The QPACE Project

T. Streuer

DESY

Technical Seminar 29.4.2008

Outline

QCD

Quantum Chromodynamics:
Theory describing strong interactions between quarks and gluons
$SU(3)$ gauge theory
Characteristic properties:

- Weak coupling at high energies
  $\longrightarrow$ Perturbation theory applicable
- Strong coupling at low energies
  $\longrightarrow$ Perturbation theory breaks down

Lattice QCD

Lattice QCD:
Only known first-principles method to compute low-energy QCD quantities

- Start from (Euclidean) functional integral formulation of QCD
- Replace continous 4d space-time by discrete lattice with
  - Finite lattice spacing a
  - Finite volume V

  $\Rightarrow$ Finite number of lattice sites $n = (V/a^4)$
  $\Rightarrow$ Functional integrals become ordinary finite-dimensional integrals

Typical values:
$a \approx 0.08\mathrm{fm}$, $V \approx (3\mathrm{fm})^4$
$\Rightarrow n = O(10^6)$

Lattice QCD

Fundamental calculations: integrals of the type

$$\int DU\, G(U) \ldots G(U) \det D$$

D: Lattice Dirac operator
G: Quark propagator
Integrate over all links in lattice
Only feasible method: Monte Carlo-integration

Lattice QCD

Well suited for parallelisation:

- Natural partitioning (domain decomposition)
- Only finite-range communication necessary
  Nearest neighbour comm. sufficient in many cases
- Computational kernels relatively simple

But: we need

- High bandwidth
- Low Latency

Lattice QCD machines

APE machines
"Array Processor Experiment"
Developed in Italy/Germany/France
APE(1989), APE100(1994), APEmille(2000), apeNEXT(2005)
apeNEXT:

- Custom VLIW processor
- 3d Torus Network

Installations: Bielefeld, Rome, Zeuthen

Lattice QCD machines

QCDOC
"QCD On a Chip"
Developed by US lattice community (+UKQCD) together with IBM

- Standard IBM PowerPC 440 CPU
- 6d Torus Network

Installations: Brookhaven, Edinburgh

Lattice QCD machines

Other approaches:

- PC Clusters
  (e.g. Tsukuba, JLAB, Wuppertal)
  - Moderate price/performance
  - Moderate scaling
  - Easy to program
- Graphic Cards
  Wuppertal, Budapest (Z.Fodor)
  - Superior price/performance
  - Hard to program
  - Only single-processor systems

QPACE

Qcd PArallel computing on CEll
QPACE collaboration:

- Academic partners:

  U Regensburg
  U Ferrara
  U Milano
  FZ Jülich
  U Wuppertal
  DESY Zeuthen

- Industrial partner:

  IBM Böblingen

QPACE Collaboration

Academic partners:

- U Regensburg
  S. Heybrock
  D. Hierl
  T. Maurer
  N. Meyer
  A. Schäfer
  S. Solbrig
  T. Wettig
- U Ferrara
  M. Pivanti
  F. Schifano
  L. Tripiccione
- U Milano
  A. Nobile
  H. Simma
- FZ Jülich
  M. Drochner
  N. Eicker
  T. Lippert
- DESY Zeuthen
  D. Pleiter
  T. Streuer
  K. Sulanke
  F. Winter
- U Wuppertal
  Z. Fodor

QPACE

Design Goals:
Build a lattice QCD machine with

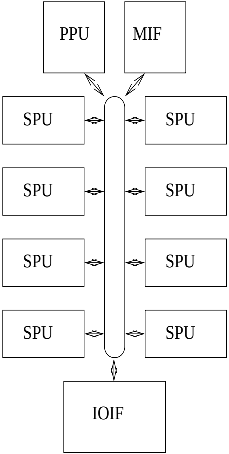
- Scalable network:
  - Low latency: $< 1\mu s$
  - Bandwidth: $1GB/s$ bidirectional
- Low power consumption

CPU: Cell Broadband Engine

- Developed by Sony, Toshiba, IBM ("STI") since 2001
- Basic idea:
  One general-purpose core
  Multiple specialized coprocessor cores
- Implementation:
  First version 2006
  "Enhanced Cell" 2008
- Applications:
  Playstation 3 (2006)
  IBM QS20 blades (2006)
  IBM Roadrunner (2008?)
  IBM QS22 blades (2008)

Cell Overview

- Power Processor Unit
- 8 Synergistic Processor Units
- Element Interconnect Bus
- IO InterFace
- Memory InterFace



Cell - PPU

PPU : Power Processor Unit
64-bit PowerPC compatible core

- Two hardware threads
  (separate register sets, shared execution units)
- AltiVec SIMD extensions (single precision)
- 2-Levels cache hierarchy (64 kB L1, 512kB L2)

Cell - SPU

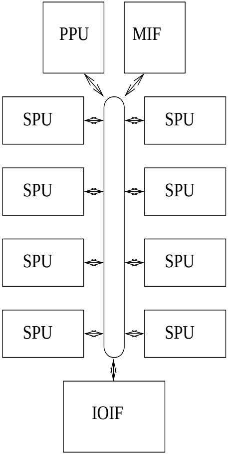
SPU: Synergistic Processor Unit

- Computing coprocessor
- RISC instruction set (different from PowerPC)
- 256kb local store (data+code)
- No RAM access, no cache
- SIMD instructions, operating on 128bit registers
- DMA engine for data transfer LS $\leftrightarrow$ RAM
- Peak performance:
  12.8 GFlop/s (double precision) at 3.2 GHz

Cell - EIB

Element interconnect bus:

- Internal bus connecting PPU, SPUs, IOIF, MIF
- Bi-directional Ring structure
- Total Bandwidth 200 GB/s
- Transfer Granularity: 128 Bytes (=1 PPU Cache line)



External interfaces

- RAM interface
  Interface to DDR2 RAM
- I/O interface
  RAMBUS “FlexIO”
  Max. bandwidth 25 GB/s inbound, 35 GB/s outbound
  Can be used to
  - Connect 2 Cells directly
  - Attach external device

Cell

Programming model:

- PPU: Linux kernel
  Management of SPUs via library calls
- SPU: no operating system
  Code execution controled by PPU
  OS services (I/O, paging, ...) provided by PPU
- Communication between PPU/SPUs:
  - DMA transfers
  - "Mailboxes"
  - Interrupts

Cell and Lattice QCD

Suitable for lattice QCD?

Programming model:

- PPU used for program control
- SPUs used for computation

Simple performance model:

- All tasks run at maximum throughput
- All latencies can be hidden

Maximum performance for Dirac operator:
34% of peak for reasonably-sized system

Limiting factor: RAM bandwidth

QPACE

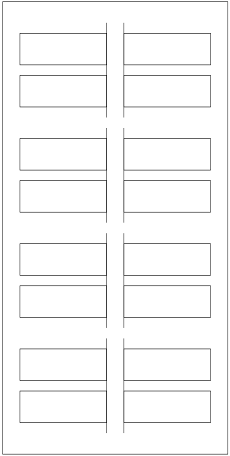
Machine overview

- O(2048) compute nodes
  Cell CPU
  4GB RAM
  Custum Network Processor
- Custom 3d torus network
  1GB/s bandwidth

QPACE Rack

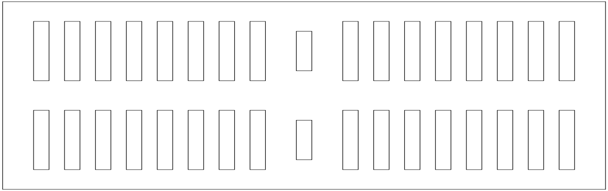
- 8 Backplanes in a rack
- 32 nodecards attached to each backplane

$\rightarrow$ 256 nodes/rack
$\approx$ 25 TFlop/s
(peak, double precision)



(Side view)

QPACE Backplane

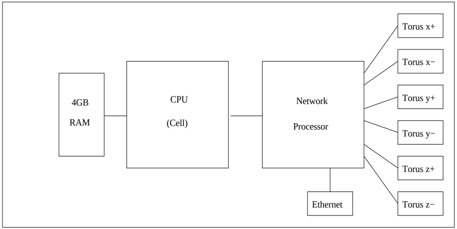


Each backplanes contains connectors for

- 32 Nodecards
- 2 Rootcards

Links in z-direction (“red”) on backplane
Connectors for Links in x,y-directions (“blue”, “green”)

QPACE Nodecard



- CPU: Cell BE
- 4GB Ram
- Custom Network Processor
- 6 Torus links
- Ethernet link

Network Processor

NWP implemented in FPGA ("Field Programmable Gate Array")

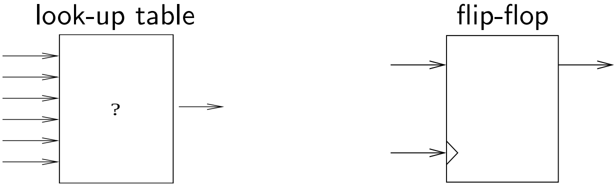
"gate array" = array of logic gates
"field programmable" : function of gates, connections between gates configurable (read from flash memory at power-on)

Large array of logic gates which can be re-programmed
Comparison to ASIC:

- (much) smaller NRE costs
- configurable
- higher per-unit costs
- lower performance

FPGA

Basic building blocks:



some other components:

- RAM, FIFO logic
- Fixed-point arithmetic units
- Clocking resources (PLLs, ...)
- I/O buffers

NWP

FPGA: Xilinx Virtex5-110LXT

- 550 MHz clock rates (in theory), for our design $\approx$ 300 Mhz
- 16 High-speed (3GHz) transceivers ("Rocket I/O")
- 666 kB RAM
- 69120 LUTs, flip-flops
- 680 I/O Pins

Network

3d Torus

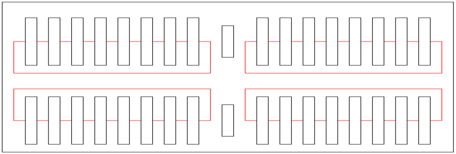
- "Red" links within backplane
  Max. Size 8
- "Green" links within half-rack
  Max. Size 16
- "Blue" links across racks
  Max. Size $2n_{\mathrm{rack}}$

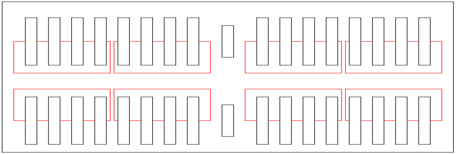
Switches on nodecard enable torus reconfiguration

Network configuration

"Red" links: within backplane

Two possible configurations:

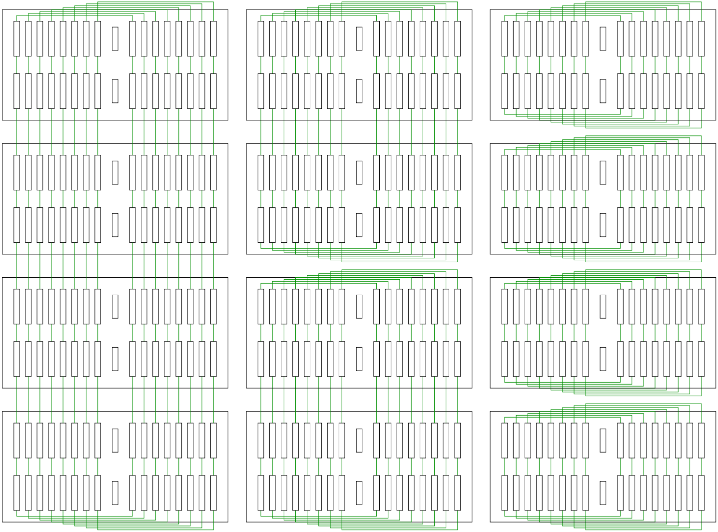




Network configuration

“Green” links: within (half-)rack

Three possible configurations:



Network Protocol

Low-level protocol: XAUI (10GbE)

- 4 bi-directional lanes at 2.5 GHz
- Differential signaling
- No separate clock transmitted
- 8b/10b encoding:
  10 bits transmitted for each data byte
  eases clock recovery
  some level of error protection

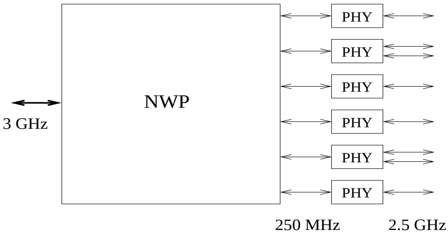
$\rightarrow$ 10 GBit/sec "raw" data, 1 GByte/sec usable data

Network Protocol

High-level protocol: custom protocol

- Only nearest neighbour communication
- Four virtual channels per link
- Packet order preserved within channel
- Data packets:
  - 128 Bytes payload
  - Header (Target address, channel)
  - CRC checksum

Network



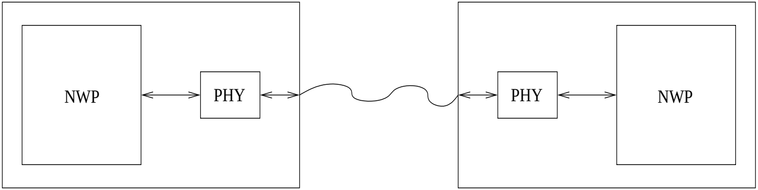
Network processor:

- routing between Cell and torus links
- data buffering

GBe PHYs:

- serializing/deserializing
- 8b/10b encoding
- switches to reconfigure topology

Link Testbed

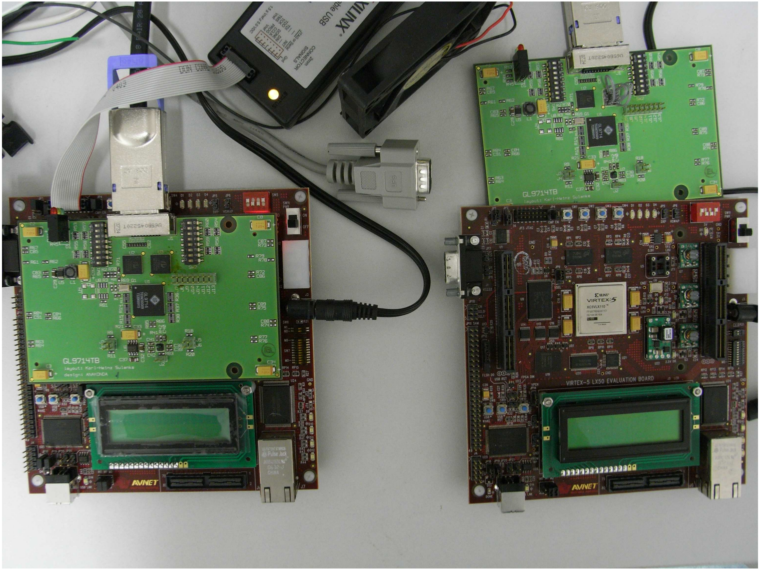


Testbed for torus link:
Two testboards with NWP and one PCIe PHY

- 1 GB/s data rate
- 50cm/3m infiniband cable
- $> 24\mathrm{h}$ without bit error (O(100TB))

But:
have to replace PHY because of area, power consumption
$\Rightarrow$ will repeat test with different PHY

Link Testbed



Rootcard

1 Rootcard per 16 Nodecards

- Controls power-on
- Switch for control/monitoring signals
- Clock generation/distribution:
  - Each rootcard contains clock generator (25 MHz)
  - Only one is active
  - All nodes run on same frequency
  - Input clocks for Cell, NWP, ... derived from global clock on nodecard
- Collects/distributes global signals

Global Signal Tree

- Tree network
- Two bits per direction (up/down)
- Independent of torus network
- Functions:
  - Node synchronization
  - "Kill" on fatal error
  - Evaluate global conditions

Communication model

User's view of network:

- Simple communication library (no MPI)
- Expected usage:
  SPMD (single program, multiple data)
  Matching send, receive commands

Project status

- Nodecard/Rootcard schematics: April 2008
- Nodecard Power-on: May 2008
- Bring-up prototype (nodecard, rootcard, backplane): July 2008
- Small test system (32 nodes): End 2008
- Large System (2x1024 nodes): Spring 2009