Building BEE2: a Case for High-End Reconfigurable Computer (HERC)

BEE2 Team

UC Berkeley

http://bwrc.eecs.berkeley.edu/Research/BEE

Outline

- Motivations
  - HERC & target applications
  - Why's is HERC a good idea?
- BEE2 design
  - Hardware
  - Compiler tool chain
  - Operating system
- Performance modeling

High-End Reconfigurable Computer

(HERC)

is a computer, whose hardware can be configured on a per-application-basis to match the requirements of the task by exploiting all levels parallelism in both temporal and spatial dimensions, to achieve magnitude orders improvement in terms of cost and energy consumption for computationally demanding applications.

Applications of Interest

- **High-performance DSP/communication systems**
  - Radio telescope signal processing
  - Cognitive radio systems
  - Image processing and navigation
- **Scientific computation and simulation**
  - E & M simulation for antenna design
- **Others**
  - Large Ad-Hoc network simulation
  - Bioinformatics
  - CAD tool acceleration

High-performance DSP

- "Stream-based" computation model
  - Usually hard real-time requirement
  - High-bandwidth data I/O
- Low numerical precision requirements
  - Mostly fix-point operations
  - Rarely needs reduced floating point
- Data flow processing dominated
  - few control branch points

Scientific Computing

- Computationally demanding
  - Double-precision floating point
  - Large matrix operations, linear systems solvers (lapack), 2D/3D FFTs
- Traditionally not real-time processing, but real-time processing would offer new applications.
- Opportunities to innovate on the algorithm and mapping for reconfigurable computing

Large Ad-hoc network simulation

- Simulate 1K ~ 1M nodes in the network, communicating with ad-hoc protocol, for environment sensing and actuation applications
- Massively coarse grain level parallelism
- Event-based concurrency makes the simulation inefficient on conventional processor based parallel computers
- Near real-time performance needed to validate network design

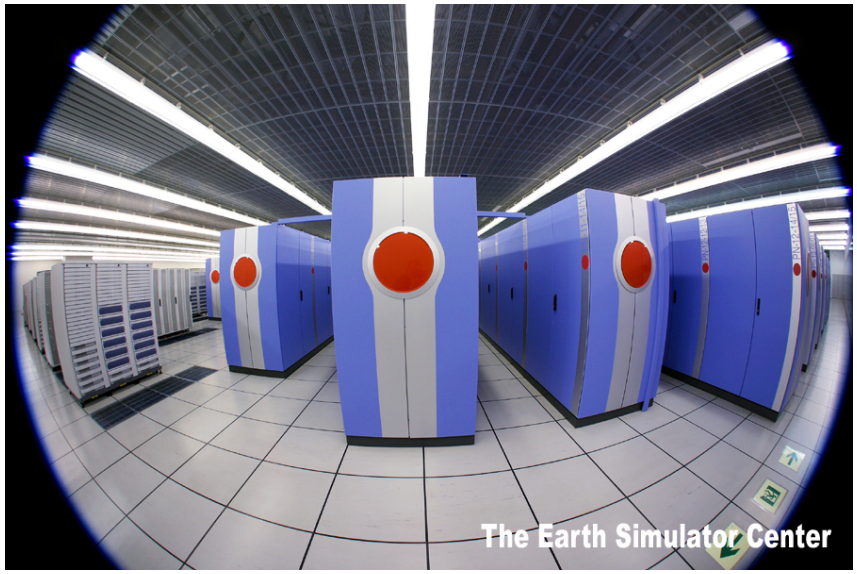
Bioinformatics

- Implicitly parallel algorithms
  - Stream-like data processing
  - Integer operations sufficient
- History of success with reconfigurable/ASIC architectures. (TimeLogic, Paracell)
- High-capacity persistent storage devices required for matching large database

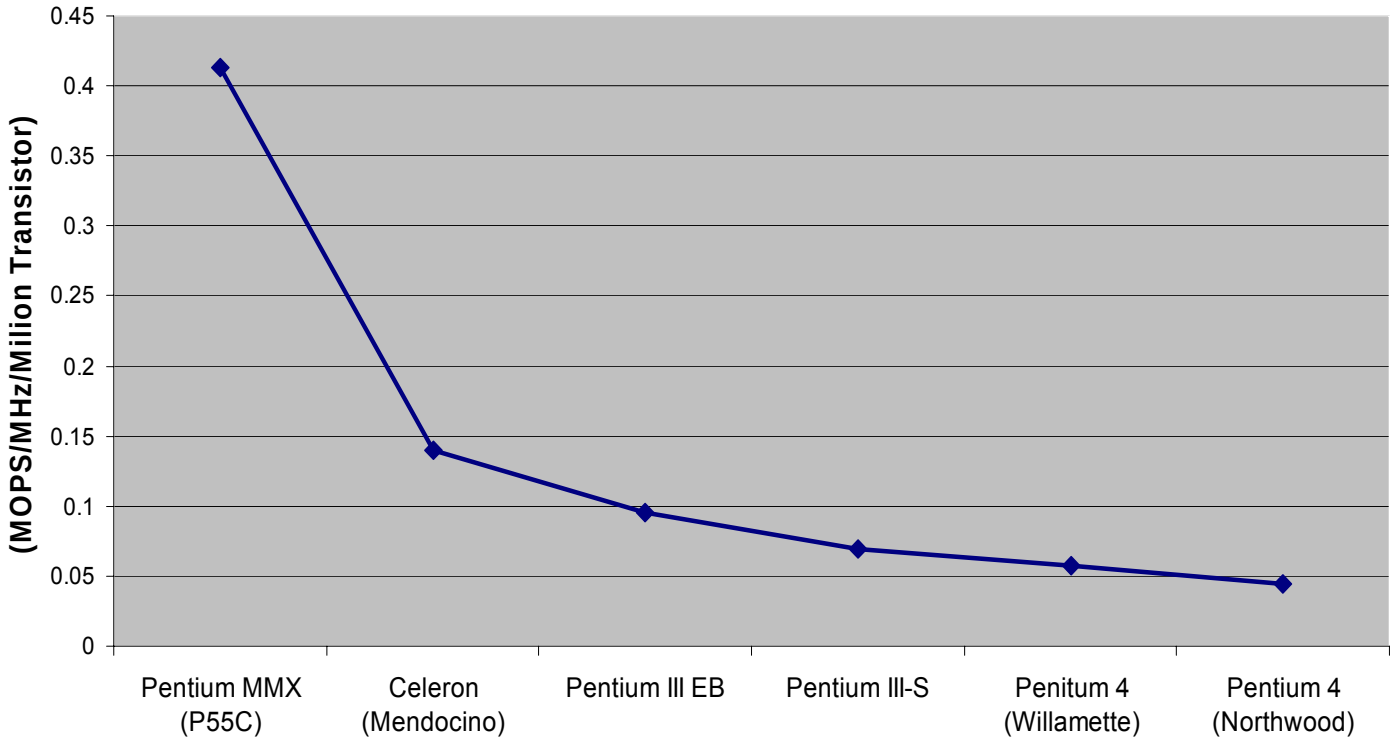
Conventional High-End Computers

Clusters of “commodity” microprocessors

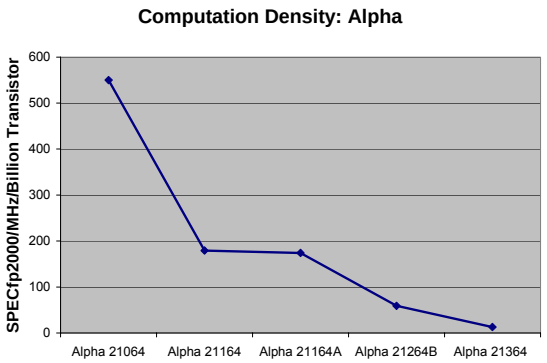
- System performance in the 100’s of GFLOPs to 10’s of TFLOP range.
- Original intension is to use commodity microprocessors for scalability and low cost
  - Cost of Earth Simulator is around **$350M**, for peak performance of 40TFLOPS
  - **$8,500 per GFLOPs**

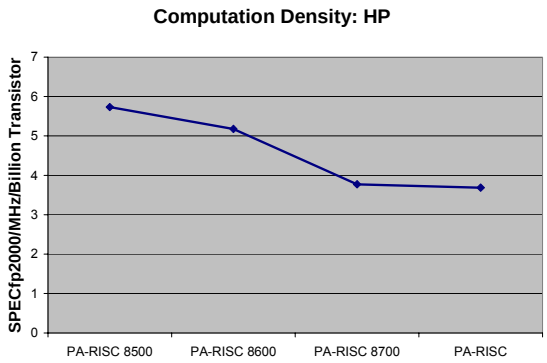


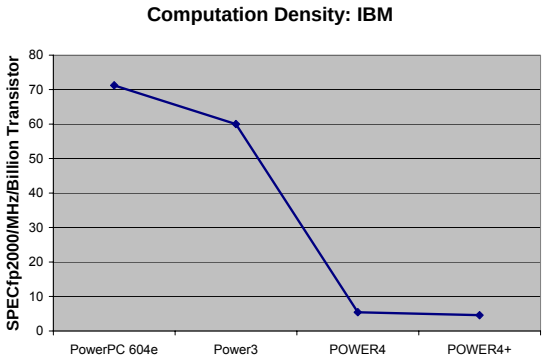
Computation Density of Processors

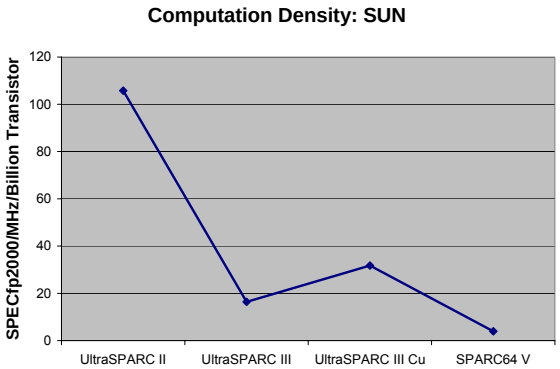


Computation Density (cont.)









Existing multi-processor HEC system problems

- Memory organization
  - Inadequate memory bandwidth
  - Processor vs. memory speed gap
  - Cache inefficiencies
- Scalability
  - Power is the limiting factor
  - Global interconnect is the bottleneck

Memory bandwidth is limited by the package pin count

- Given a specific fabrication technology, the memory I/O speed is fixed
  - The throughput per pin on the memory is fixed
- The only way to increase memory bandwidth without sacrificing latency is to increase the number of pins connected to the memory
- Microprocessors have fixed number of pins for memory access, typically supporting 2 DDR memory banks (64-bit data each)

Processor memory speed gap widens with Moore's Law

- Multiple layers of caches are necessary to keep the processor busy
- Multi-processor cache coherency and consistency problem kills the performance and limits the scalability
- Most of the power consumed on the processor is by the cache and cache related controllers
- HEC applications derive little or no benefit from cache

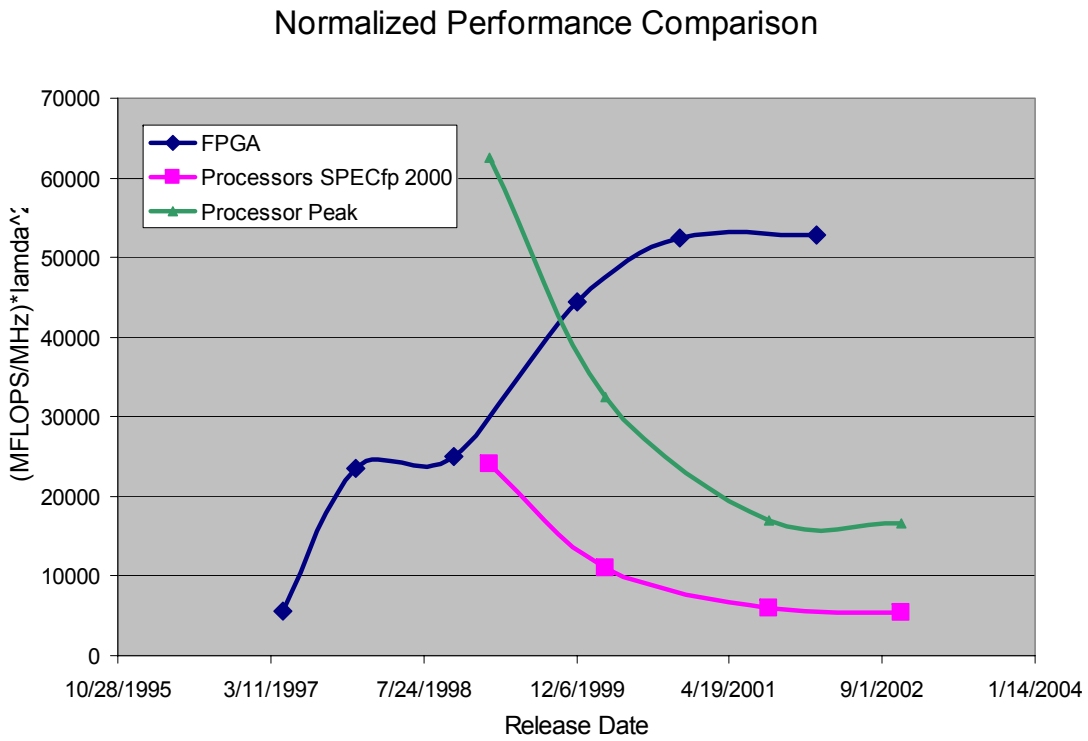
Performance is limited by power

- Processor clock rate cannot be arbitrarily increase due to the limited power budget for cooling and power distribution
  - 100 Watts = 1 Volt @ 100 Amps
- Power density limits the density of the computing fabric
  - 20 processors at 100 Watts each cannot fit in a 1U rack chassis

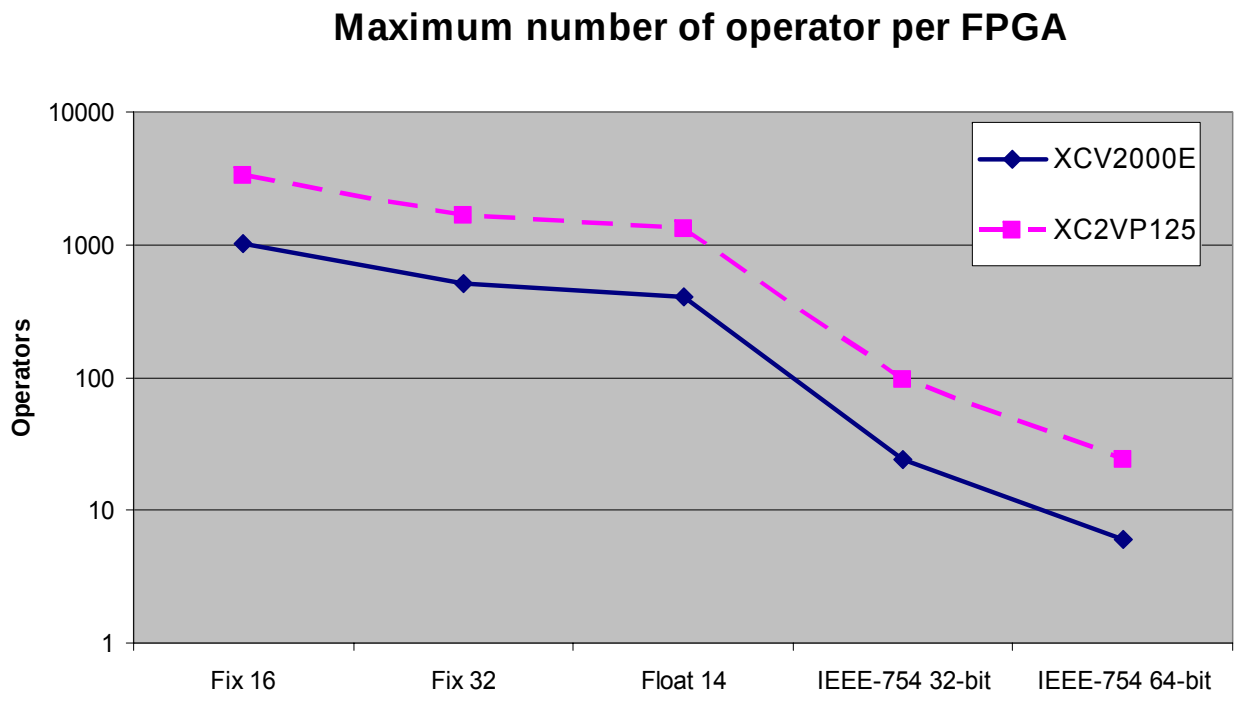
Global interconnect latency is determined by the physical machine size

- Interconnect latency increases linearly with the diameter of the physical machine size
- Interconnect latency increases with the number of computing nodes
  - 32 nodes to achieve 1 TFLOP only requires 5 layers of binary switches in a crossbar topology
  - 64 nodes to achieve 1 TFLOP requires 6 layers
- Increasing computing density lowers the global interconnect latency

Xilinx FPGA vs. Intel Processors

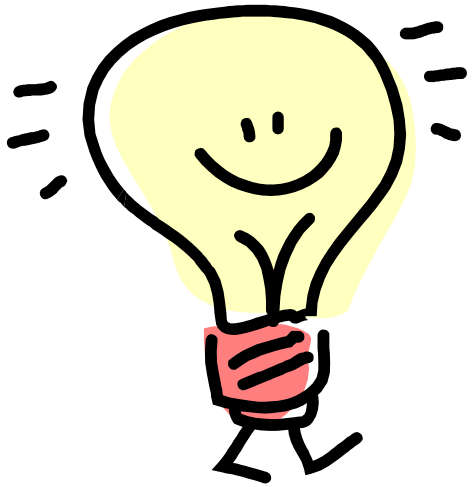


FPGA Computational Flexibility



The HERC Ideas

- A cluster of FPGAs
- Match the hardware to the application
- Do as much computation in hardware as possible, leave processor cores to handle the rest
- Intelligent memory and network architecture
- High-level parallel programming model
- A complete computer solution



Characteristics of HERC

- High computation density, lower clock rate, low power conosumption.
- No cache, computing elements operate at the same speed as memory
  - Multiple independently addressed memory banks per node
  - Internal FPGA SRAM can be user-controlled cache if needed.
- Flexible interconnection network (circuit/packet switching).
- Predictable memory and network latency permit static scheduling in real-time applications, and performance tuning.

How to design a HERC?

- The challenge
  - Free to change all aspects of the system, where to start?
  - How to qualitatively evaluate design choices? Benchmarks? “typical” programs? Usage models?
  - What’s the metric? Performance? Usability? Reliability?

Approach

- Start with a single best overall performance and economic hardware architecture
- Fuse it with existing commonly used parallel programming models best fit to the target application domains
- Evaluate the system over the target application domains by analyzing a set of representative benchmark applications on the high-level system architecture

BEE2 Design Goals

- Successor of BEE as a bigger faster hardware emulator and simulation accelerator
- As a flexible platform for experimenting with “reconfigurable computing” programming models and applications
- Demonstrate supercomputer level computation at fraction of cost and size

Hardware Design Issues

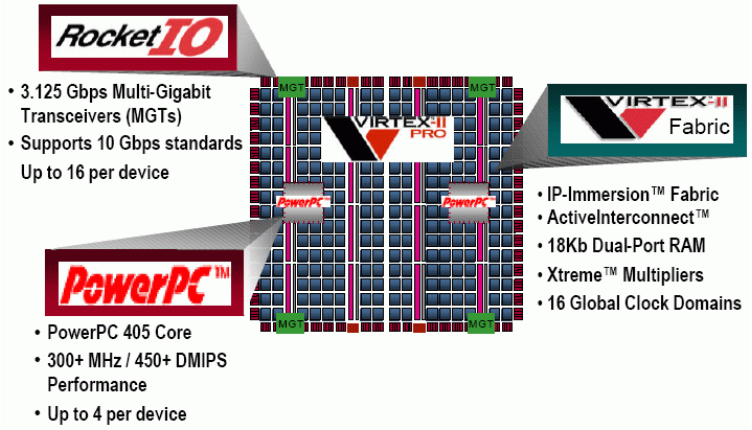
- Interconnect topology
- Memory organization
- Storage organization
- I/O & Peripherals
- Power distribution
- Mechanical chassis and cooling

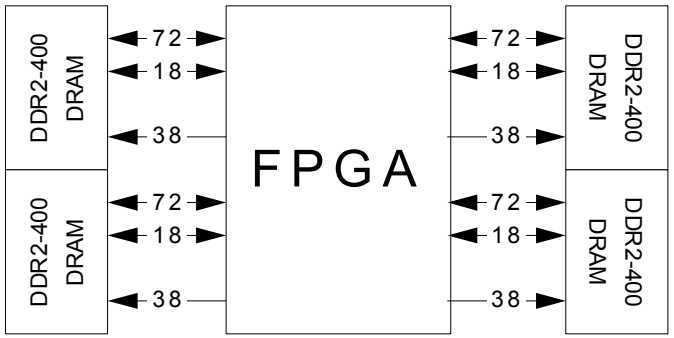
Topology comparison (1024 FPGA)

	Parallel IO			MGTs			B2 proposal	
Topology	**3D Mesh**	**4-ary Tree**	**N/4-way FPGA Crossbar**	**3D Mesh**	**4-ary Tree**	**N/4-way FPGA Crossbar**	**N/4-way Infiniband switch**	**Hybrid 4-ary Tree of mesh**
Latency (min) (ns)	40.0	80.0	160.0	120.0	240.0	480.0	80.0	80.0
Latency (avg) (ns)	806.3	373.7	642.5	2419.0	1121.2	1927.5	921.5	827.4
Latency (max) (ns)	1089.5	400.0	720.0	3268.6	1200.0	2160.0	1263.0	880.0
Bisection bandwidth (Gbps)	1300	31	2458	1693	40	3200	5120	40
Local bandwidth (Gbps)	13	15	19	17	20	25	30	30
FPGA Overhead	256	341	1280	256	341	1280	256	341
Overhead percentage	25.00%	33.30%	125.00%	25.00%	33.30%	125.00%	25.00%	33.30%
connectors off module per FPGA	4	9	8	4	4.25	8	0.25	4.25
edge length required per FPGA (in)	8	17	16	4	4.25	8	0.25	4.25

Basic Processing Element

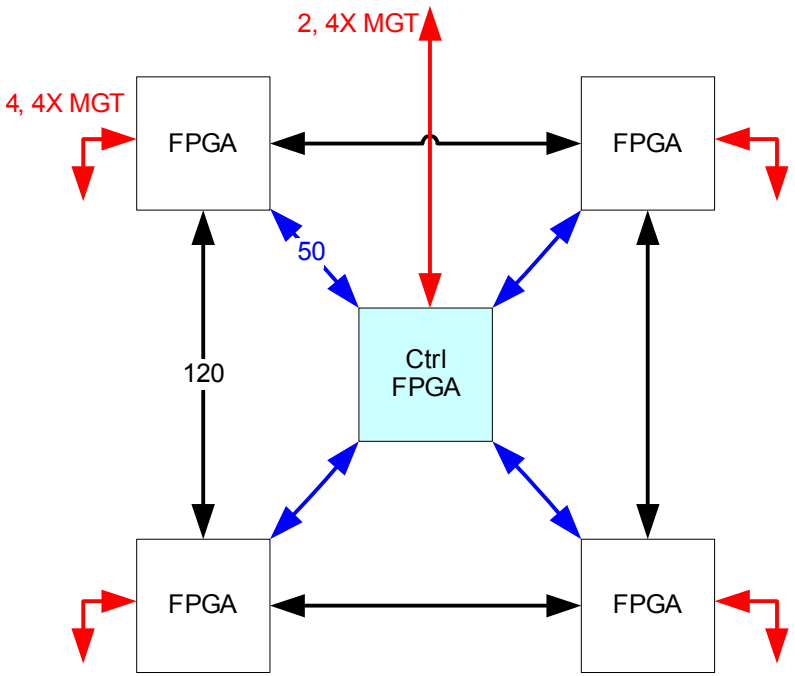
- Single Xilinx Virtex 2 Pro 100 FPGA
  - ~100K logic cells
  - 2 PowerPC405 cores
  - 444 dedicated multipliers (18-bit)
  - 1MB SRAM on-chip
  - 20X 3.125-Gbit/s duplex serial communication links (MGTs)
- 4 physical DDR2-400 banks
  - Each banks has 72 data bits with ECC
  - Independently addressed with 16 banks total
  - Up to 12.8 GBps memory bandwidth, with maximum 8 GB capacity
  - Up to 4 GBps/GFLOPs





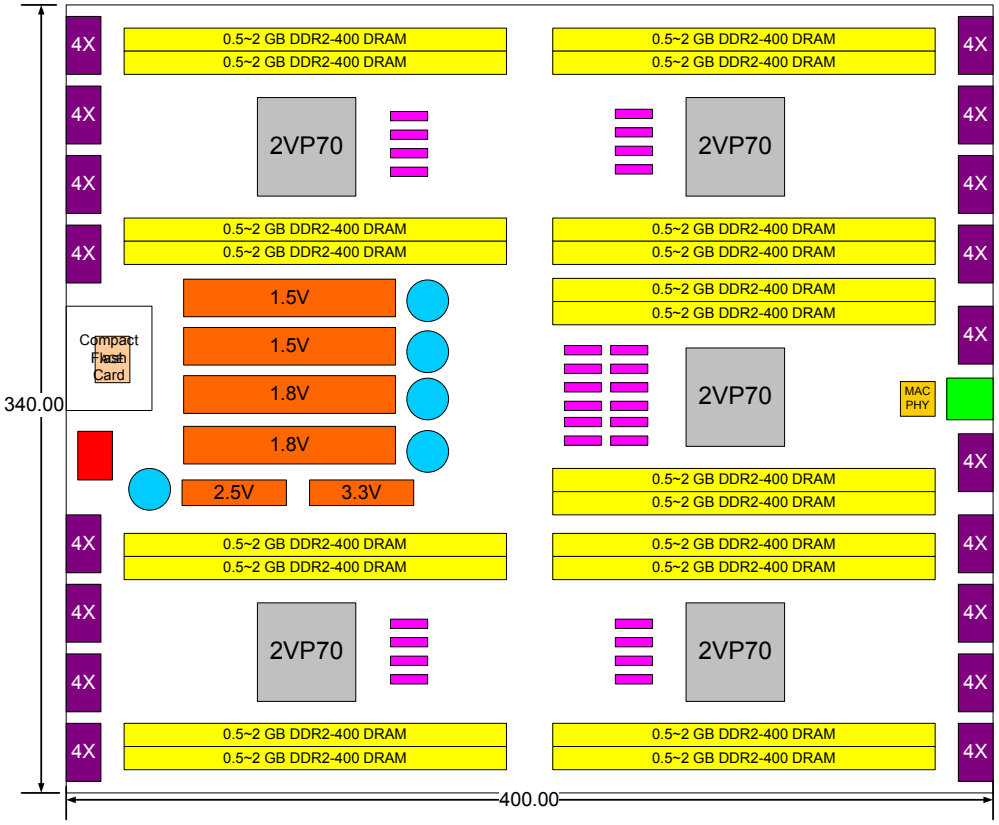
Computing Node

- 4 processing element + 1 control element
- Direct mesh connection between computing nodes
  - 120 bit 200MHz DDR
  - 48 Gbps per link
- Star connection from control node to computing nodes
  - 50 bit 200 MHZ DDR
  - 20 Gbps per link



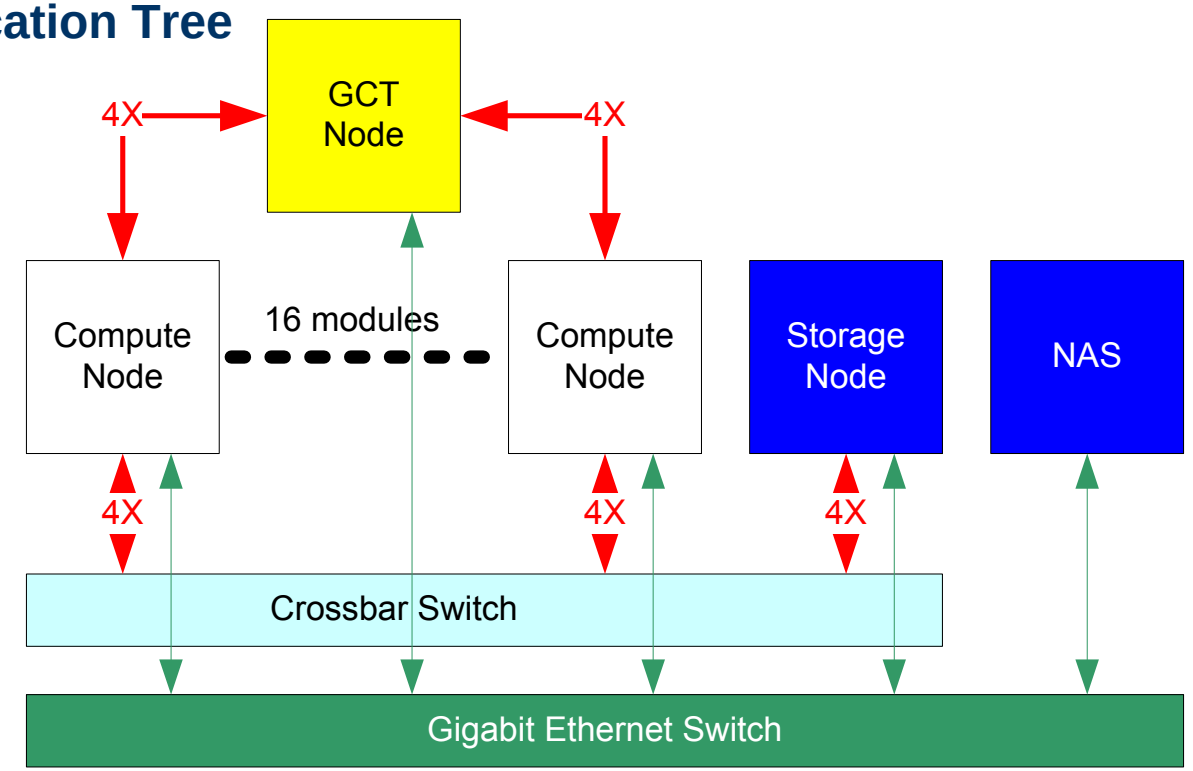
B2 Module: board layout

- Muti-purpose module
  - Compute node
  - GCT node
  - Crossbar node
  - Storage node
- Configuration through PCB assembly-time population of components



Internode Connections

Global Communication Tree

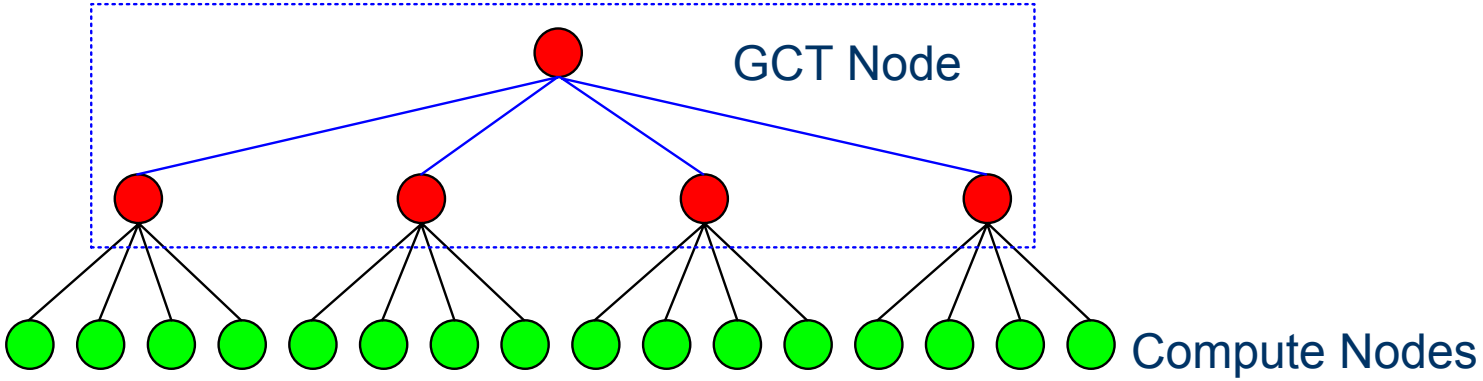


MPI_B2, Stream

Admin, UI, NFS

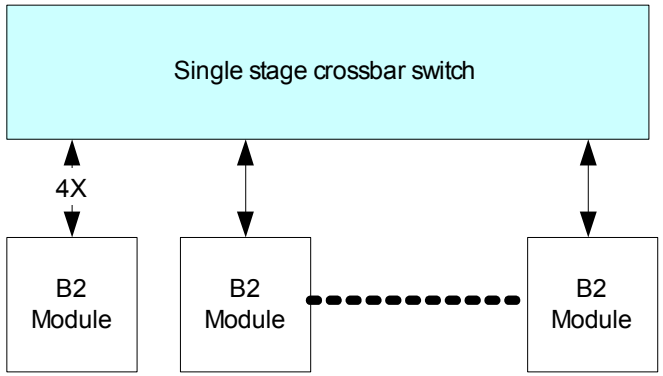
Global Communication Tree

- 4-ary tree configuration
  - Off-module 4X Infiniband connection @ 10 Gbps duplex
  - On-module 50-bit 200DDR connection @ 20 Gbps
- Every 16th B2 modules act as a tree node
- Each tree node/leaf node has up to 8GB DRAM



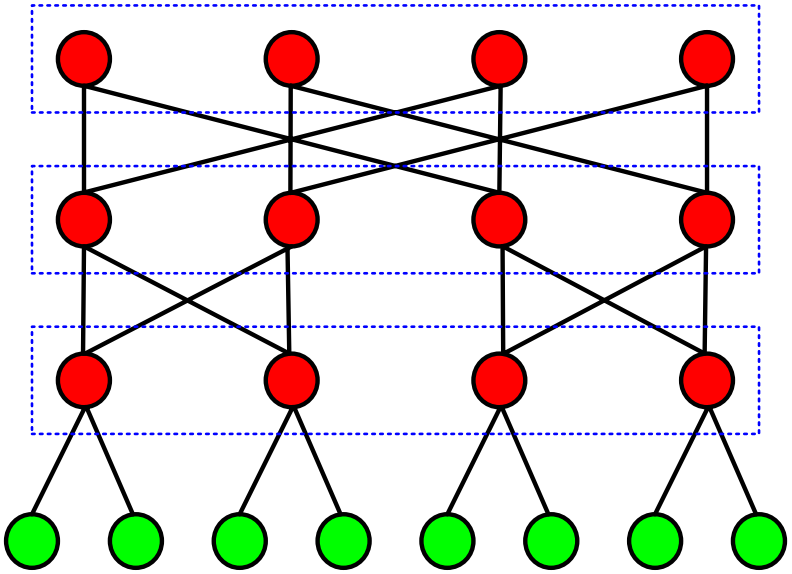
Crossbar Implementation #1

- Specialized crossbar switch implemented in ASIC (Mellanox, Voltaire)
- 24 ~ 96 4X ports
- 200~1000 ns switch latency
- 400~1200 ns FPGA to FPGA latency
- 480Gbps ~ 1.92Tbps full duplex constant cross section bandwidth
- Good for larger systems with 16 or more compute nodes
- ~$400 per port



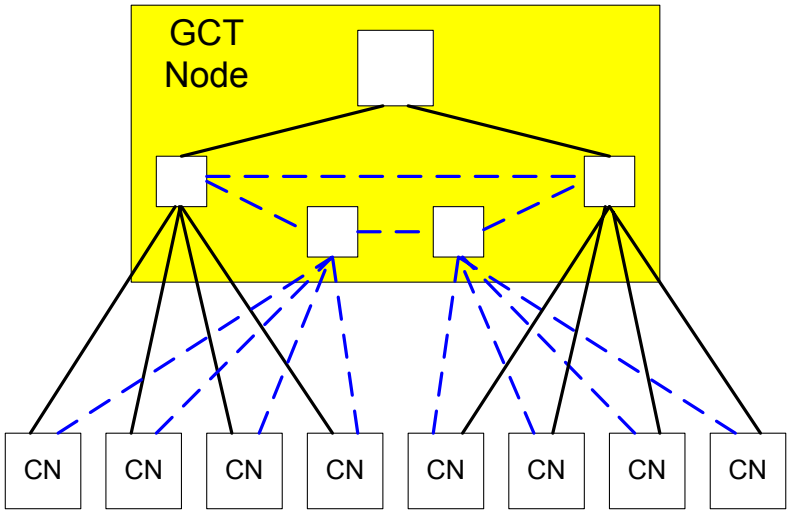
Crossbar Implementation #2

- Uses B2 modules as crossbar node
- Efficient for small system with less than 16 compute nodes
- Can shared GCT node resources when less than 8 compute nodes



8 Compute Nodes System Example

- Only 1 more B2 modules need acting as both the GCT node and crossbar switch
- In a 4 compute nodes system, only half of the GCT node FPGAs are used



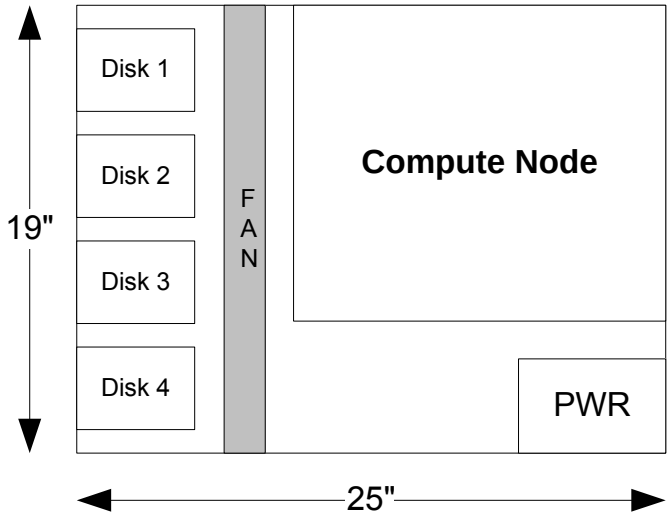
Disk Storage Schemes

- Compute node local working storage
  - OS, scratch and swap space
  - 1~2 SATA disks per node
  - 250 ~ 500 GB capacity per node, RAID 0
- Multi-node shared working storage
  - User work area
  - implemented with storage nodes
  - Up to 12 SATA disks (3TB capacity) per storage nodes
  - Any number of storage node in the system, RAID 5
- External persistent storage
  - User home directories
  - External SAN or NAS

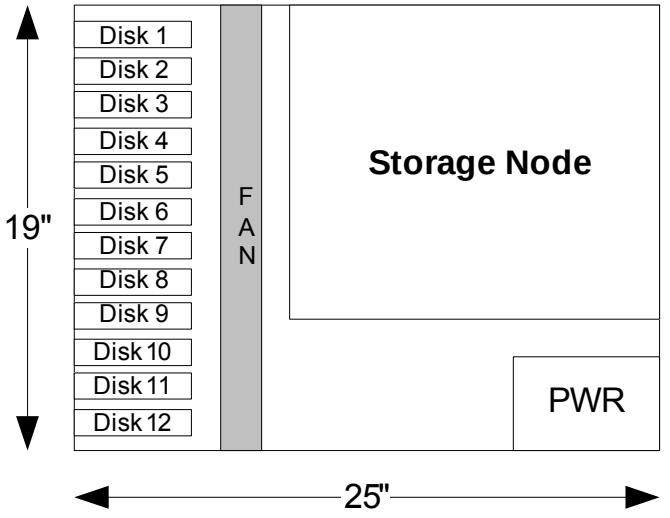
I/O and Peripherals

- Each leaf module provides up to 320 Gbps I/O bandwidth in 16 Infiniband 4X connectors
- No peripherals directly on module
- Gigabit Ethernet interface used as the primary user interface through remote access

Chassis Configurations



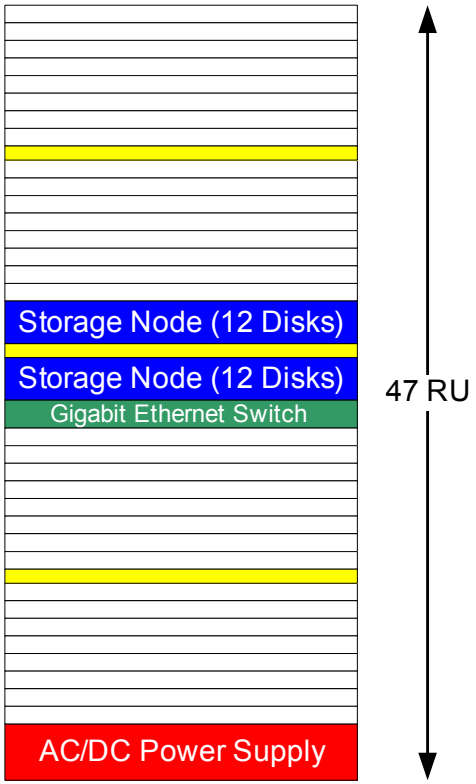
1RU



3RU

Rack Capacity

- 32 compute nodes, 3 DHSM modules, 2 Storage node
- Peak performance: 25 TOPS or 1 TFLOPS
- 512 GB local memory, 128 GB global shared memory
- 16 TB local disk, 6 TB shared work storage
- 5 Tbps I/O bandwidth
- 12 Kwatt power consumption
- Cost: ~ $250K-500K
- Performance/Price: ~$250-$500/Gflops



Power Distribution

- Each rack has a single 12KW 220VAC to 48VDC power supply, distributed through external power bus bar
- Each 1U chassis has one 300W 48VDC to 5VDC regulator, distributed through internal power cables

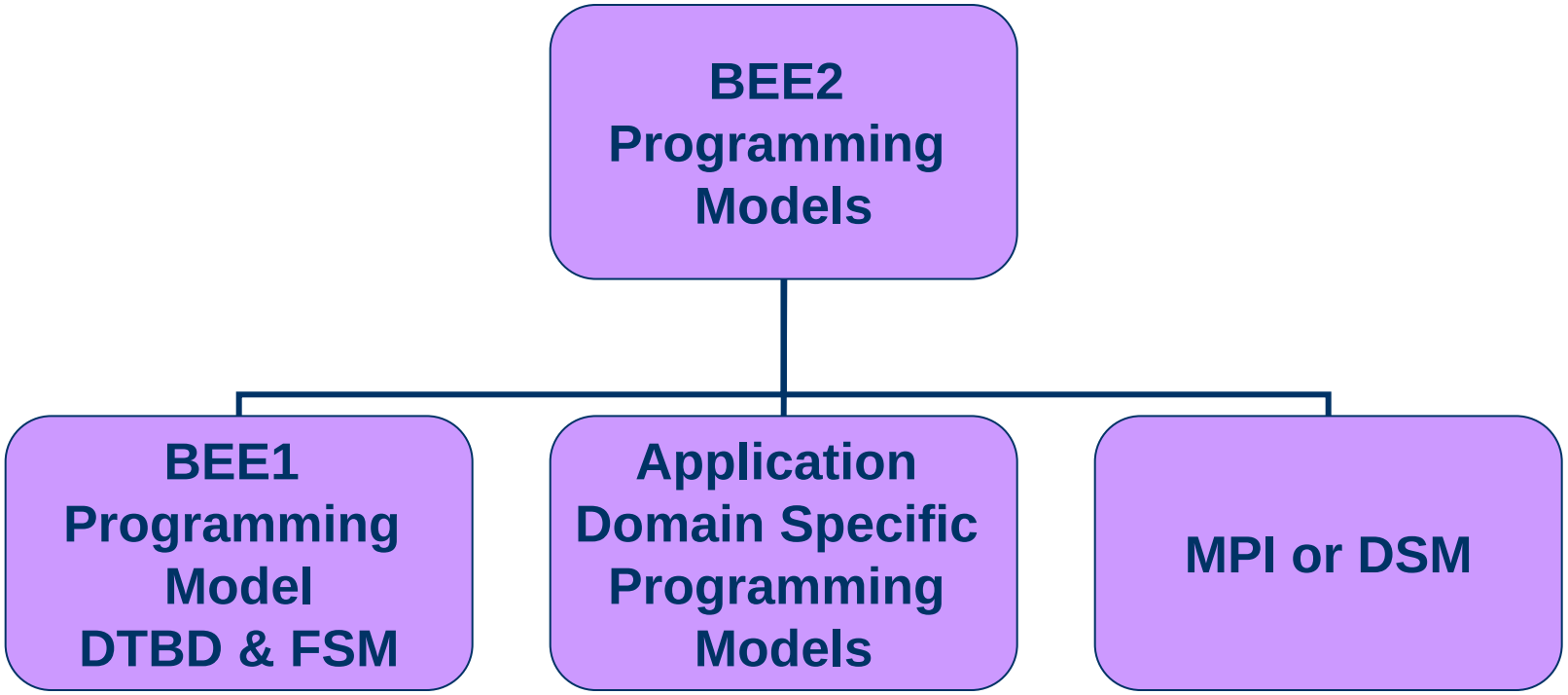
Hardware Design Summary

- Highly modular design, single PCB module reused across the whole system
- All commodity COTS component
- Flexible machine architecture/interconnect
- Scalable from single module to thousands of FPGAs in a system

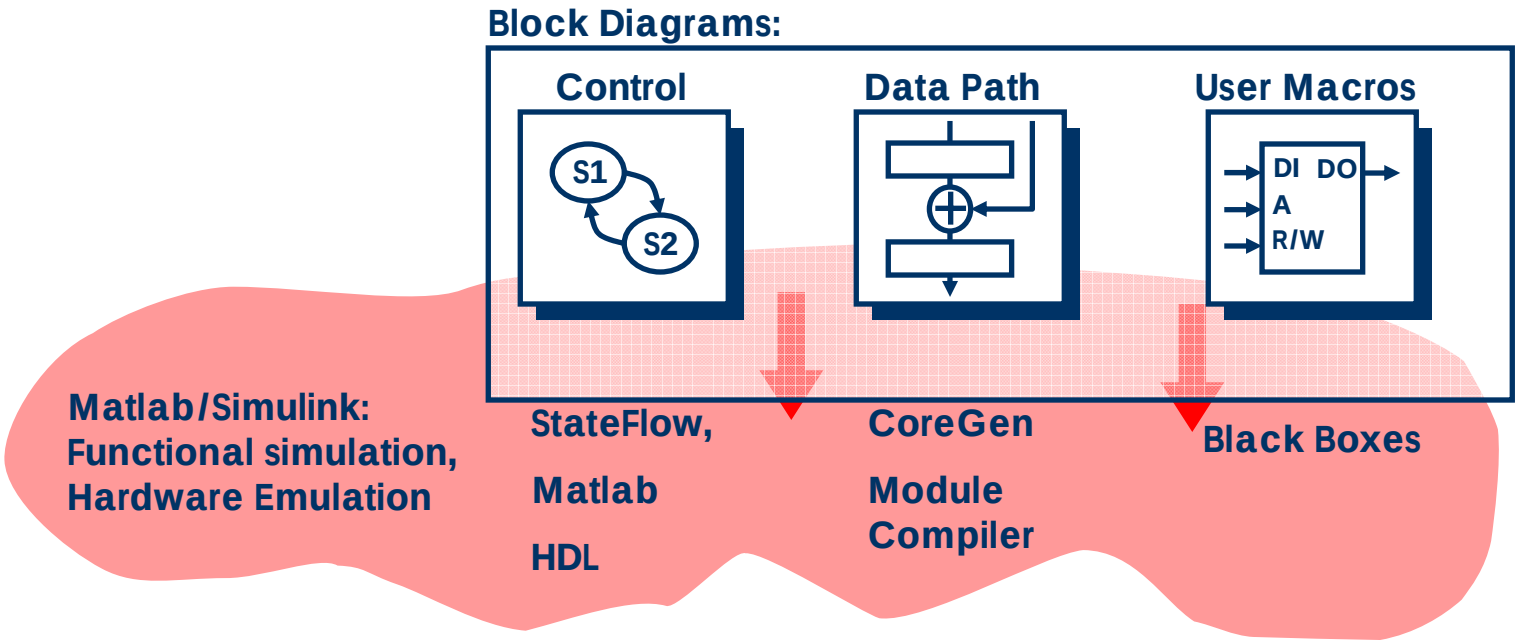
Software Design Issues

- User programming model
- Run-time libraries
- Operating system requirements
- Debugging methodology

Programming Model Overview



Programming Model (1)
BEE1: Discrete Time Block Diagram with FSM



Programming Model (2)
Special Application Domains

- For particular application domain of interest
  - Specialized computation model, programming model, specification language, debugging, and visualization
  - To maximize BEE2 hardware system performance, and increase user productivity
- Example applications domains
  - Regular grid/mesh finite element simulation
  - Network simulation

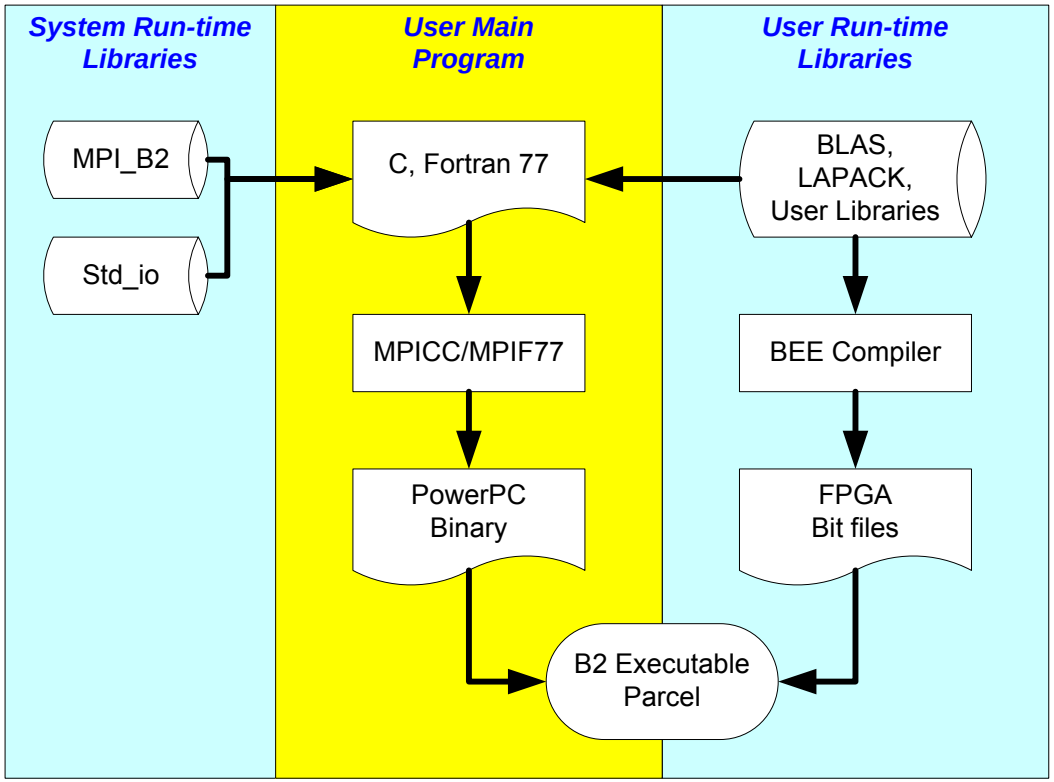
Programming Model (3a)
General Purpose Parallel Programs

- Most commonly used for distributed memory computer systems
  - Message passing interface (MPI)
  - Distributed shared memory (DSM)
- BEE2 can support both through common MPI-like communication layer in hardware
- Use GCT structure to optimize collective communications and more efficient global memory sharing

Programming Model (3b)
User application kernels & libraries

- BEE1 programming model + abstractions
  - *DTBD with FSM*
  - *Abstraction layer extensions*
    - Complex data types: single, double, complex, vector, matrix
    - Virtual memory and I/O
    - Stream data access and processing
    - Shared memory model with on-die processor cores
- Can be automated by compilers and/or application domain specific library generators

Compiler Tool Chain Overview



System Run-time Libraries

- MPI_B2
  - Based on MPIch
  - Collective communication implemented with direct hardware support, and common reduction functions in hardware
  - Includes some MPI2 extensions, such as single-sided communication, collective I/O.
- Std_io
  - Leverage standard Linux std_io library.

User Libraries

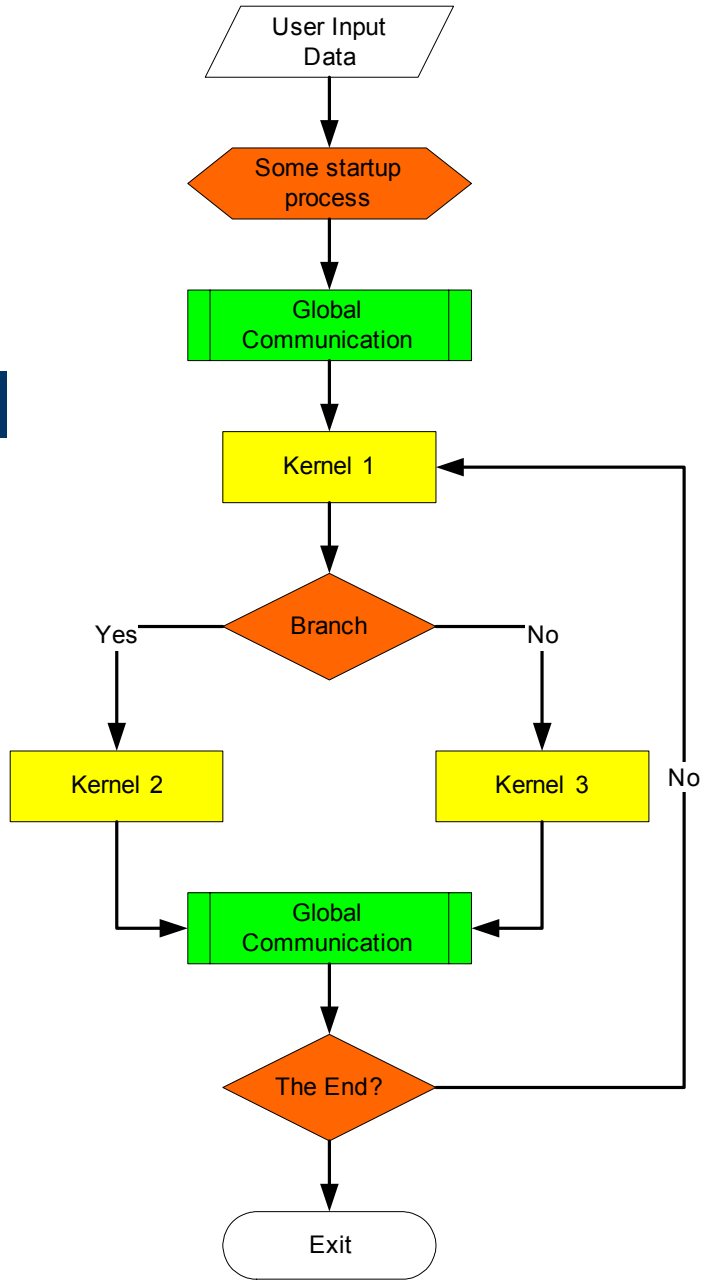
- Application domain specific libraries ported to BEE2 FPGA hardware implementation
  - BLAS
  - LAPACK
- Performance tunable at installation time to specific machine parameters

Program Execution Model

- Parallel program level
  - Single Program Multiple Data (SPMD)
  - Communicating through MPI
- Individual process level
  - Sequential execution with predefined library functions
  - Library kernels implemented on FPGA fabric and dynamically loaded at run-time
- Library kernel level
  - Data parallel execution of streaming data

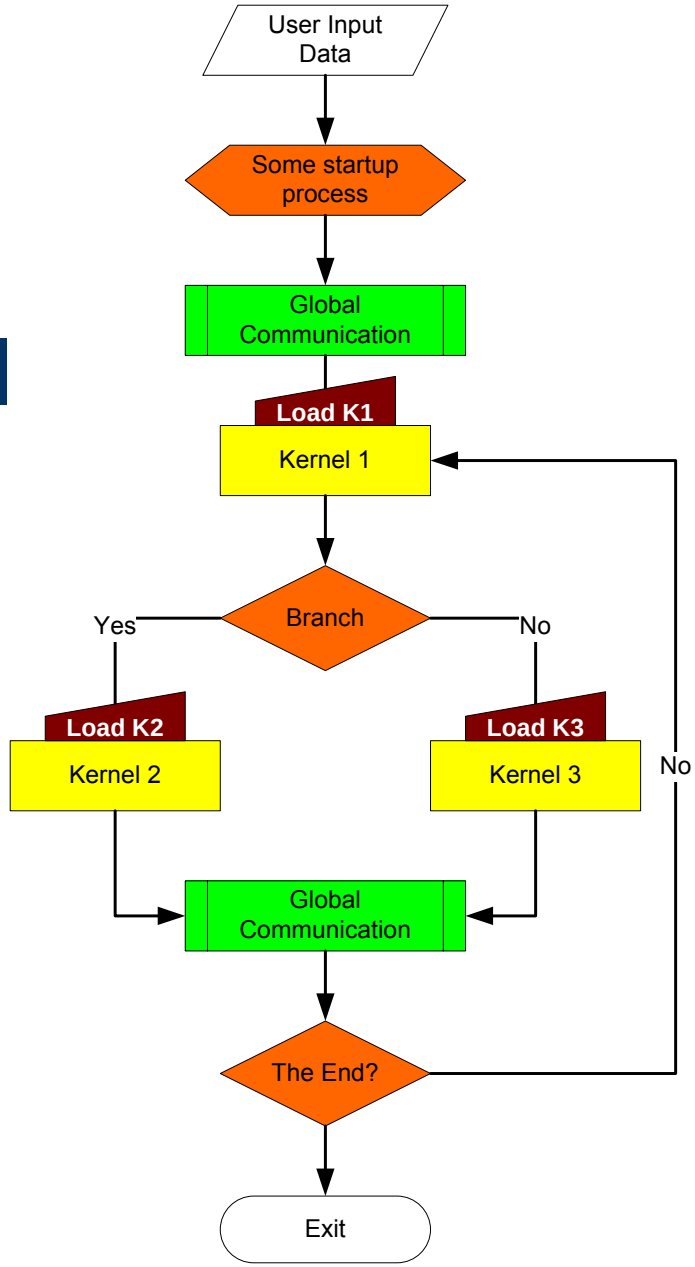
Individual Process

- Single process per PowerPC core (maximum 2 per FPGA)
- Kernels implemented on FPGA hardware fabric
- Global communication through MPI, and might be overlapped with computation



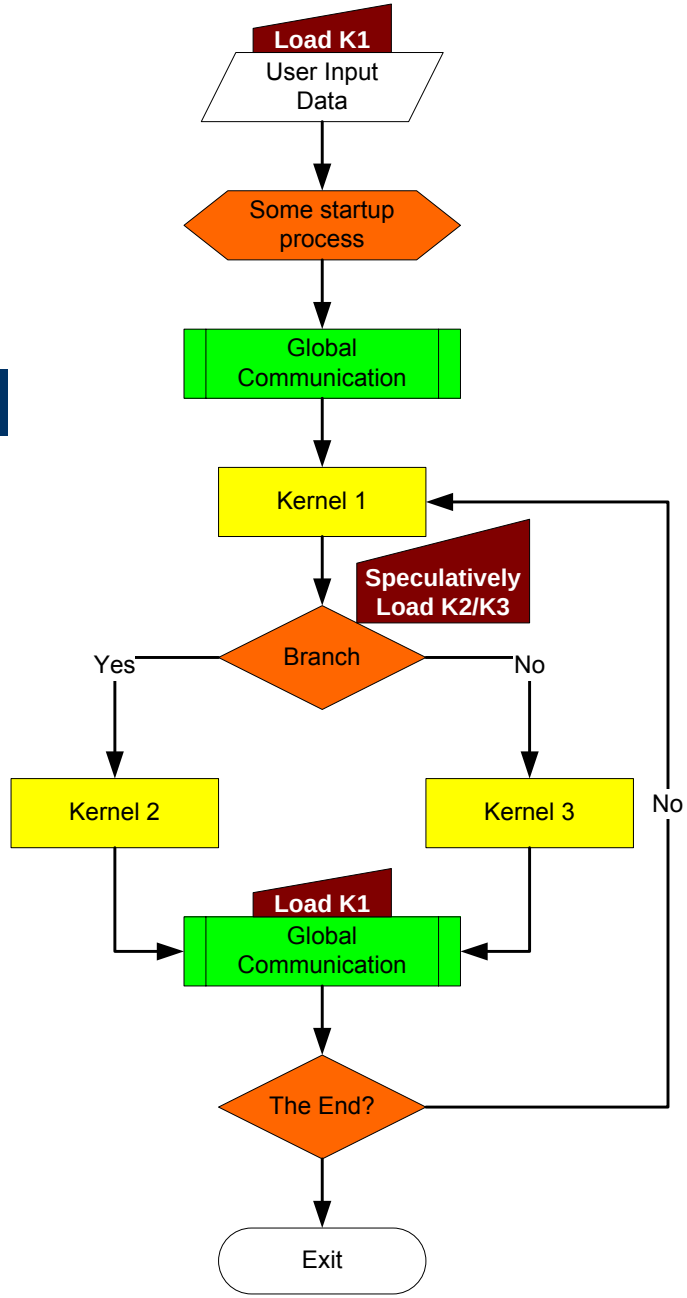
Dynamic Kernel Loading

- Naïve way: just-in-time (JIT) loading right before using the kernel
- Easy to implement
- Load latency of 25~50ms always visible
- Degrade performance



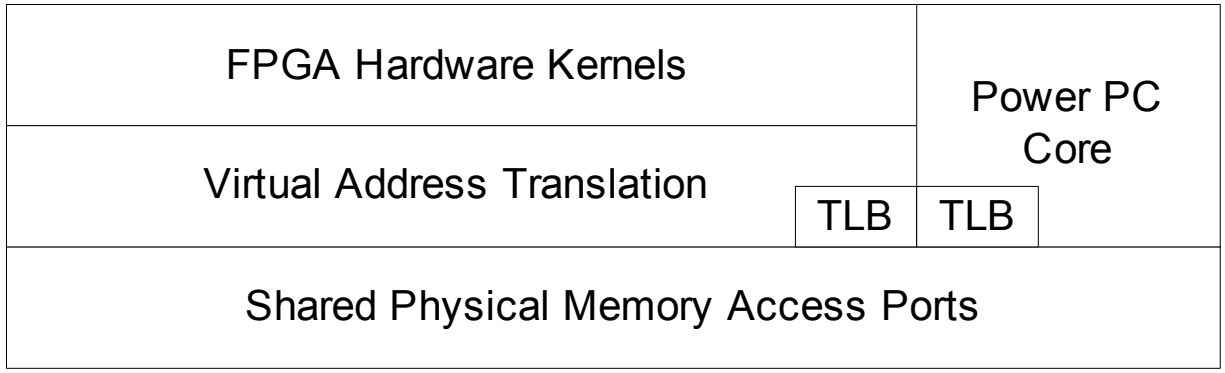
Dynamic Kernel Loading (cont.)

- Smarter way: pre-load as early as possible, so overlap load time with useful sequential computation and/or global communication
- Speculate when kernel choice depend on branch outcome
- Check if has the correct kernel before running it (kernel tagging)

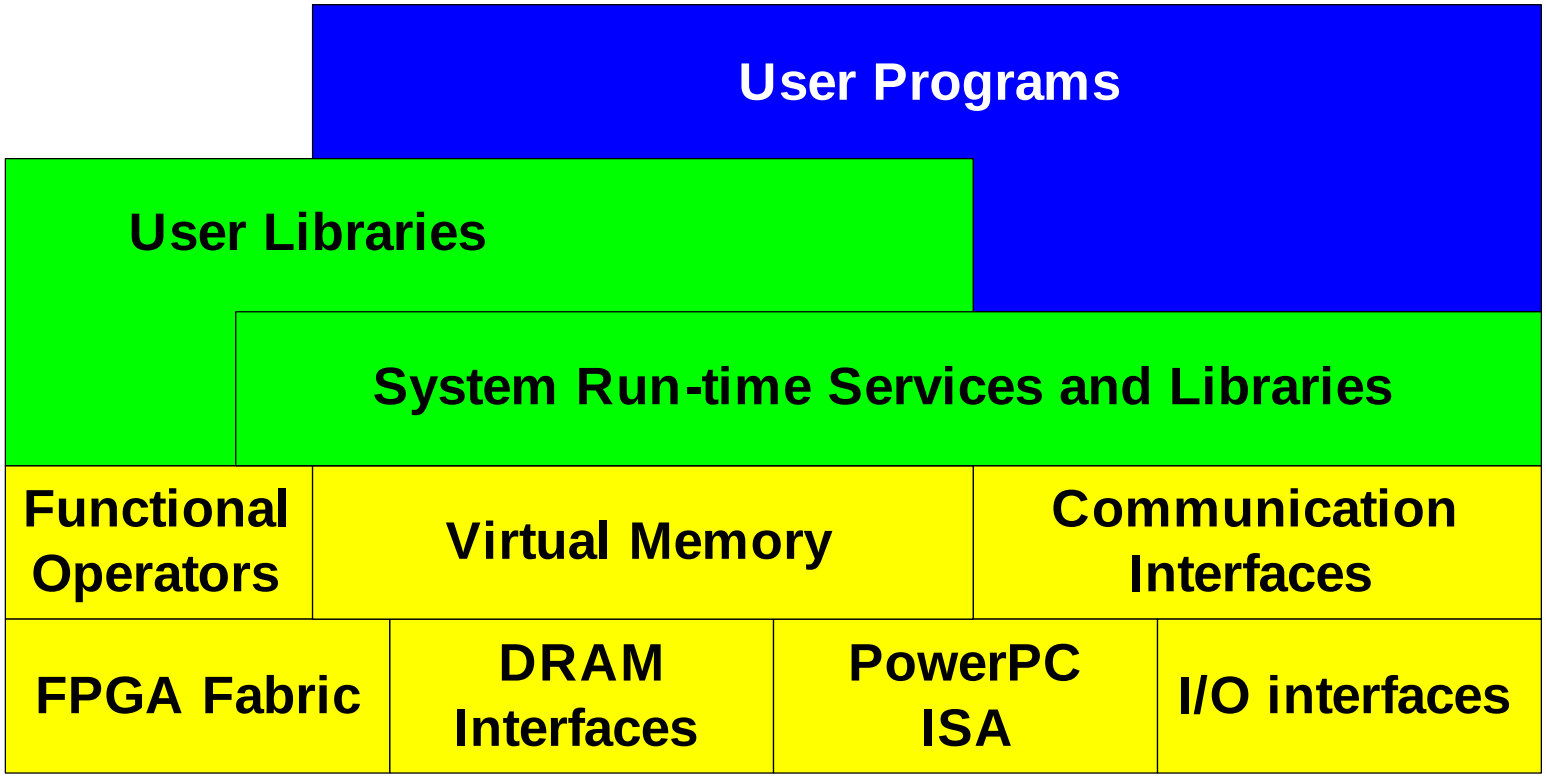


Memory Access Model

- Global distributed memory, communication between exclusive memory through MPI
- Local memory shared between the processor and the FPGA hardware kernel
- Virtual address needed for protection



Program Abstraction Layers



Operating System Overview

- Each computing node runs one copy of Linux on the control FPGA as the master, the rest processing FPGAs act as slaves
- A small micro kernel resides in the kernel memory space of each processing FPGA for program loading/unload and exception handling

User Access Management

- Work queue based multi-user environment running batch jobs on separate compute nodes
- Single compute node is the minimum job allocation unit
- Relative position of processes per FPGA can be specified on both intra and inter node levels

Resource allocation and protection

- FPGA resource allocated per user job, and spatially isolated from other user jobs
- Global communication and I/O resources are shared between jobs, and protected by the user access privilege table maintained by the global OS
- Local memory allocated through local OS, and protected through virtual address translation

Exception Handling

- Each hardware kernel has an exception status register (ESR)
- On error/faults:
  - stall pipeline, write error code to ESR
  - Release control to PowerPC, jump to exception handler
  - Exception handler choose to recover in-place, restart, or abort
- Page faults
  - On TLB miss, OS fixes the TLB in virtual memory manager.

Diagnostics and Debug

- Boot-time diagnostics
  - Memory and communication checks
  - OS and files system checks
- Run-time system monitors
  - Monitor each FPGA die temperature
  - Thermal shutdown capability
- Run-time debug

Error Detection and Correction

- Memory access
  - ECC for single-bit correction and double bit detection
- Global communication
  - Physical layer 8B/10B encoding
  - Link layer error detection through CRC field in packets
  - Transport layer reliable communication with retransmit

Amdahl's Law Modified

- Comparing BEE2 global performance to common micro-processor based cluster solutions
- Additional term is the sequential processing slow-down factor *S*
- The on-chip PowerPC core on FPGA is about 5~10 times slower than fastest sequential processors

$$Speedup = \frac{T_{sequential}}{T_{parallel}} = \lim_{P \to \infty} \frac{1}{\frac{(1-f)}{P} + f \cdot S} \approx \frac{1}{f \cdot S}$$

f: is the fraction of sequential code

P: is the parallelization factor

Performance Modeling

- Computation Time: $\frac{t_{seq}}{P} + 2(d-1)t_{ck}$
  - Sequential processing time: tseq
  - Pipeline latency: d
  - Hardware clock period: tck
  - Parallelization factor: P
- Communication Time: $\alpha + \beta \cdot m$
  - Message overhead: α
  - Network bandwidth: β
  - Message size: m

Performance Advantage over Clusters of Processors

- Fine grain data level parallelism found only on FPGA fabric
- Higher computation density leads to higher process-level parallelism
- Higher memory bandwidth and application specific data buffering schemes
- Reconfigurable hardware kernel libraries best matched to individual applications
- Efficient low-latency global communication API