

# **Towards Reconfigurable High Performance Computing based on Co-Design Concept (*Keynote Speech at HEART2014*)**

Taisuke Boku

Deputy Director, Center for Computational Sciences /  
Faculty of Systems and Information Engineering  
University of Tsukuba



# Outline

- Overview of today's HPC
- FPGA for HPC - past
- Co-design as key concept on FPGA for HPC
- Case study – TCA and PEACH2
- Conclusions



# Overview of today's HPC



# HPC – High Performance Computing

- Large (ultra) scale high end computing, mainly focusing on floating point calculation and high bandwidth/capacity of memory, network and storage
  - large scale scientific computing (computational science or engineering)
  - large scale data preservation and analysis
  - everything is high parallelized
- FLOPS (floating point operations per second) and bandwidth are essential
  - in many cases, double precision
  - bandwidth requirement is strongly affected by applications

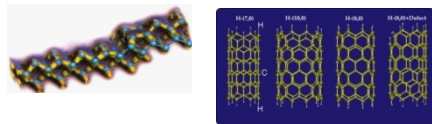


# Computational Science

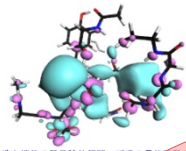
- In all fields of science, "simulation" covers "experiments" and "theory" ⇒ "computational science"

supercomputers

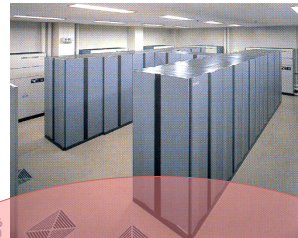
nano/material



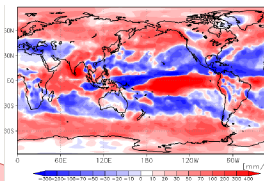
bio



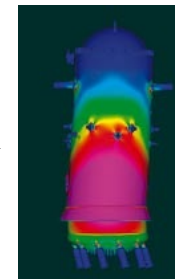
構造と機能の量子論的解明、呼吸の最終段階を司るシトクローム酸化酵素における電子移動量の正が



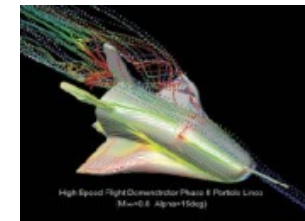
environment



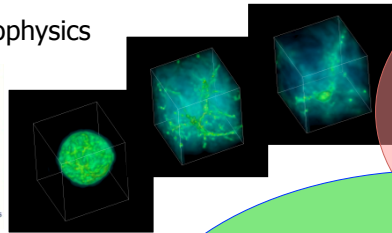
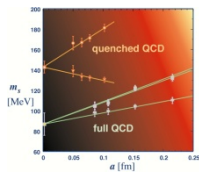
structure



fluid dynamics



particle/astrophysics

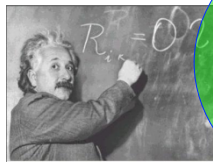


**Simulation**

**Computational Science**

**Theory**

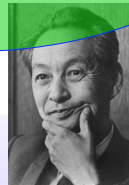
**Experiments/  
Observation**



theory of relativity



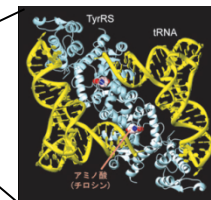
mesonic theory



renormalization theory



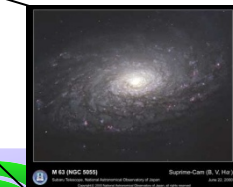
Spring-8



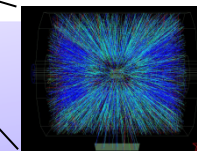
protein analysis



Subaru telescope



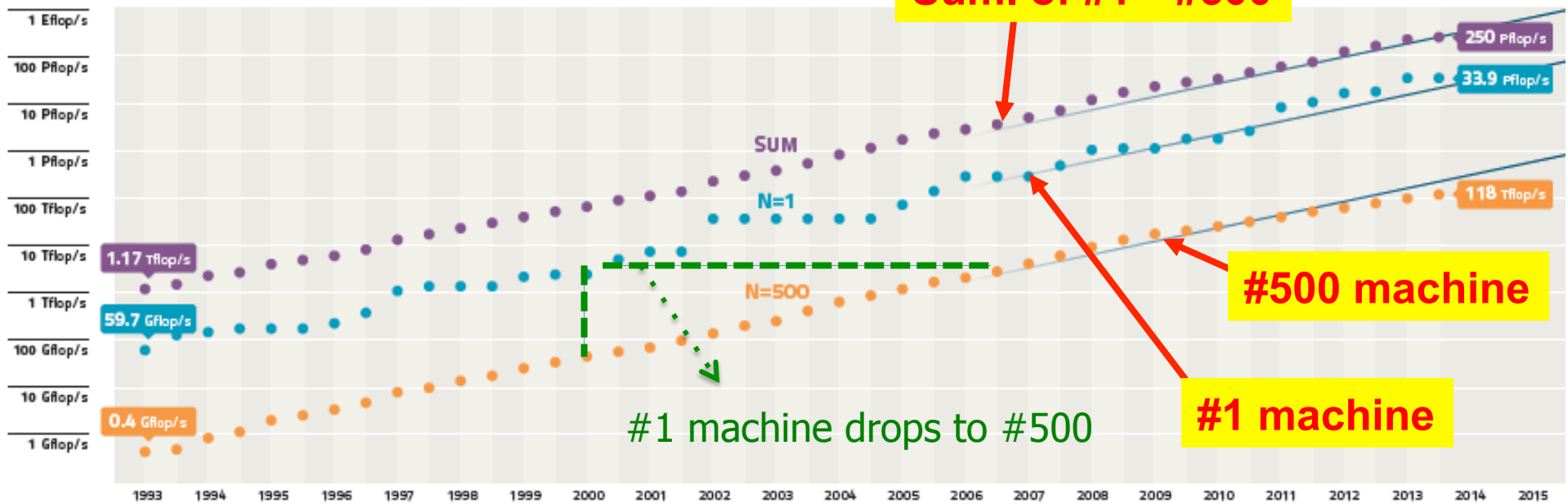
particle accelerator



# TOP500 List (Linpack: dense matrix equation solver)

www.top500.org

## PERFORMANCE DEVELOPMENT



	NAME	SPECS	SITE	COUNTRY	CORES	R <sub>MAX</sub> PFLOP/S	POWER MW
1	<b>Tianhe-2 (Milkyway-2)</b>	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NSCC Guangzhou	China	3,120,000	<b>33.9</b>	17.8
2	<b>Titan</b>	Cray XK7, Operon 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	<b>17.6</b>	8.2
3	<b>Sequoia</b>	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	<b>17.2</b>	7.9
4	<b>K computer</b>	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	<b>10.5</b>	12.7
5	<b>Mira</b>	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	<b>8.59</b>	3.95

# TOP10 machines in TOP500 (as on Nov. 2013)

Machine	Architecture	Country	Rmax (GFLOPS)	Rpeak (GFLOPS)	MFLOPS/W
<b>Tianhe-2 (MilkyWay-2)</b>	<b>Cluster (CPU + MIC)</b>	<b>China</b>	<b>33862700</b>	<b>54902400</b>	<b>1901.5</b>
<b>Titan</b>	<b>MPP (Cray XK7: CPU + GPU)</b>	<b>United States</b>	<b>17590000</b>	<b>27112550</b>	<b>2142.8</b>
<b>Sequoia</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>United States</b>	<b>17173224</b>	<b>20132659</b>	<b>2176.6</b>
<b>K Computer</b>	<b>MPP (Fujitsu)</b>	<b>Japan</b>	<b>10510000</b>	<b>11280384</b>	<b>830.2</b>
<b>Mira</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>United States</b>	<b>8586612</b>	<b>10066330</b>	<b>2176.6</b>
<b>Piz Daint</b>	<b>MPP (Cray XC30: CPU + GPU)</b>	<b>Switzerland</b>	<b>6271000</b>	<b>7788853</b>	<b>2697.2</b>
<b>Stampede</b>	<b>Cluster (CPU + MIC)</b>	<b>United States</b>	<b>5168110</b>	<b>8520112</b>	<b>1145.9</b>
<b>JUQUEEN</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>Germany</b>	<b>5008857</b>	<b>5872026</b>	<b>2176.8</b>
<b>Vulcan</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>United States</b>	<b>4293306</b>	<b>5033165</b>	<b>2177.1</b>
<b>SuperMUC</b>	<b>Cluster (CPU only)</b>	<b>Germany</b>	<b>2897000</b>	<b>3185050</b>	<b>846.4</b>



# TOP10 machines in TOP500 (as on Nov. 2013)

Machine	Architecture	Country	Rmax (GFLOPS)	Rpeak (GFLOPS)	MFLOPS/W
Tianhe-2 (MilkyWay-2)	Cluster (CPU + <b>MIC</b> )	China	33862700	54902400	1901.5
Titan	MPP (Cray XK7: CPU + <b>GPU</b> )	United States	17590000	27112550	2142.8
Sequoia	MPP (IBM BlueGene/Q)	United States	17173224	20132659	2176.6
K Computer	MPP (Fujitsu)	Japan	10510000	11280384	830.2
Mira	MPP (IBM BlueGene/Q)	United States	8586612	10066330	2176.6
Piz Daint	MPP (Cray XC30: CPU + <b>GPU</b> )	Switzerland	6271000	7788853	2697.2
Stampede	Cluster (CPU + <b>MIC</b> )	United States	5168110	8520112	1145.9
JUQUEEN	MPP (IBM BlueGene/Q)	Germany	5008857	5872026	2176.8
Vulcan	MPP (IBM BlueGene/Q)	United States	4293306	5033165	2177.1
SuperMUC	Cluster (CPU only)	Germany	2897000	3185050	846.4





# TOP10 machines in TOP500 (as on Nov. 2013)

Machine	Architecture	Country	Rmax (GFLOPS)	Rpeak (GFLOPS)	MFLOPS/W
Tianhe-2 (MilkyWay-2)	Cluster (CPU + MIC)	China	33862700	54902400	1901.5
Titan	MPP (Cray XK7: CPU + GPU)	United States	17590000	27112550	2142.8
Sequoia	MPP (IBM BlueGene/Q)	United States	17173224	20132659	2176.6
K Computer	MPP (Fujitsu)	Japan	10510000	11280384	830.2
Mira	MPP (IBM BlueGene/Q)	United States	8586612	10066330	2176.6
Piz Daint	MPP (Cray XC30: CPU + GPU)	Switzerland	6271000	7788853	2697.2
Stampede	Cluster (CPU + MIC)	United States	5168110	8520112	1145.9
JUQUEEN	MPP (IBM BlueGene/Q)	Germany	5008857	5872026	2176.8
Vulcan	MPP (IBM BlueGene/Q)	United States	4293306	5033165	2177.1
SuperMUC	Cluster (CPU only)	Germany	2897000	3185050	846.4



# TOP10 machines in TOP500 (as on Nov. 2013)

Machine	Architecture	Country	Rmax (GFLOPS)	Rpeak (GFLOPS)	MFLOPS/W
<b>Tianhe-2 (MilkyWay-2)</b>	<b>Cluster (CPU + MIC)</b>	<b>China</b>	<b>33862700</b>	<b>54902400</b>	<b>1901.5</b>
<b>Titan</b>	<b>MPP (Cray XK7: CPU + GPU)</b>	<b>United States</b>	<b>17590000</b>	<b>27112550</b>	<b>2142.8</b>
<b>Sequoia</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>United States</b>	<b>17173224</b>	<b>20132659</b>	<b>2176.6</b>
<b>K Computer</b>	<b>MPP (Fujitsu)</b>	<b>Japan</b>	<b>10510000</b>	<b>11280384</b>	<b>830.2</b>
<b>Mira</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>United States</b>	<b>8586612</b>	<b>10066330</b>	<b>2176.6</b>
<b>Piz Daint</b>	<b>MPP (Cray XC30: CPU + GPU)</b>	<b>Switzerland</b>	<b>6271000</b>	<b>7788853</b>	<b>2697.2</b>
<b>Stampede</b>	<b>Cluster (CPU + MIC)</b>	<b>United States</b>	<b>5168110</b>	<b>8520112</b>	<b>1145.9</b>
<b>JUQUEEN</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>Germany</b>	<b>5008857</b>	<b>5872026</b>	<b>2176.8</b>
<b>Vulcan</b>	<b>MPP (IBM BlueGene/Q)</b>	<b>United States</b>	<b>4293306</b>	<b>5033165</b>	<b>2177.1</b>
<b>SuperMUC</b>	<b>Cluster (CPU only)</b>	<b>Germany</b>	<b>2897000</b>	<b>3185050</b>	<b>846.4</b>



# Analysis

- #1, #2 and several machines are equipped with Accelerators (GPU or MIC)
  - MIC: Many Integrated Cores – Intel Xeon Phi
- IBM BG/Q is still very strong based on embedded multi-core processor (more power effective than K Computer's SPARC64 VIIIfx)  
-> but IBM will not make further BG system
- Accelerators provide high power efficiency
- Only one system in TOP10 as Cluster without accelerator (CPU only)
- Linpack (HPL) is a benchmark requiring "weak memory performance" (high cache hit ratio)



# Role of accelerators

- GPU (esp. NVIDIA Kepler architecture) is the most powerful computing resource today, although its sustained performance is lower than Linpack
- MIC (Xeon Phi) is a bit lower than GPU for perf./power, but much easier to program
  - > still need very careful tuning for performance
- Both GPU and MIC provide high memory bandwidth per device  
CAUTION: not so high per FLOPS
  - CPU: Intel Xeon E5-2670v2 = 0.3 Byte/FLOP
  - GPU: NVIDIA K20X = 0.2 Byte/FLOP
  - MIC: Intel Xeon Phi 5110P = 0.3 Byte/FLOP ??
  - > strong for computation bound applications



# What happens in near future ?

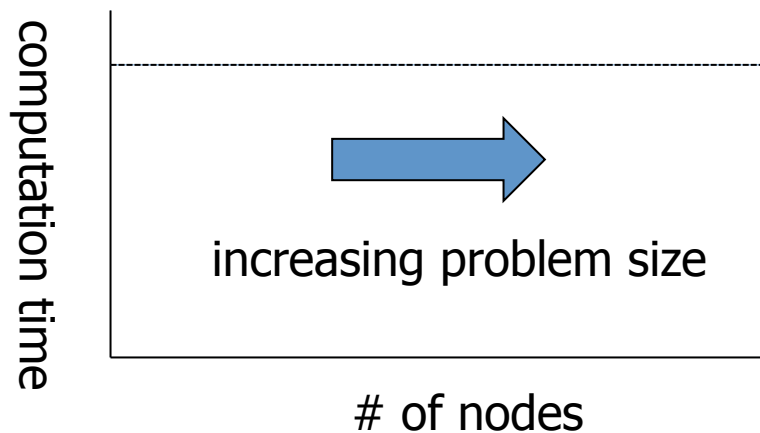
- “Exascale Systems” are expected to appear on year 2018-2020 according to TOP500 scaling
- Power limit is about 20~25 MW
  - 40~50 GF/W is required while today’s GPU provides 3GF/W
  - How to fill this gap of 15x ?
- Memory is the most essential issue
  - Bandwidth: we cannot keep 0.3 B/F in near future
    - > new memory technology like HBM or HMC partially saves
  - Capacity: we cannot keep 2GB/core anymore
    - > MIC 8GB/60core = 0.13GB/core
    - > *Strong Scaling* is essential



# Weak Scaling vs Strong Scaling

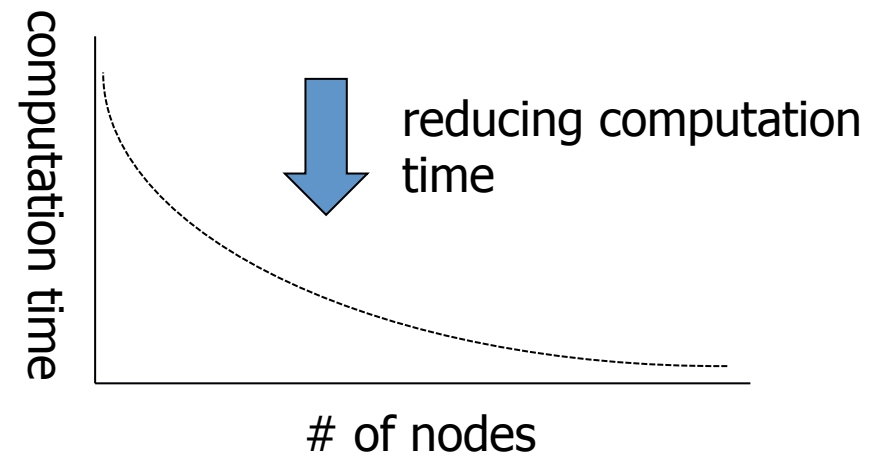
## ■ Weak Scaling

- fixed problem size per computation node
- increasing the number of nodes for larger problem size
- calculation time should be kept on each node



## ■ Strong Scaling

- fixed problem size per entire system
- increasing the number of node to shorten time-to-solution
- communication latency is crucial

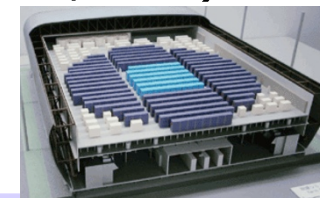


# Traditional FPGA with HPC



# Main players for high-end HPC

- late '70 ~ late '80
  - Vector machines dominated the world
  - Cray, NEC, Fujitsu, Hitachi (even IBM)
- early '90 ~ mid '90
  - dawn of "cluster" by killer micro, esp. RISC CPUs
  - MPP (massively parallel processor) including much of experimental machines
  - vector machines started to lose the power
- 2000 ~
  - MPP, cluster and heterogeneous system
  - vector machine got only once in #1 (Earth Simulator, Japan, NEC)
- 2010 ~
  - Cluster + accelerators





# FPGA for HPC (behind main players)

- Application specific solution
  - replacing ASIC with effectiveness/cost for designing
  - spatial repetition of simple computation
    - difference method
    - Lattice Boltzmann method
    - gravity calculation (N-body)
  - most of stencil computation can be solved with relatively small amount of data storage (register) on each computing module  
ex) [http://sacsis.hpcc.jp/2008/tutorial\\_sano.pdf](http://sacsis.hpcc.jp/2008/tutorial_sano.pdf) (Prof. K. Sano's tutorial at SAC SIS2008)
  - N-body is an ideal problem with perfect pipelining of single i-particle and stream of j-particles, also with very small ratio of I/O vs computation



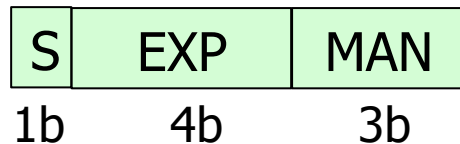
# Example of special purpose machine - GRAPE

- GRAPE (GRAVity PipE)
  - series of systems for N-body calculation started at Univ. of Tokyo (by D. Sugimoto), and continued at NAOJ (J. Makino) and RIKEN (M. Taiji)
  - It started with discrete TTL logic (GRAPE-1), then ASIC and FPGA (-> I have checked entire circuit of GRAPE-1 to help the team)
  - PROGRAPE-1 ~ PROGRAPE-4 (1999~2005) are based on FPGA (by T. Hamada)
  - GRAPE-7 is based on FPGA
  - GRAPE-6, GRAPE-8 use FPGA for glue chip
  - Gordon Bell Prize: 2001, 2003

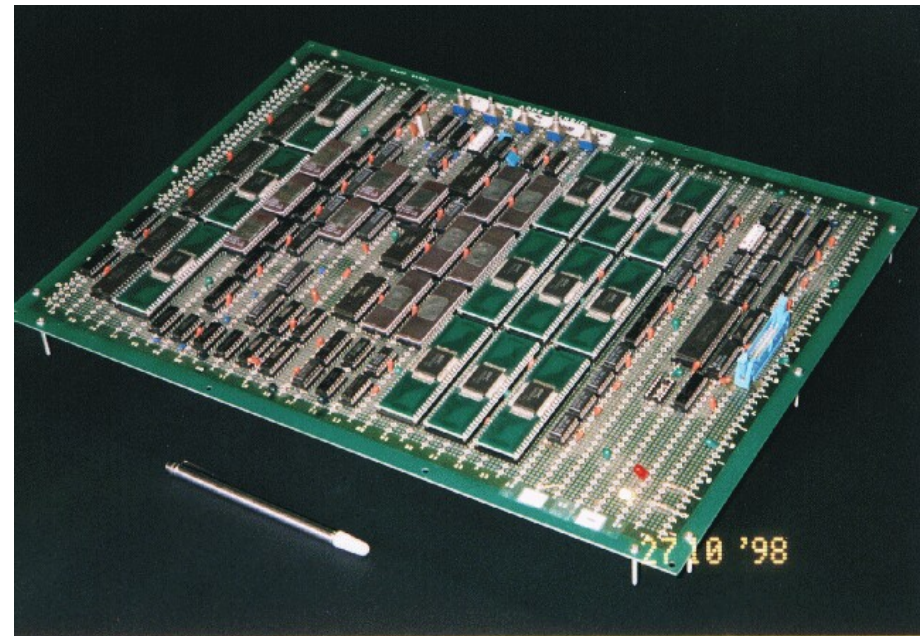


# Key issue – “precision control”

- GRAPE-1 ('89) has a strange handling of values  
-> “something completely different!” (for me)
- all the data (mass and coordinate) of particles (stars) are in “8-bit floating point”

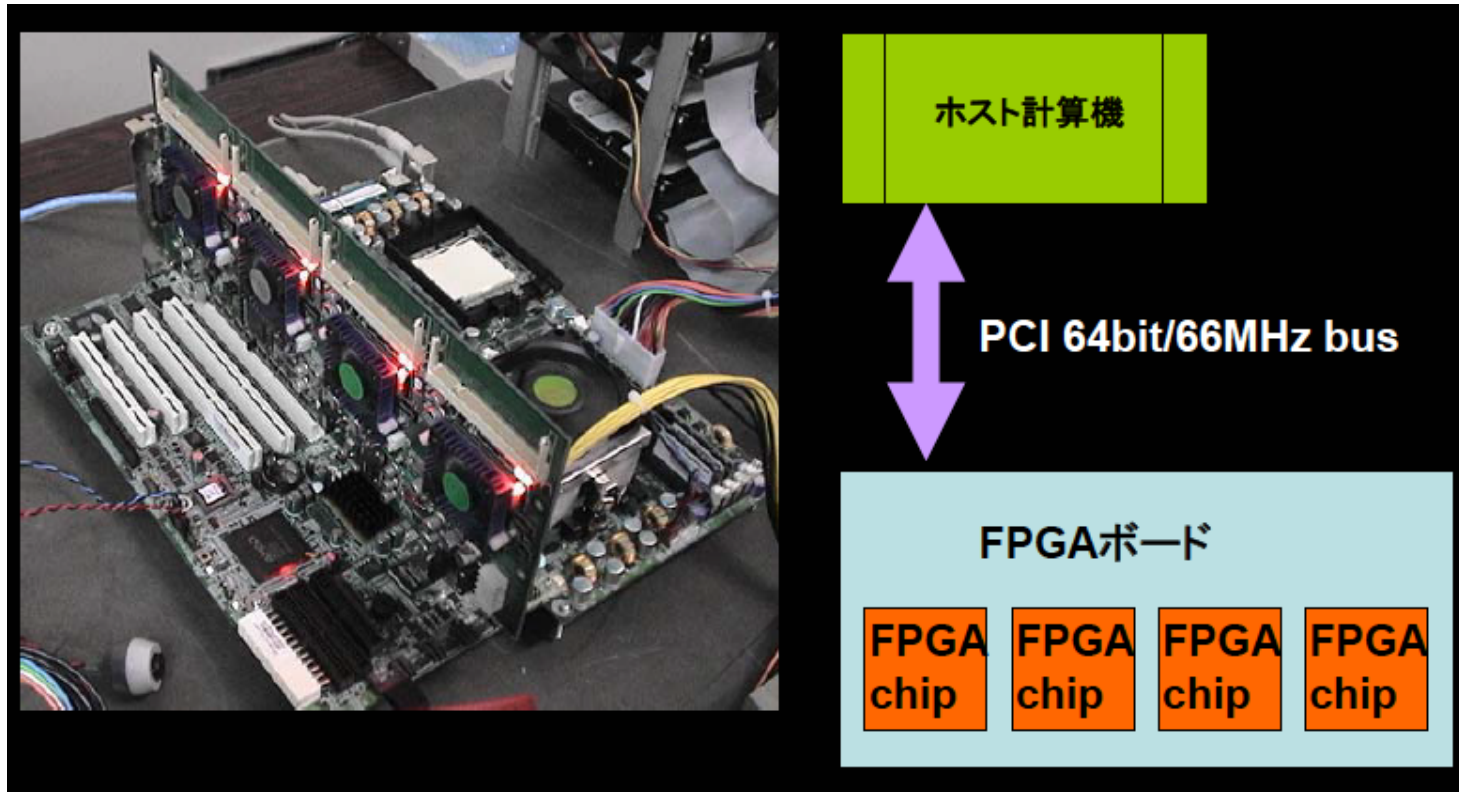


- > without calculation (16bit ROM table lookup instead of calculation)
- summation of force is stored in “48bit fixed point”
- it is enough for very dense star gravity calculation
- 1000x faster than 16-bit processor



GRAPE-1 board (from J. Makino's library)

# PROGRAPE-3 (by T. Hamada)



(from NAOJ talk by N. Nakasato)

- N-body & SPH (Smoothed Particle Hydrodynamics) calculation
- easy to modify the floating point components (EXP and MAN)
- 5x~10x faster than host processor

# CPU vs Special Purpose chip (in GRAPE)

- CPU
  - easy programming
  - cheap FLOPS and bandwidth by commodity
  - fixed format of value
  - high power due to complicated control (branch pred., shadow reg., large cache, etc.)
- Special chip (ASIC)
  - difficult to implement
  - high FLOPS but narrow bandwidth (memory)
  - variable format of value
  - low power to concentrate to computation with simple control

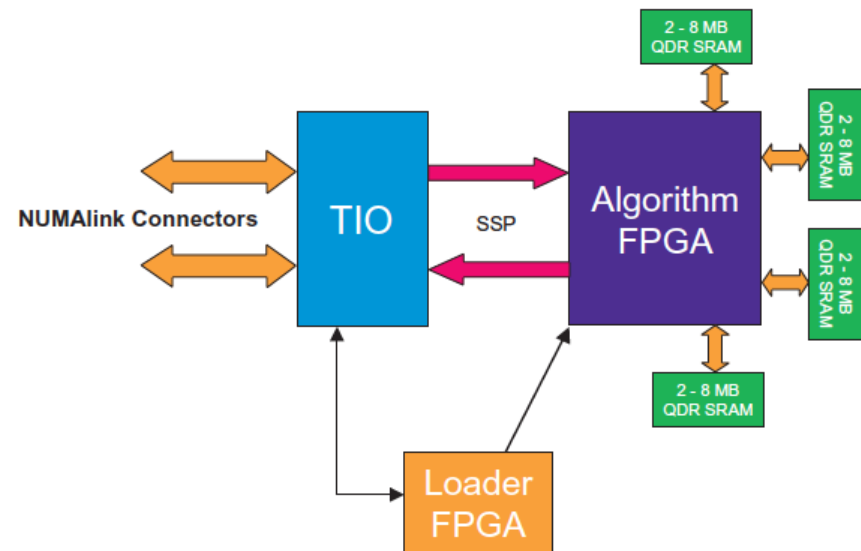


# FPGA-base commercial HPC systems (1)

- SGI Altix RASC technology
  - Reconfigurable Application Specific Computing
  - Xilinx Virtex-2 or 4 on each node
  - NUMA link is available for scalable parallel computing



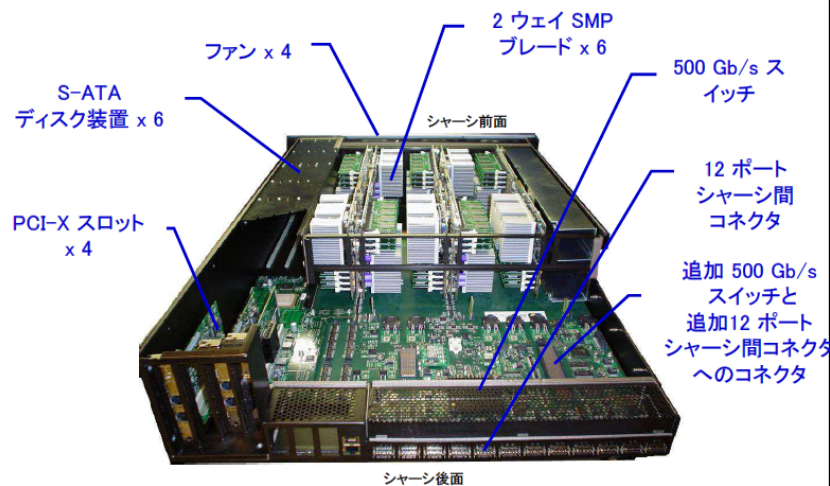
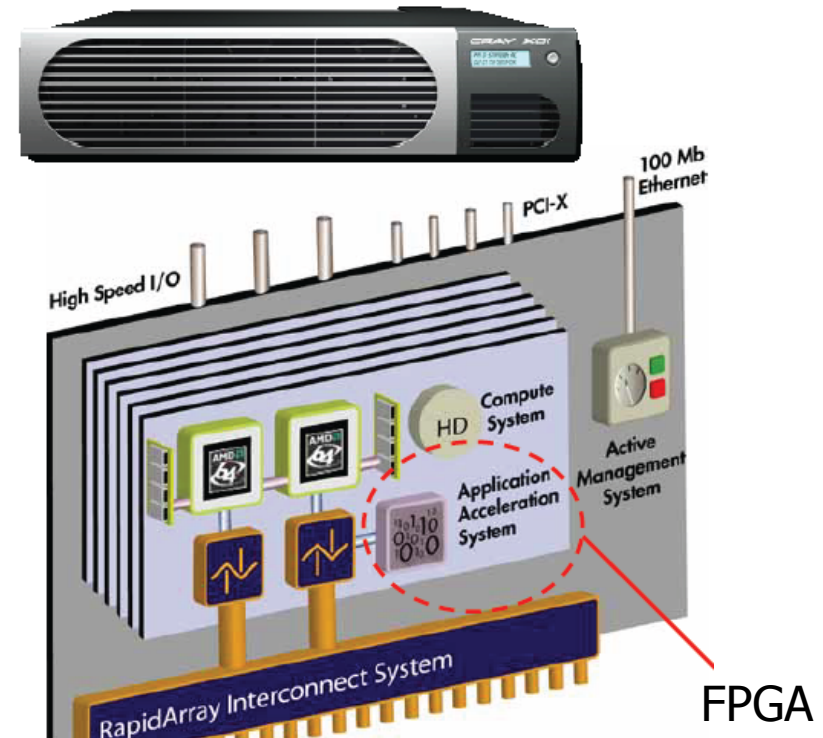
(from SGI home page)



# FPGA-base commercial HPC systems (2)

## ■ CRAY XD1

- computation node with 6 modules
- each module has 2 CPUs + FPGA (Application Acceleration System) Xilinx Virtex 4/II Pro
- RapidArray Interconnect for scalable parallel system



(courtesy by M. Nakano, Cray Inc.)

# What is CPU's Achilles tendon

- Too generic and too clever to carry out all the programmed code
  - “latency core” – specially tuned for sequential code
  - fixed format of values – generic but redundant
  - high power consumption not just for calculation
- from “Windows” to “Linpack”
  - not concentrated for calculation
  - floating point SIMD instruction saves partially
- Can we reach to Exascale ?  
-> quite difficult





# What is FPGA's Achilles tendon (for HPC)

- Slow frequency
  - absolute performance on “regular format” of calculation is weaker than CPU anymore
  - slightly older technology on silicon from CPU
- Memory
  - not so strong for memory bandwidth (inside/outside)
  - small storage (SRAM or register) on chip
- Hard to program/implement
- It is difficult to overcome CPU in conventional way, so how to do it ?



# Co-Design

## a key concept of FPGA for HPC



# Today's key word of HPC – Co-Design

- Definition of co-design in HPC (by DOE Co-design Center)
  - *"Co-design refers to a computer system design process where scientific problem requirements influence architecture design and technology and constraints inform formulation and design of algorithms and software."*
- Three Co-Design Centers
  - Exascale Co-Design Center for Materials in Extreme Environments (ExMatEx): LANL, LLNL, ORNL, SNL, Stanford, CalTech
  - Center for Exascale Simulation of Advanced Reactors (CESAR): ANL, U. Chicago
  - Center for Exascale Simulation of Combustion in Turbulence (ExaCT): LBNL, LLNL, ORNL, LANL, SNL



# Today's key word of HPC – Co-Design

- What co-design means
  - thinking from the target application and algorithm requirement for design of system and hardware
  - science/application driven design of the system
- Hardware is not a slave of software anymore
  - Conventionally, application users program the codes as they wish and think “hey, hardware guys, make it run fast!”  
ex) vector machine without changing basic code  
-> legend of “4 Byte/FLOP”
  - Facing to the hardware limit, software must “co-work” with hardware design
- Example
  - S->H: design for # of registers, limit of bandwidth, cache size, etc.
  - H->S: new algorithm to reduce bandwidth, registers, etc.



# FPGA - more aggressive player in co-design

- Currently co-design is recognized to seek the way for good compromise point between hardware and software
- In most of co-design concept, it is still considered within a general architecture
- FPGA is originally in co-design way
  - application specific
  - application is limited
  - all the design parameters follow the application requirement
- If HPC software is remodeled under co-design concept, there is a room for much more utilization of FPGA
- But brute-force computation is still difficult...



# Hint to glue FPGA and HPC

- Finding out where CPU or general (commodity) solution cannot reach
  - value formation control on each occupancy of partial computation
    - ex) QCD (Quantum Chromo Dynamics) requires “half precision” computation on preconditioning of CG
    - ex) GRAPE
  - very computation intensive body of code
    - ex) innermost loop calculation only on registers
    - ex) main body of stencil computation
  - FPGA can play role of glue as interface circuit
    - between CPU and interconnection network
    - between CPU and accelerators
    - etc.



# Case Study

## TCA and PEACH2



# Issues on accelerated computing in near future

- Limited memory capacity
  - current GPU = 6GB : 1.3TFlops = 1 : 200  
-> Co-design for memory capacity saving
- Limited dynamism on computation
  - on GPU, “warp splitting” makes heavy performance degradation -> depending on application/algorithm
  - but it should be solved by application/algorithm  
-> Co-design for effective vector feature
- Limited capacity on fast storage (register)
  - # of cores is large, but each core is very small  
-> Co-design for loop-level parallelism
- We need a large scale parallel system even based on accelerators
- Performance, bandwidth and capacity are covered by the way of co-design





# Issues on accelerated computing in future (cont'd)

- **Trading-off: Power vs Dynamism (Flexibility)**
  - fine grain individual cores consume much power, then ultra-wide SIMD feature to exploit maximum Flops is needed
- **Interconnection is a serious issue**
  - current accelerators are hosted by general CPU, and the system is not stand-alone
  - current accelerators are connected by some interface bus with CPU then interconnection
  - current accelerators are connected through network interface attached to the host CPU
- **Latency is essential (not just bandwidth)**
  - with the problem of memory capacity, "strong scaling" is required to solve the problems
  - "weak scaling" doesn't work in some case because of time to solution limit
  - in many algorithms, reduction of just a scalar value over millions of node is required



Accelerators must be tightly coupled with each other, meaning **"They should be equipped with communication facility of their own"**

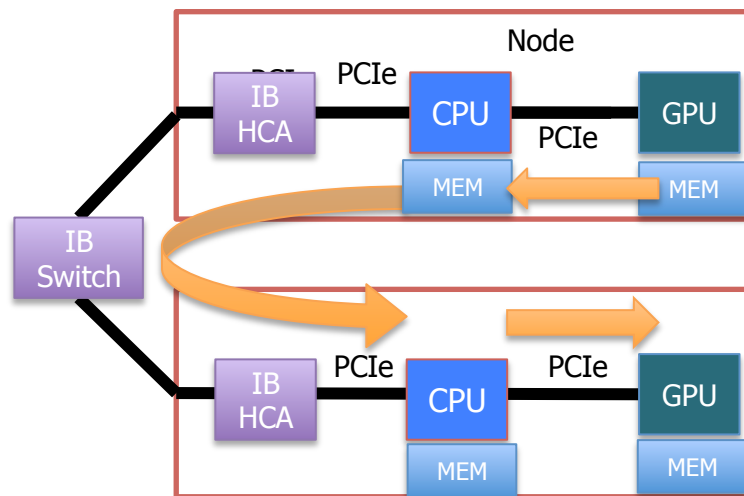


# TCA (Tightly Coupled Accelerators) Architecture

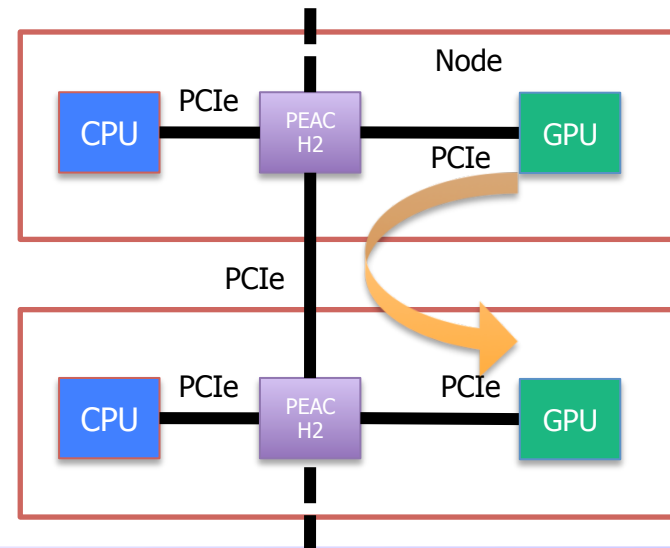


## ■ True GPU-direct

- current GPU clusters require 3-hop communication (3-5 times memory copy)
- For strong scaling, inter-GPU direct communication protocol is needed for lower latency and higher throughput

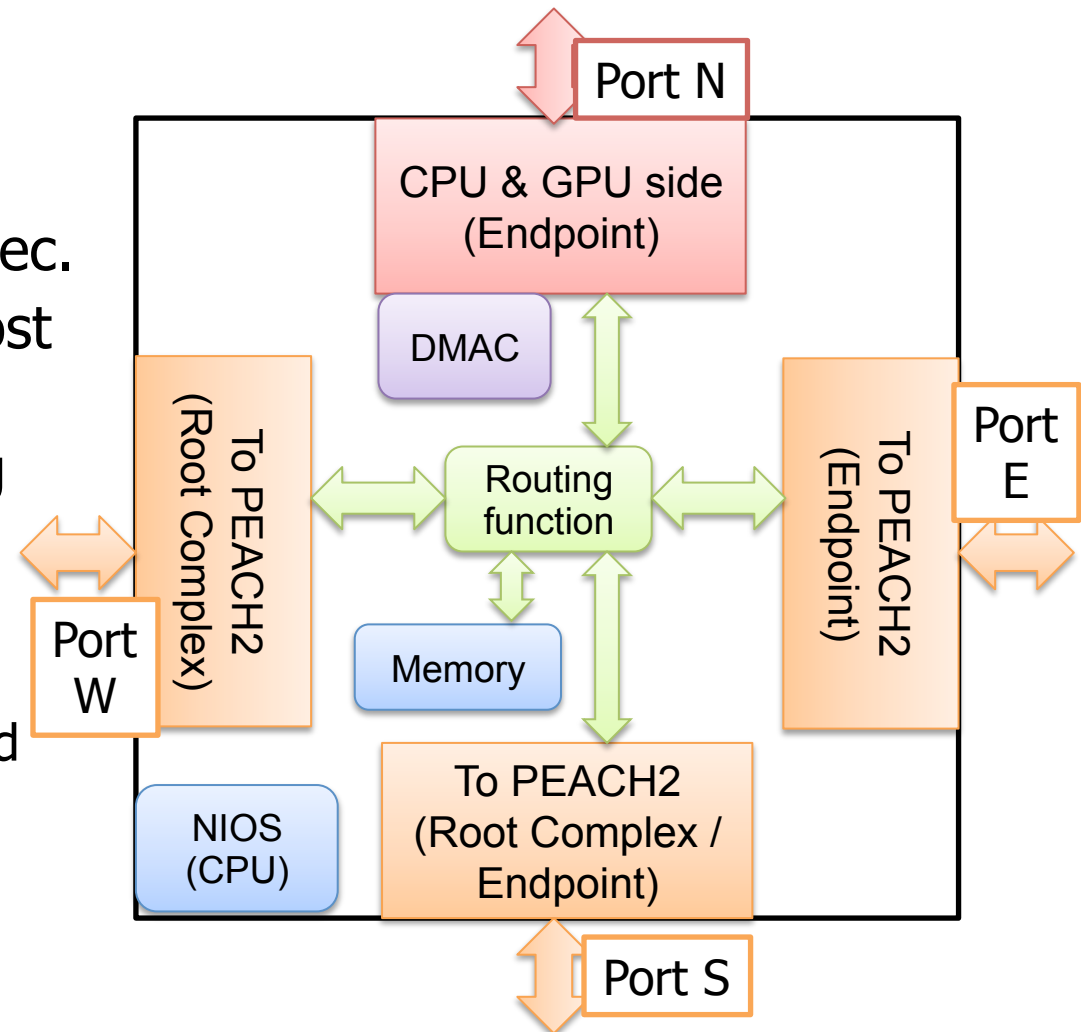


## ■ **PEACH2** *(PCI Express Advanced Communication Hub ver.2)*



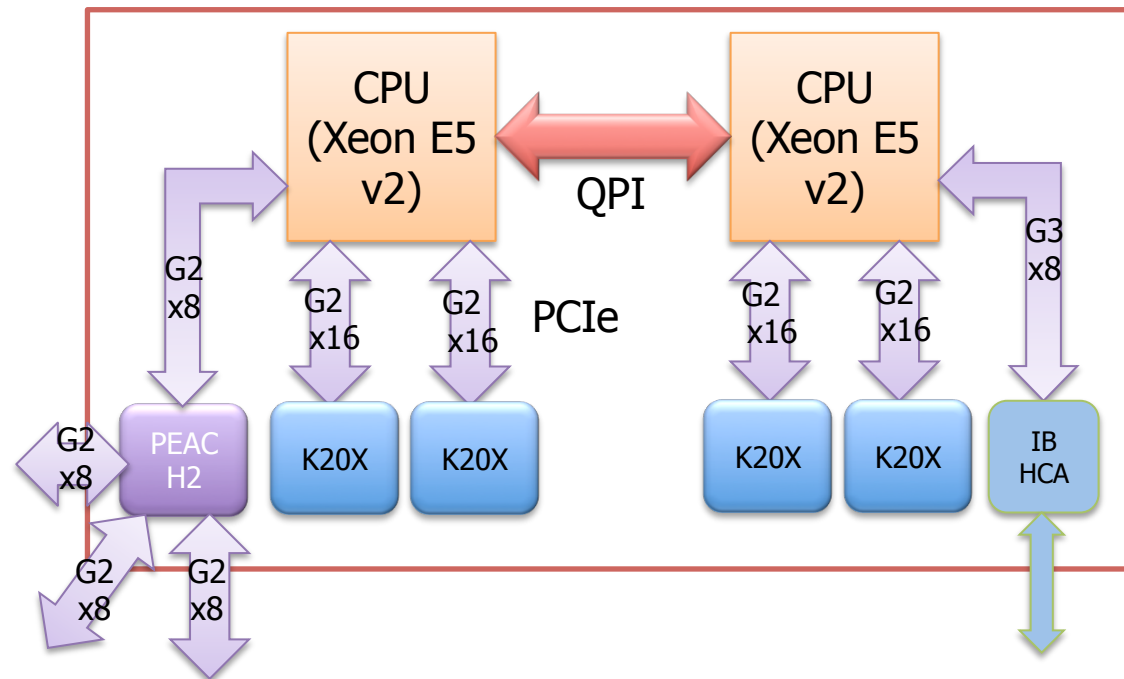
# Overview of PEACH2 chip

- Fully compatible with PCIe Gen2 spec.
- Root and EndPoint must be paired according to PCIe spec.
- **Port N**: connected to the host and GPUs
- **Port E and W**: form the ring topology
- **Port S**: connected to the other ring
  - Selectable between Root and Endpoint
- Write only except Port N
  - Instead, "Proxy write" on remote node realizes pseudo-read.



# TCA test-bed node structure

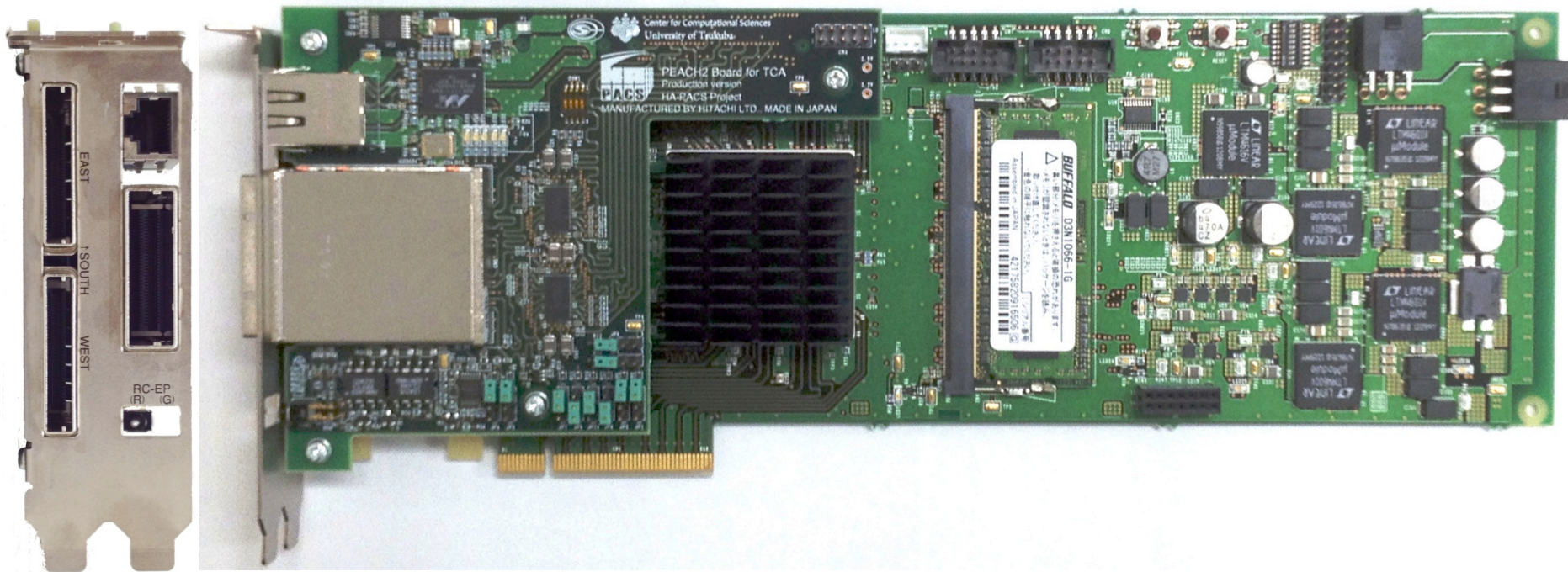
- CPU can uniformly access to GPUs.
- PEACH2 can access every GPUs
  - Kepler architecture + CUDA 5.0 "GPUDirect Support for RDMA"
  - Performance over QPI is quite bad.  
=> support only for two GPUs on the same socket
- Connect among 3 nodes
- This configuration is similar to HA-PACS base cluster except PEACH2.
  - All the PCIe lanes (80 lanes) embedded in CPUs are used.



# PEACH2 board



- PCI Express Gen2 x8 peripheral board
  - Compatible with PCIe Spec.

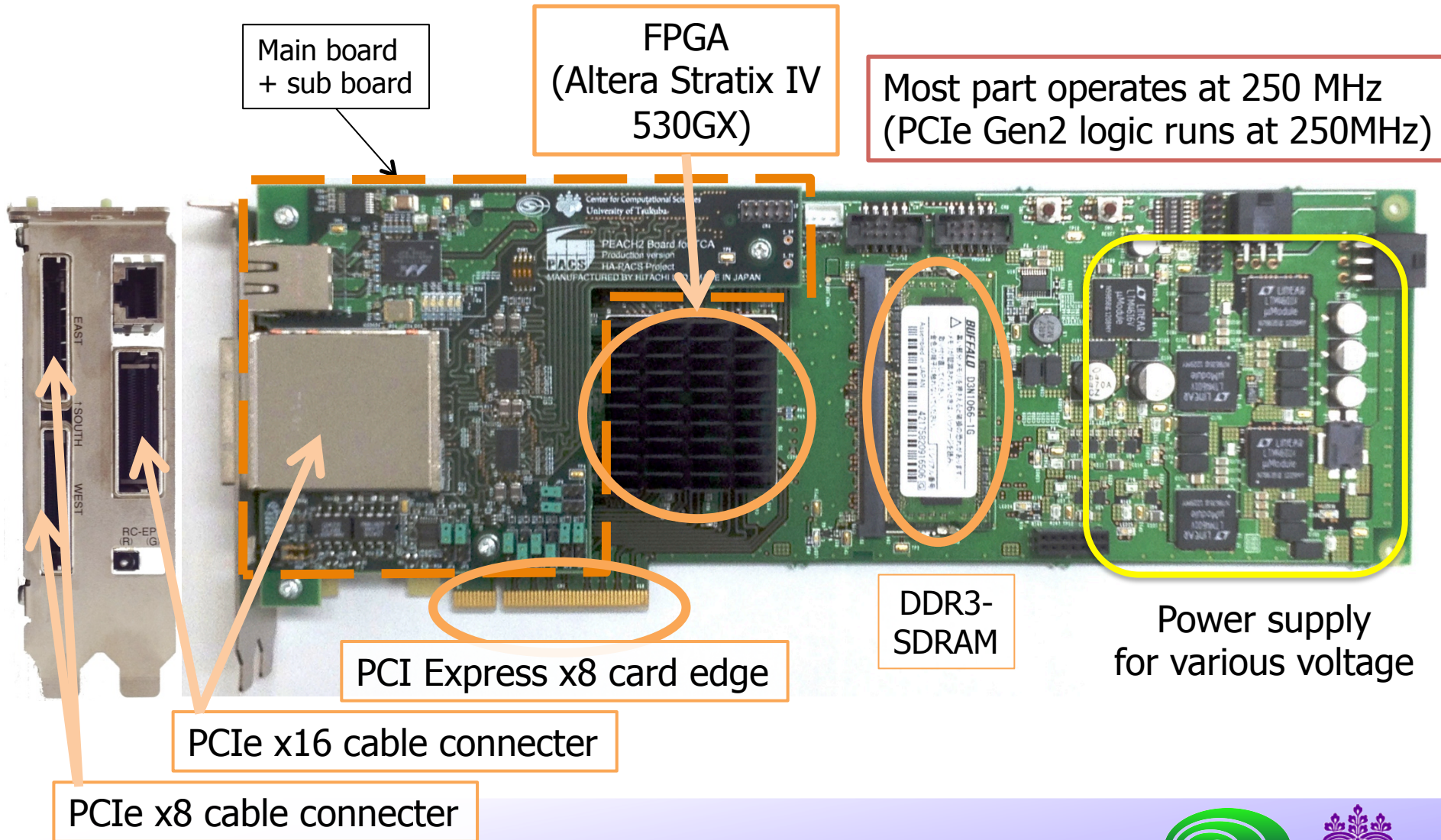


Side View

Top View



# PEACH2 board



# HA-PACS/TCA

- Practical test-bed for TCA architecture with advanced GPU cluster computation node with PEACH2 board and its network
- HA-PACS (Highly Accelerated Parallel Advanced System for Computational Sciences) project
  - Three year project for 2011-2013
  - Base cluster with commodity GPU cluster technology
  - TCA part for advanced experiment on TCA and PEACH2
- Base cluster part with 268 nodes
  - Intel SandyBridge CPU x 2 + NVIDIA M2090 (Fermi) x 4
  - dual rail InfiniBand QDR
- TCA part with 64 nodes
  - Intel IvyBridge CPU x 2 + NVIDIA K20X (Kepler) x 4
  - PEACH2 board is installed to all nodes and connected by its network (additionally to original InfiniBand QDR x 2)

# HA-PACS Base Cluster + TCA (TCA part starts operation on Nov. 1<sup>st</sup> 2013)



- HA-PACS Base Cluster = 2.99 TFlops x 268 node = 802 TFlops
- HA-PACS/TCA = 5.69 TFlops x 64 node = 364 TFlops
- TOTAL: 1.166 PFlops
- TCA part (individually) ranked as #3 in Green500, Nov. 2013



# HA-PACS/TCA computation node inside

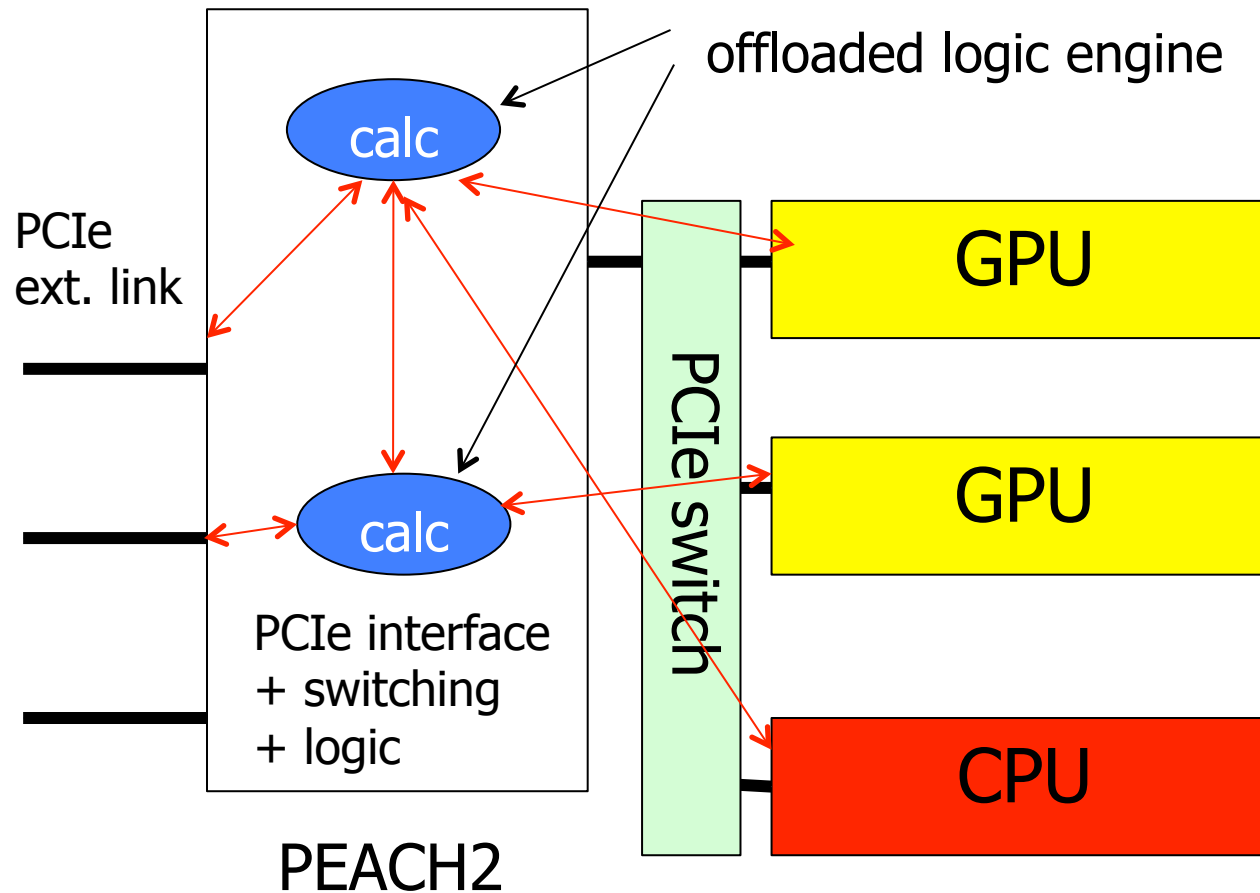


# How FPGA plays on TCA ?

- Communication glue to control PCIe without media conversion (not using InfiniBand)
  - All the accelerators today rely on PCIe to connect to host CPU
  - Overhead is minimized with PCIe communication only
- Off-loading several features not just for computation
  - Collective communication (with multiple nodes) with partial computation ex) global sum
  - Application specific computation around data communication without invoking CPU or GPU computing
- On strong-scaling problem, latency on node-node and CPU-GPU is crucial, and there are many of computation related to (surrounded by) communication exist



# TCA communication + offload image



TCA/PEACH2 detail is shown in talk by Y. Kodama and demonstration by H. Amano's group at HEART2014

# Example - Astrophysics



- Simulation of early galaxy construction
  - fluid dynamics
  - SPH (Smoothed Particle Hydrodynamics) for gas
  - gravity
- Fluid dynamics – in stencil computation, “sleeve” area computation can be offloaded to FPGA
- SPH – partial calculation on the data transferred from other nodes
- Relatively small amount and complicated computation which is not suitable for GPU – regularization, post Newtonian
- Ultimate solution – utilizing all the resources in “right men in the right places” manner for CPU, GPU and FPGA
- We will expand to use TCA concept for “Computational Astro-Biology” soon

# Conclusions

- Co-Design era is a big chance to use FPGA technology aggressively in HPC
- HPC is not just on “double precision” and it is seriously considered to utilize various form of floating point values
- To reduce the power at minimum satisfying computation requirement, we need “stoic” hardware/software co-designed system
- HPC people are facing to cliff and searching for solution not just based on conventional way (CPU, GPU, etc.)
- Let’s handshake with HPC researchers who are looking for something different solution



# Thank You !!

