The Technological Frontier in Lattice QCD: Cost Optimized Machines

Thomas Lippert and John W. Negele

Santorini Workshop on Advanced Computing in Nuclear and Hadronic physics

October 1, 2001

Outline

Cost-Optimized Machines for Lattice QCD

Clusters

- Motivation
- Wuppertal / NIC Clusters
- MIT / Jlab Clusters
- Future prospects

Custom Parallel Machines

- QCDSP
- QCDOC
- APE mille

Resources and plans

Cost-Optimized Custom Machines for Lattice QCD

- Highly parallel custom machines much cheaper than general purpose supercomputers

  regular grid structure

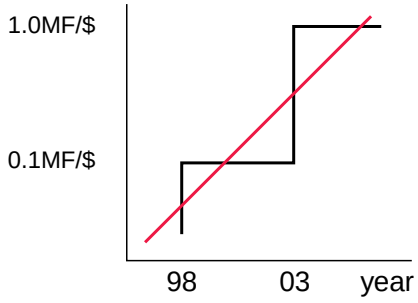
  local communications

  overlapping computation and communications

- Dual approach

  optimization of commodity clusters

  fully custom parallel machine



- Robust strategy to pursue both

Motivation for Commodity Based Cluster

- General-purpose architecture and software environment

  Linux operating system, standard compilers

  Source code compatibility with workstations

  Flexibility, ability to implement innovative new approaches

  Efficient use by everyone in community

  Ideal for local development by dispersed collaborators

- Double-precision needed for many applications
- Commodity processors and networks offer maximal flexibility

  Follow best technology in each generation

  Purchase state-of-art technology any year
  physics or funding motivate it

- Cost-effective:

  Market forces on processors and communications

  Processor chip price/performance improving
  like Moore's Law

  System integration costs falling

  Large cluster improvement better than Moore's Law

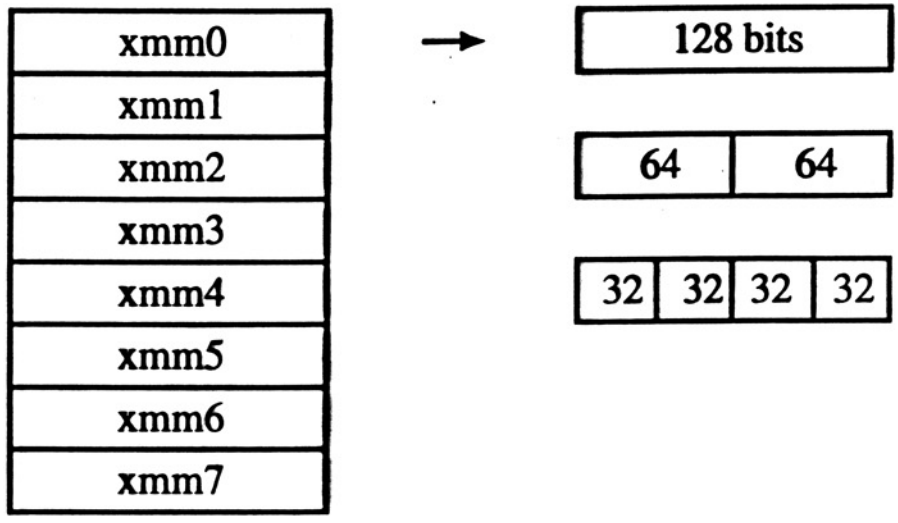
- Current technology: Alpha $\rightarrow$ Pentium P4
- Next generation: computer on a chip?

Advanced features of current PC processors

- Vector arithmetic (SSE, 3DNow!, AltiVec)
- On-die cache @ processor clock speed
- Memory-to-cache prefetch
- Streaming memory access

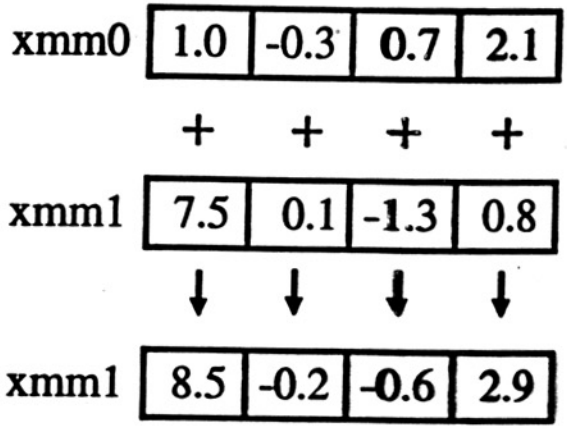
Streaming SIMD Extension (SSE)

- Supported on PIII, P4, Itanium, AMD Sledgehammer
- 8 additional registers for 4 floats or 2 doubles



- SIMD instructions operating on these
- Cache manipulation instructions

SIMD instructions



* Also `subps`, `mulps`, `divps`, `sqrtps`
* IEEE-754 compliant arithmetic
* Instructions for data moving & shuffling

Programming example

Consider the C code

```
for (i=0;i<100;i++)
  a[i]=b[i]+c[i];
```

Using SSE instructions (and GCC) this becomes

```
for (i=0;i<100;i+=4)
  __asm__ __volatile__ (
              "movaps %1, %%xmm0 \n\t"
              "movaps %2, %%xmm1 \n\t"
              "addps %%xmm0, %%xmm1 \n\t"
              "movaps %%xmm1, %0"
              :
              "=m" (a[i])
              :
              "m" (b[i]),
              "m" (c[i]));
```

SU(3) matrix × vector multiplication

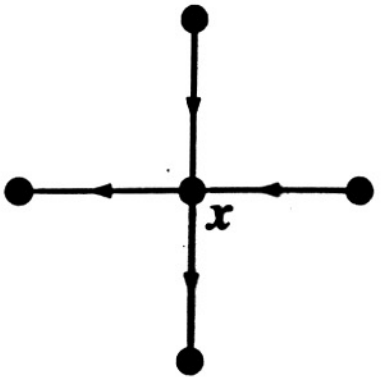
```
#define _sse_su3_multiply(u)
  __asm__ __volatile__ (
          "movss %0, %%xmm3 \n\t"
          "movss %1, %%xmm6 \n\t"
          "movss %2, %%xmm4 \n\t"
          "movss %3, %%xmm7 \n\t"
          "movss %4, %%xmm5 \n\t"
          "shufps $0x0, %%xmm3, %%xmm3 \n\t"
          "shufps $0x0, %%xmm6, %%xmm6 \n\t"
          "shufps $0x0, %%xmm4, %%xmm4 \n\t"
          "mulps %%xmm0, %%xmm3 \n\t"
          "shufps $0x0, %%xmm7, %%xmm7 \n\t"
          "mulps %%xmm1, %%xmm6 \n\t"
          "shufps $0x0, %%xmm5, %%xmm5 \n\t"
          "mulps %%xmm0, %%xmm4 \n\t"
          "addps %%xmm6, %%xmm3 \n\t"
          "mulps %%xmm2, %%xmm7 \n\t"
          "mulps %%xmm0, %%xmm5 \n\t"
          "addps %%xmm7, %%xmm4 \n\t"
           :                    :
          "addps %%xmm6, %%xmm4 \n\t"
          "addps %%xmm7, %%xmm5"
          :
          :
          "m" ((u).c11.re),
          "m" ((u).c12.re),
          "m" ((u).c21.re),
          "m" ((u).c23.re),
           :                    :
```

On the P4 this code achieves 2.6 Gflop/s [1.8 flop/cycle] !

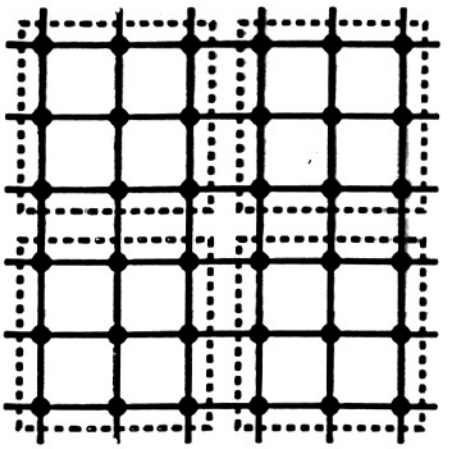
Cache management

Strip-mining

Computation of $(D_{\mathrm{w}} + m_0)\psi(x)$



$\Rightarrow$ data are reused many times

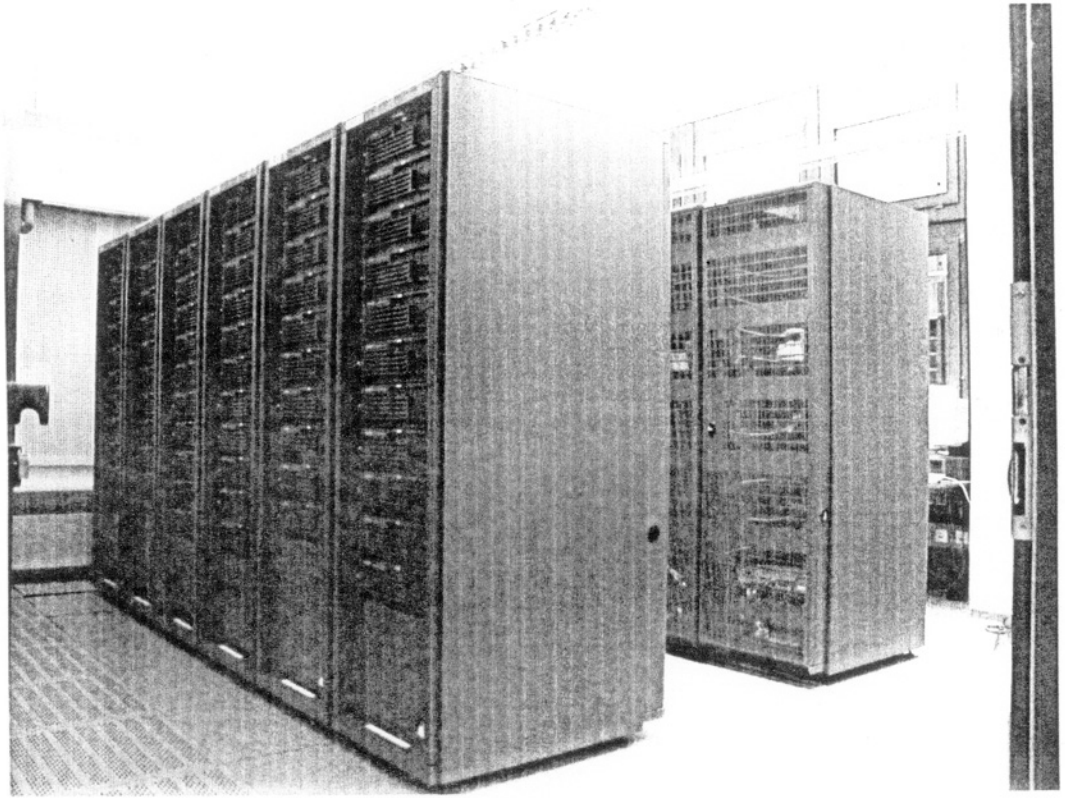


$\Rightarrow$ data blocking enhances the cache-hit probability

PC clusters for LQCD *(incomplete)*

institution	processors	# nodes	status
Fermilab	2×PIII 0.7 GHz	80	running
TC Dublin	2×PIII 1.0 GHz	32	running
MPI Munich	2×PIII 1.0 GHz	8	ordered
DESY-Zeuthen	P4	16	asking for bids
DESY-Hamburg	P4	32	approved
Rome II & CERN	P4	2×64	fast network development
Fermilab	P4	~ 150	approved
Jlab & MIT	2×P4	128	approved

ALiCE / Alpha Linux Cluster Engine



MIT Prototype Cluster

Joint research project with Compaq

Optimize network and memory costs using 4-processor nodes connected by Myrinet

- 12 $\times$ 4-processor ES40's – 64 Gflops

  EV67 667 MHz processors

  8 MB cache/processor

  1 GB memory/SMP

  9 GB disk

- 500 GB file server
- 4 Alpha 164 workstations
- Myrinet (Lanai 7)
- Installed 12/99–8/00

Jefferson Lab Prototype Clusters

- 16 × 1-processor XP1000's – 16 Gflops

  500 MHz

  4 MB cache

  256–512 MB memory

- 12 × 2-processor UP2000's – 32 Gflops

  667 MHz

  4 MB cache

  512 MB memory / SMP

- 400 GB file server
- Myrinet

  Lanai 9

QCDSP

CUSTOM MACHINE DESIGNED AT COLUMBIA

PROCESSOR 50 Mflops TI DSP

2 MB memory / processor

25 Mbit/sec communications each way to nearest neighbors

SINGLE PRECISION

COLUMBIA MACHINE

8192 NODES

400 GFLOPS PEAK

120 GFLOPS SUSTAINED > 30%

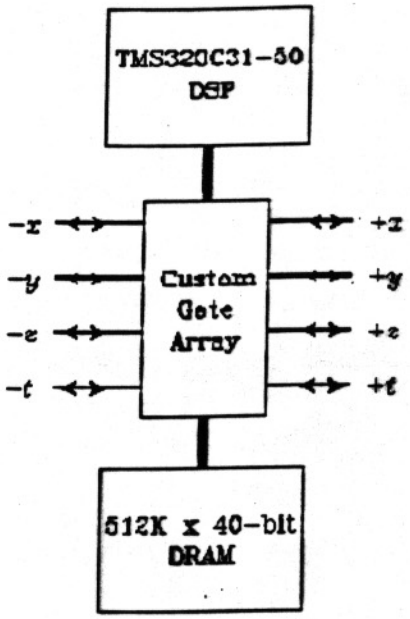
16 x 16 x 8 x 4

$1.8 M $15 / SUSTAINED MFLOP

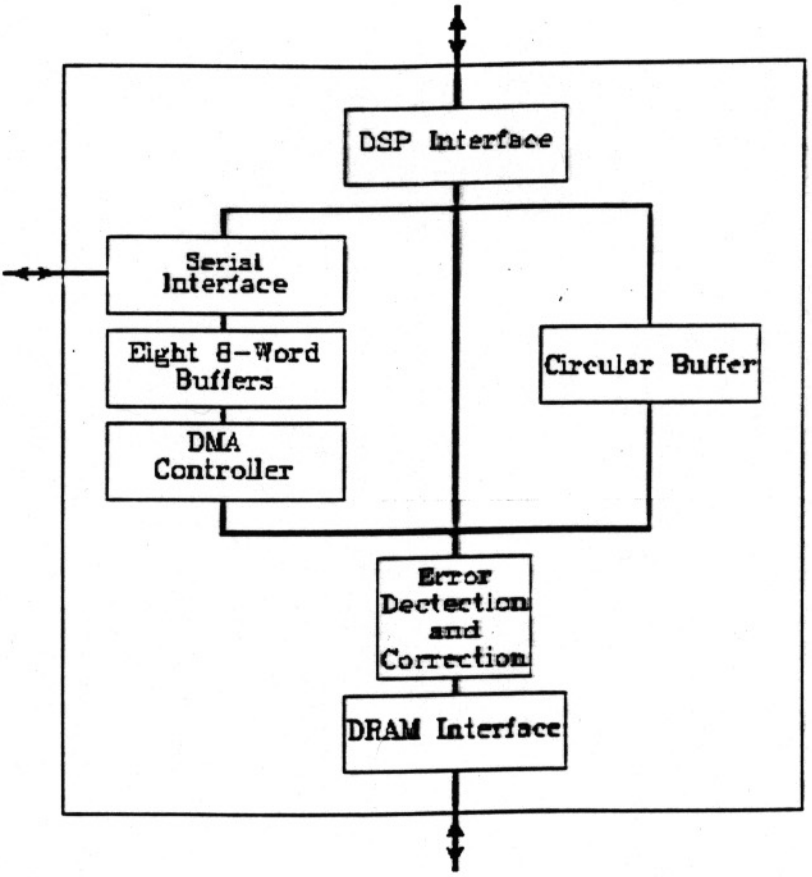
RIKEN MACHINE

600 GFLOPS PEAK

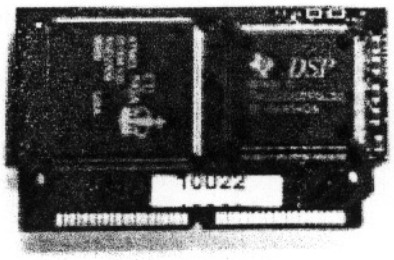
$10 / SUSTAINED MFLOP

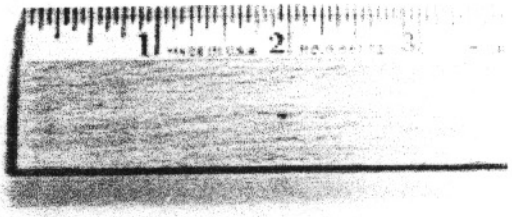


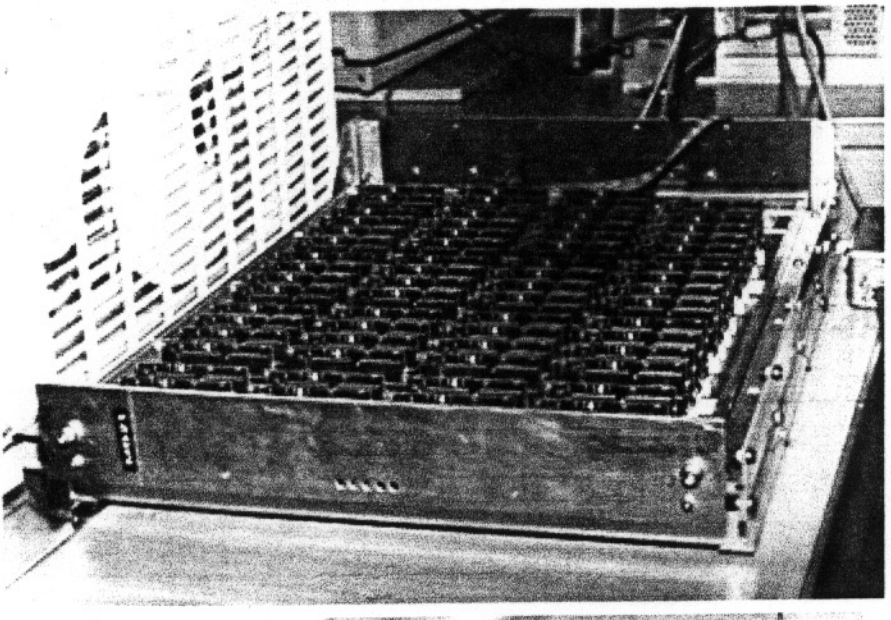
A block diagram of a single node.

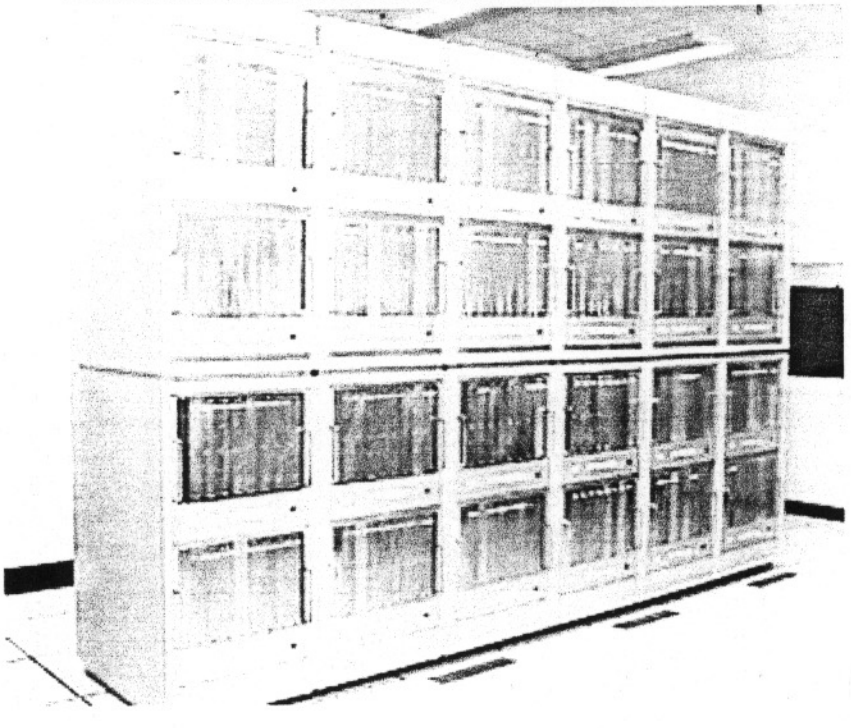


A diagram of the components contained within the gate array chip (*NGA*).


DSP
10022







APE mille

INFN / DESY SUCCESSOR TO APEcento

PROCESSOR : 528 MFLOPS CUSTOM

16-64 MBytes MEMORY / PROC

SINGLE PRECISION

44 % OF PEAK (S.P.)

TOWER

512 PROCESSORI

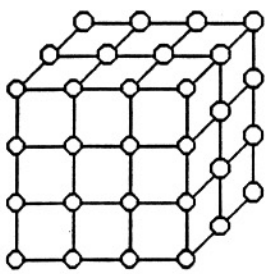
264 GFLOPS PEAK

8 x 8 x 8

APEmille

TOPOLOGY

An APEmille machine is a 3D grid of processing nodes with hardware data links between the 3D first neighbouring nodes. The smallest machine is the single Processing Board (PB) where are placed 8 processing nodes with a 2x2x2 topology, (cube). Arranging togheter more PBs it's possible to have more complicated, greater and faster machines.



- Hierarchical Hardware Topology
- Performances

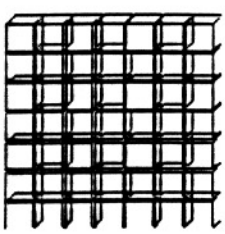
• Hierarchical Hardware Topology

The yellow cubes in this rappresentation are single APEmille boards with their 8 processing nodes supposed as placed at the vertices of the cube.

- Board: 8 Nodes (2*2*2 topology)

- SubCrate: 4 Boards (2*2*8 topology)

- Crate: 4 SubCrates -16 Boards (2*8*8 topology) 66 GFlops



- Tower: 4 Crates - 64 Boards (8*8*8) 264 GFlops
- APEMille: 4 Towers - 256 Boards(32*8*8)

• Performances

The following table reports the performance of the different configuration of an APEmille system: the maximum performance are about 1TeraFlops for the 66 MHz machine, and could be 1.6TFlops in a future 100 MHz release.

PROCESSING BOARD

- Scheme of a processing board

Fore each APEmille PB a CPU takes care of the program flow and drives the communication protocol with the host network. The CPU broadcasts a VLIW (Very Long Instruction Word) to the 8 Processing Nodes located on each PB. At the same time, as a result of its calculations, the CPU broadcasts a Global Address to the Processing Nodes. The Nodes can access their own local memory using this Global Address either directly or after adding a local offset to it, thus producing a local address.

The 8 Processing Nodes of a Board are arranged in a cubic lattice (a 2x2x2 **topology**). Each processing node has an arithmetic and logic unit including floating point adders, multipliers, a large multiport register file plus a memory controller with address generation capability, interfacing to local memory. Integer and bitwise operations and local addressing are some of the new main features of the Processing Nodes.

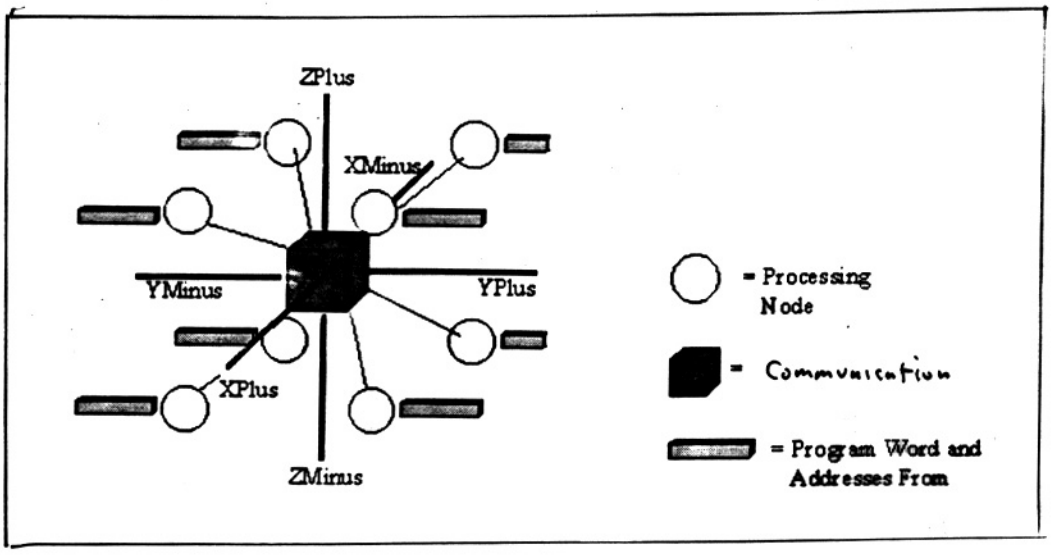
The first APEmille systems, based on presently available Syncronous Dynamic Ram and VLSI ASIC technologies, will adopt a 66 MHz, 528 MFlops processor. We expect however that the clock frequency may eventually increase up to 100 MHz. A 2048 100 MHz nodes system would have a peak **performance** of 1.6 TeraFlops.

According to our application requirements, the size of the local **memory** of each Processing Node will range from 2 to 8 MWord. Through a Communication Device each Processing Node directly accesses the data on its own 6 neighbouring Nodes. Some routing capabilities allow of the comunication device (slower) access to far-away nodes.

The Processing Board hosts:

- 1 CPU
- 8 Processing Nodes,
- 1 Communication Device
- required memories.

- **Scheme of connections among the 8 nodes of a PB**



Cmille is the custom chip which takes care of the remote data communications between processing nodes.

Tmille is the custom processor which controls the instruction flow and global addressing of each APEMille SIMD partition (a minimum of 8 processing nodes). Tmille is connected to the Host, to its own data memory, to an instruction memory and to the Communication Device. It also drives the Address Bus. There is a major difference from the Ape100 machine where one controller CPU normally drives 128 nodes and is housed in a different board.

The ***Processing Node*** is composed of a floating point processor (called JMILLE) which drives its own Local Memory, receives Instructions and Global Addresses from TMILLE and communicates with the Communication Device (CMILLE).

The Communication Device also has a channel connected to the CPU, and through this pathway, to the Host. Through this pathway the Host loads the program and data memories of the floating point processor.

All the CPUs and all the Nodes of a SIMD machine partition execute the same instruction. Each PB sends its Local Signals to a Root Board and receives Global Signals from it, in order to manage Synchronisation, Exceptions and Global Flow Control Conditions.

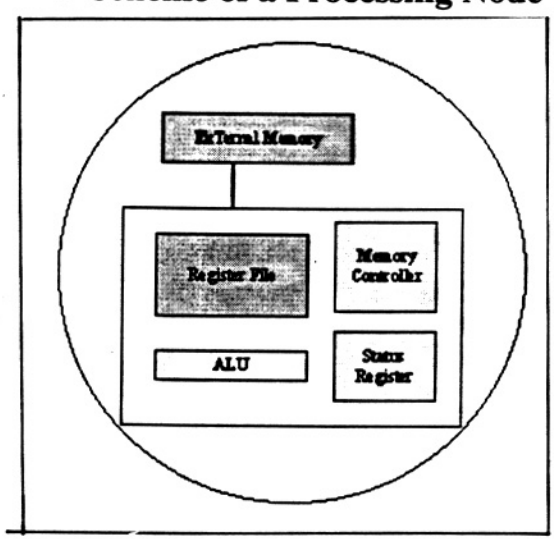
Each Processing Board interfaces the Host through a dedicated synchronous channel (APE Channel) going from the CPU to the Host Interface.

PROCESSING NODE

Each Processing Node is composed of a JMILLE floating point processor, attached to a SDRAM local memory. The Memory Controller inside Jmille generates addresses for the External Memory summing up a Global Address given by Tmille and a Local Address computed by Jmille itself. This way each APEMille Processing Node is able to generate a different Memory Address. The five addresses needed by the Multiport Register File are fixed by our compiler at compile time, and therefore distributed to Jmille inside the Program Word. This is obtained by means of our VLIW (Very Long Instruction Word) compiling technology.

Each Processing Node generates Status Signals (Global conditions, Exceptions etc.). These Status Signals are collected into a Global Status Return managed by the Control System. This connection allows the Control System to execute flow control instructions based on simultaneous logical conditions produced by the set of Processing Nodes. The instruction word read by JMILLE specifies the set of addressed location inside the large multiport Register File internal to JMILLE, and controls the Arithmetic Devices inside the processing node. Moreover it specifies Local Conditional Operation, Local Addressing functions, and Special Arithmetic function calculations to be performed by JMILLE.

• Scheme of a Processing Node



- ***Jmille*** : is the APEMille custom arithmetic processor with hardware supporting arithmetic, logical and bitwise operations on complex, double precision, single precision and integer data types.
- ***External Memory (ETM)*** : A SDRAM memory, directly attached to Jmille and controlled by Jmille itself, where all the data related to the node processing are stored.

Site	peak performance	status
Rome	260 Gflops	running
Zeuthen	130 Gflops	running
Rome II	65 Gflops	running
Bielefeld	80 Gflops	running
Milano/Parma	130 GFlops	planned, Dec. 2000
Pisa	130 Gflops	planned, Dec. 2000
Rome II	+ 65 Gflops	planned, Dec. 2000
Rome	+ 260 Gflops	planned, Dec. 2000
Zeuthen	+ 260 Gflops	planned, Spring 2001

1380

A short list of some large APEmille existing installations and of plans for the near future.

END OF 2001 (Gflops Peak)

Rome I	550
Rome II	260
Pisa	260
Bari	65
Milano	130
DESY	550
Bielefeld	146
Swansea	65
Orsay	16
	2042

QCDOC (QCD On a Chip)

- Natural extension of custom QCDSP machines at Columbia and RIKEN

- Partnership with IBM since December1999 to design and manufacture ASIC

  Processor: 1 Gflops peak 440 Power PC

  4 MB on-chip memory

  8 Gbits/sec interprocessor communication

- Schedule

  ASIC December 2001

  Prototype March 2002

  10 Tflops sustained QCDOC at BNL October 2003

  Hardware cost: $10 M

- Collaboration

  Columbia University

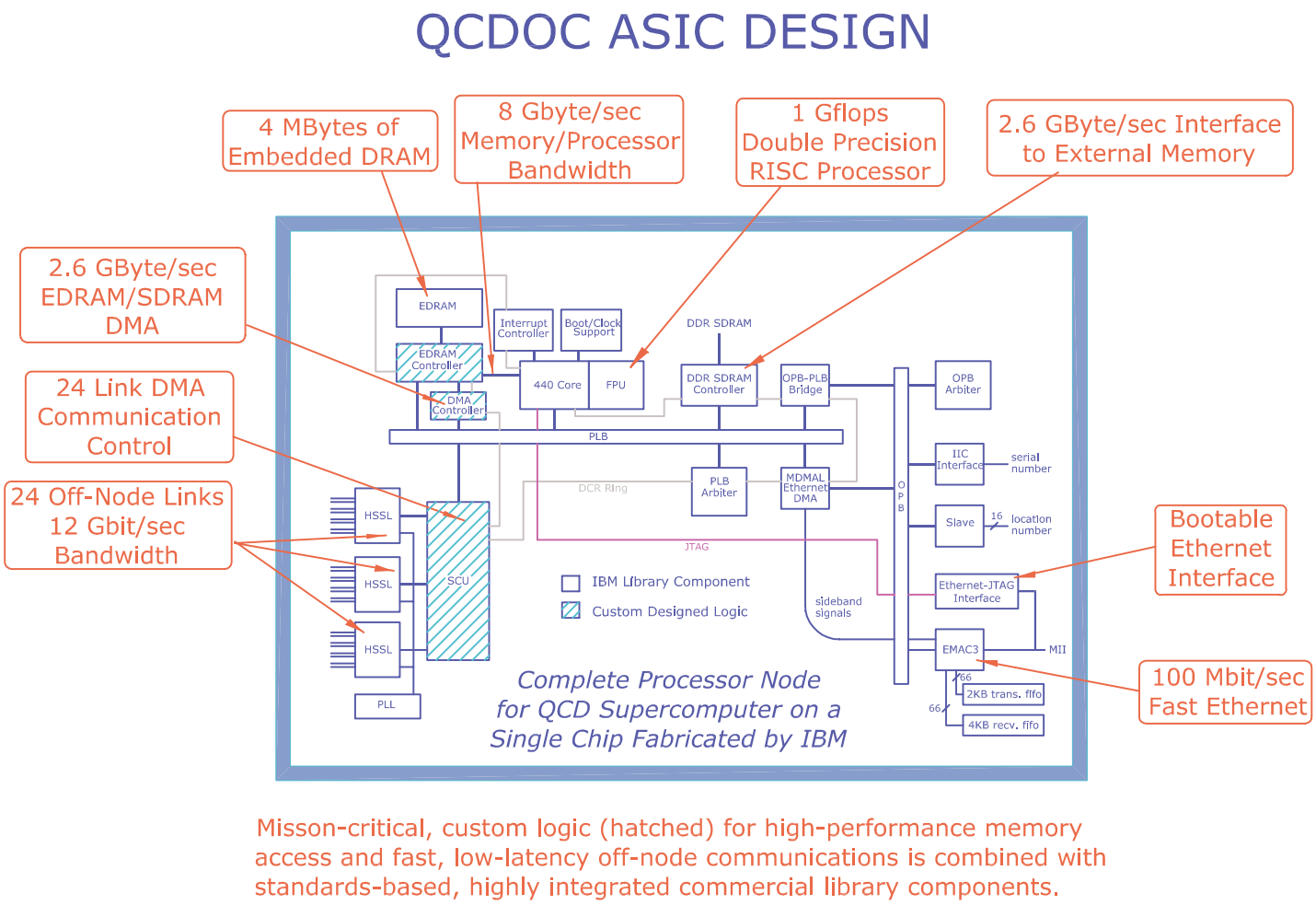
  RIKEN

  IBM

  Edinburgh

  Fermilab

QCDOC Chip Design



Misson-critical, custom logic (hatched) for high-performance memory access and fast, low-latency off-node communications is combined with standards-based, highly integrated commercial library components.

Lattice resources available internationally

(from whitepaper http://thy.phy.bnl/gov/www/qcdocworkshop.html)

State-of-art quenched calculation	50 Gflops-yrs
Largest NERSC FY00 QCD allocation	2 Gflops-yrs
Largest NSF FY00 QCD allocation	10 Gflops-yrs

Country	**FY00 Sustained Gflops**
Germany	264
Italy	554
Japan	980
U.K.	41
U.S. (Columbia)	120
U.S. (RIKEN)	180
U.S. (open)	20

European Committee for Future Accelerators (ECFA) report

- Several 10 Tflops computers in FY03
- UKQCD committed to buy QCDOC

Custom Machine Summary

	QCDSP	QCDOC	APEmille
Processor	TI DSP 50 MF	Power PC 440 1 GF	Custom 528 MF
Memory	2MB / proc	4MB/proc	16-64 MB/proc
Interconnect	25 Mbit/sec X16	500Mbit/s X24	
Performance	30%	50% goal	44%
Machines	400 GF 600 GF	plan 5 TF 10 TF	264 GF
Cost	$10-15 / sust. MF	$1/sust. MF	

4 Compute Engines for Lattice-QCD

- APE100: INFN, 25.6 Gflops (QH4 largest unit) (1994)
- CP-PACS: Tsukuba Center for Computational Physics and HITACHI, 600 Gflops (1996)
- Columbia-QCDSP: Columbia University, 100 Gflops (largest unit?) (1997)
- Cluster Computers: e.g. ALiCE, 160 Gflops (2000)
- APEmille: INFN/DESY, 64 Gflops (largest unit) (2001)

- Columbia-~~QCDSP~~ QCDOC: Columbia University/UKQCD, 10 Tflops (2003)
- apeNEXT: INFN/DESY, 5 Tflops (2004)
- Clusters: Jefferson Lab/MIT, FNAL, 10 Tflops (2004/5)