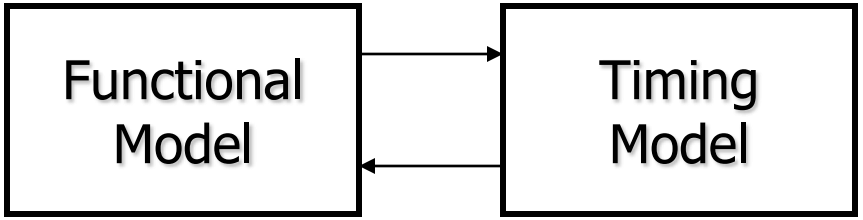
Our challenge to support all the above styles, including connecting modules using different styles.

RAMP

Virtualized RTL Improves FPGA Resource Usage

- RAMP allows units to run at varying target-host clock ratios to optimize area and overall performance
- Example 1: Multi-ported register file
  - Example, Sun Niagara has 3 read ports and 2 write ports to 6KB of register storage
  - If RTL mapped directly, requires 48K flip-flops
    - Slow cycle time, large area
  - If mapping into block RAMs (one read+one write per cycle), takes 3 host cycles and 3x2KB block RAMs
    - Faster cycle time (~3X) and far less resources
- Example 2: Large L2/L3 caches
  - Current FPGAs only have ~1MB of on-chip SRAM
  - Use on-chip SRAM to build cache of active piece of L2/L3 cache, stall target cycle if access misses and fetch data from off-chip DRAM

Split Functional/Timing Models
(HASIM Emer MIT/Intel, FAST Chiou, UT Austin)

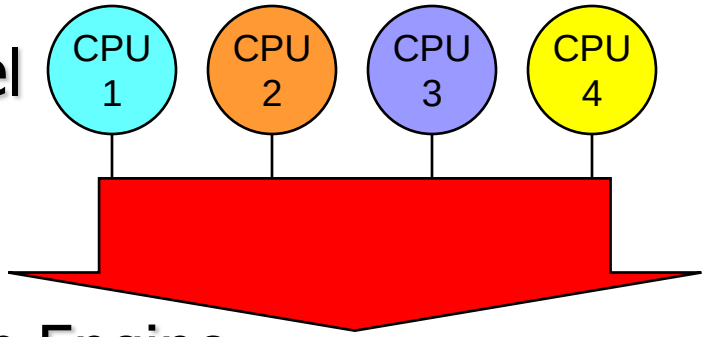


- Functional model executes CPU ISA correctly, no timing information
  - Only need to develop functional model once for each ISA
- Timing model captures pipeline timing details, does not need to execute code
  - Much easier to change timing model for architectural experimentation
- Many possible splits between timing and functional model (e.g., implement x86 ISA in software, timing in hardware)

RAMP Gold: Multithreading

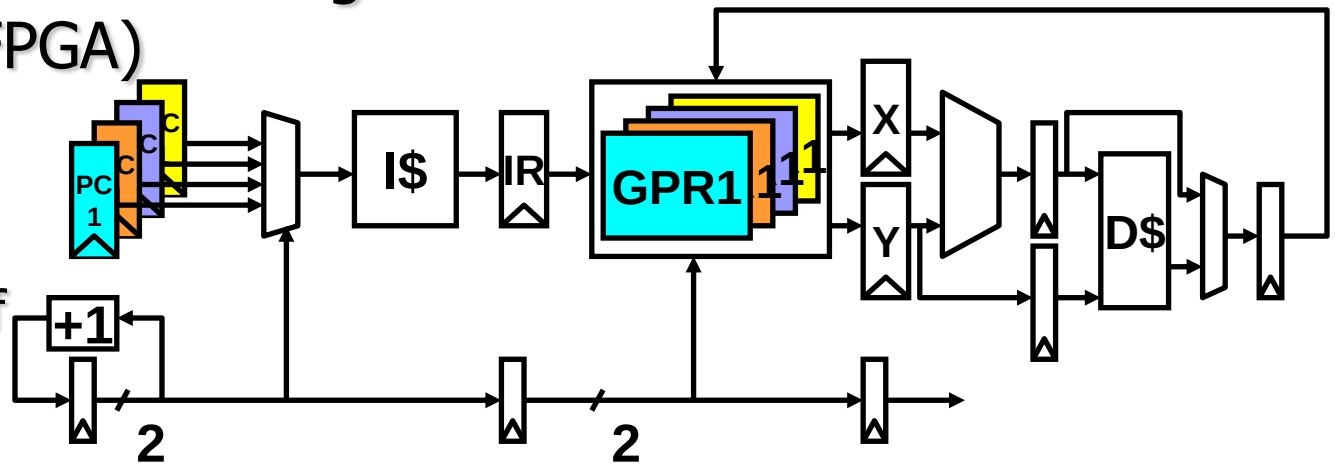
The Parallel Computing Laboratory

Target Model



Multithreaded Emulation Engine (on FPGA)

Single hardware pipeline with multiple copies of CPU state



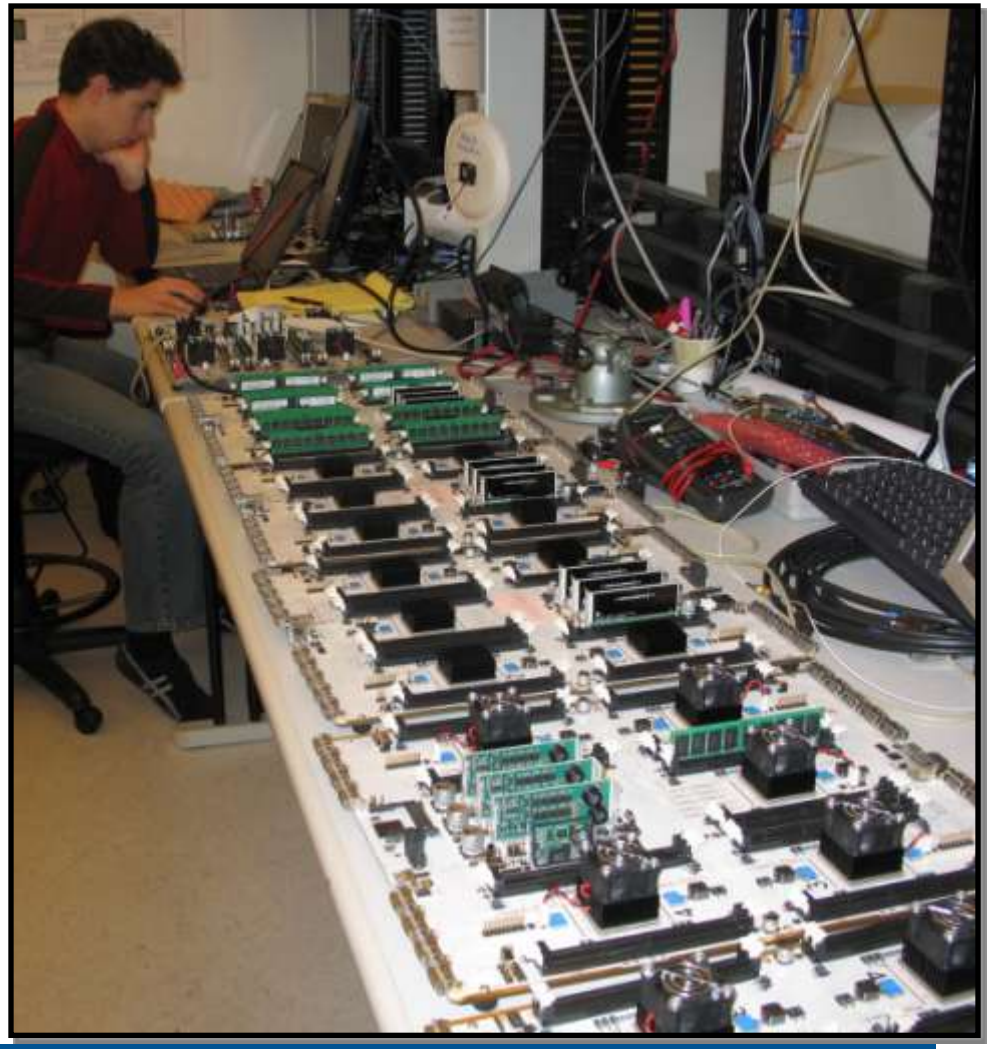
Multithreading emulation engine reduces FPGA resource use and improves emulator throughput

Hides emulation latencies (e.g., communicating across FPGAs)

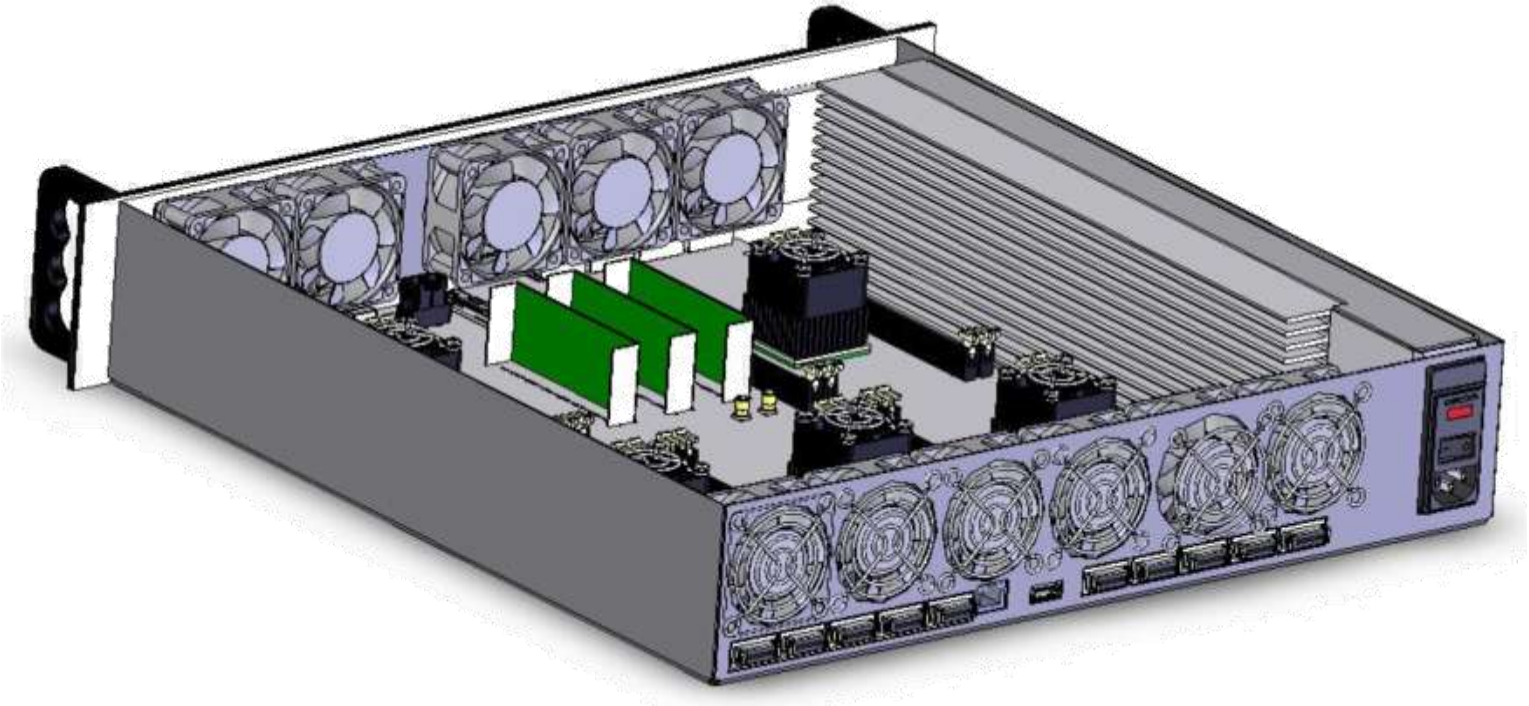
RAMP

Dealing with "Accidental Success"

- "How do I buy one?"
- How does the university sell these?
- No "Design for Manufacturing" or volume production:
  - Tight spacings
  - Thick board, DIMM socket problems
  - No standard power supply
  - No chassis
- How do we support it?



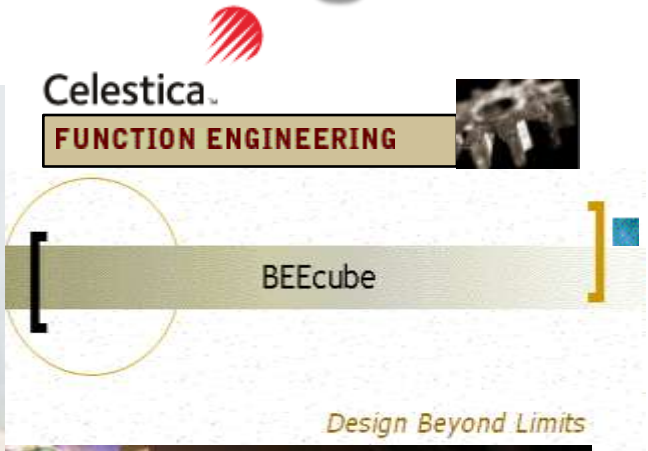
BEE2 Module Packaging

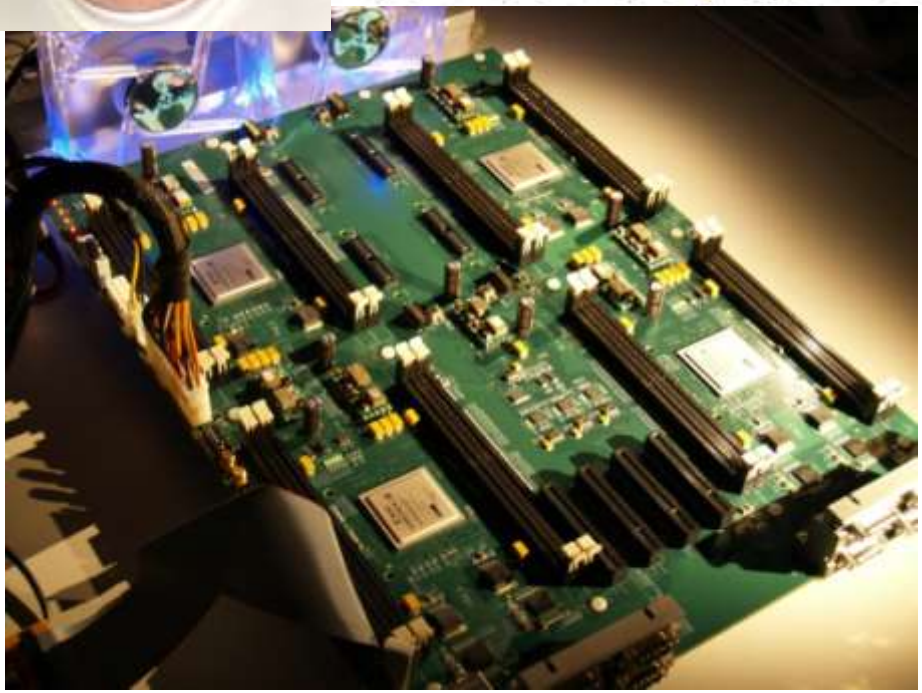


- Custom 2u rack mountable chassis
- Custom power supply (550 W), 80A @ 5volts + 12volts
- Adequate fans for cooling max power draw (300 W)
- Typical operation ~150W

BEE3 Design

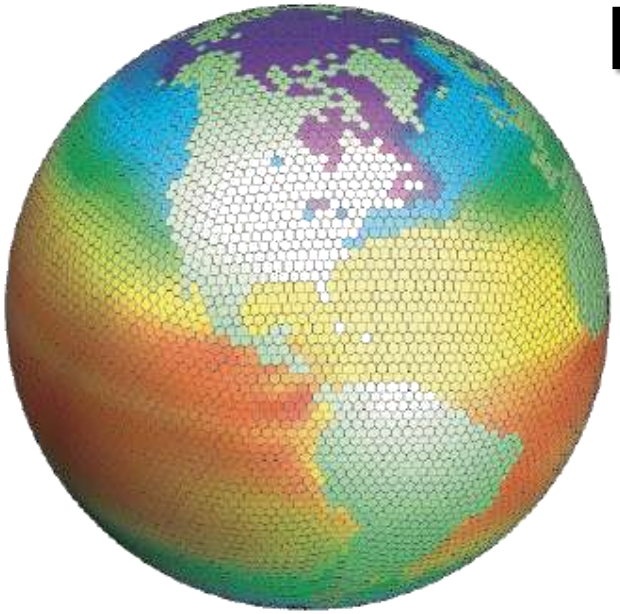
Chen Chang





- New RAMP systems to be based on Berkeley Emulation Engine version 3 (BEE3).
- *BEECube, Inc.*
  - (UC Berkeley spinout startup company)
  - Manufacturing, distribution, and support to commercial and academic users.
  - General availability 2Q08
  - Contracted to Celestica, Function.

BEE3 Development



RAMP and Climate Change

Lenny Oliker, John Shalf, Michael Wehner

Computational Research Division
Nat. Energy Research Scientific Computing Ctr
Lawrence Berkeley National Laboratory

- Accurate **climate modeling** is one of the most critical challenges facing computational scientists today
  - Ramifications in trillions of dollars
- Current horizontal resolutions (20km, best case) fail to resolve critical phenomena important to understanding the climate systems
- ***1 km-resolution are needed, at 1000x realtime***
  - *Century scale simulations complete in a month*
- Radical new approach based on ultra-efficient embedded processors

Implementation Issues

- Building such a large design exposed several difficult bugs in both hardware and gateware development
  - Reliable low-level physical SDRAM controller has been a major challenge
  - A few MicroBlaze bugs in both gateware and GCC tool-chain (race conditions, OS bugs, GCC backend bugs)
  - RAMP Blue pushed the use of BEE2 modules to new levels - previously most aggressive users were for Radio Astronomy
    - memory errors exposed memory controller calibration loop errors (tracked down to PAR problems).
    - DIMM socket mechanical integrity problems
- **Long "recompile" (FPGA place and route) times (3-30 hours) hindered progress**

The Berkeley Team

Alex Krasnov: RAMP Blue design, implementation, debugging

Andrew Schultz: Original RAMP Blue design and implementation *(graduated)*

Pierre-Yves Droz: BEE2 Detailed Design, Gateware blocks *(graduated)*

Chen Chang: BEE2 High-level Design *(graduated, now staff)*

Greg Gibeling: RAMP Description Language (RDL)

Jue Sun: RDL version of RAMP Blue

Dan Bonachea: UPC Microblaze port

Dan Burke: Hardware Platform, gateware, RAMP support

Ken Lutz: Hardware "operations"

Dave Patterson & Students of CS252, Spring 2006: Brainstorming and help with initial implementation

For more information …

1. Chen Chang, John Wawrzynek, and Robert W. Brodersen. **BEE2: A High-End Reconfigurable Computing System**. 2005. *IEEE Design and Test of Computers*, Mar/Apr 2005 (Vol. 22, No. 2).
2. J. Wawrzynek, D. Patterson, M. Oskin, S. Lu, C. Kozyrakis, J. C. Hoe, D. Chiou, and K. Asanovic. **RAMP: A Research Accelerator for Multiple Processors**, *IEEE Micro* Mar/Apr 2007.
3. Andrew Schultz. **RAMP Blue: Design and Implementation of a Message Passing Multi-processor System on the BEE2**. Master's Report 2006. *http://ramp.eecs.berkeley.edu/Publications/Andrew Schultz Masters.pdf*
4. Jue Sun. **RAMP Blue in RDL**. Master's Report May 2007. *http://ramp.eecs.berkeley.edu/Jue Sun Masters.pdf*.

Thank you!