-AC

FSU-SCRI-97C-109

QCDSP: A Teraflop Scale Massively Parallel Supercomputer

Dong Chen
Dept. of Physics
Massachusetts Institute of Technology
chen@ctpa03.mit.edu

Ping Chen
Dept. of Physics
Columbia University
pchen@cuphyf.phys.columbia.edu

Norman H. Christ
Dept. of Physics
Columbia University
nhc@cuphyf.phys.columbia.edu

Robert G. Edwards
Supercomputer Computations Research Institute
Florida State University
edwards@scri.fsu.edu

George Fleming
Dept. of Physics
Columbia University
gfleming@cuphyf.phys.columbia.edu

Alan Gara
Nevis Labs
Columbia University Nevis Labs
gara@nevis.nevis.columbia.edu

Sten Hansen
Fermilab National Acceleration Lab
hansen@fnal.fnal.gov

Anthony D. Kennedy
Supercomputer Computations Research Institute
Florida State University
adk@scri.fsu.edu

Greg Kilcup
Dept. of Physics
Ohio State University
kilcup@pacific.mps.ohio-state.edu

Yu Bing Luo
Dept. of Physics
Columbia University
roy@cuphyf.phys.columbia.edu

Catalin Malureanu
Dept. of Physics
Columbia University
catalin@cuphyf.phys.columbia.edu

Robert D. Mawhinney
Dept. of Physics
Columbia University
rdm@cuphyf.phys.columbia.edu

John Parsons
Nevis Labs
Columbia University Nevis Labs
parsons@nevis.nevis.columbia.edu

Jim Sexton
Dept. of Physics
Trinity College
Dublin, Ireland
sexton@maths.tcd.ie

Chulwoo Jung
Dept. of Physics
Columbia University
chulwoo@cuphyf.phys.columbia.edu

Adrian Kahler
Dept. of Physics
Columbia University
adrian@cuphyf.phys.columbia.edu

Stephen Kasow
Dept. of Physics
Columbia University
kasow@cuphyf.phys.columbia.edu

ChengZhong Sui
Dept. of Physics
Columbia University
sui@cuphyf.phys.columbia.edu

Pavlos Vranas
Dept. of Physics
Columbia University
vranas@cuphyf.phys.columbia.edu

Abstract:

We discuss the work of the QCDSP collaboration to build an inexpensive Teraflop scale massively parallel computer suitable for computations in Quantum Chromodynamics (QCD). The computer is a collection of nodes connected in a four dimensional toroidial grid with nearest neighbor bit serial communications. A node is composed of a Texas Instruments Digital Signal Processor (DSP), memory, and a custom made communications and memory controller chip. An 8192 node computer with a peak speed of 0.4 Teraflops is being constructed at Columbia University for a cost of $1.8 Million. A 12,288-node machine with a peak speed of 0.6 Teraflops is being constructed for the RIKEN Brookhaven Research Center. Other computers have been built including a 50 Gigaflop version for Florida State University.

Keywords:

parallel, supercomputer, digital signal processor, QCD

Introduction

The atoms and nuclei of everyday matter are now known to be made up of still tinier particles known as quarks and leptons. Their behavior is described by quantum field theories, which combine Quantum Mechanics with Einstein's Special Theory of Relativity. The theory describing the interactions of quarks, mediated by so-called gluons, is called Quantum Chromodynamics (QCD). Although this theory can be formulated in a strikingly simple form, extracting the detailed predictions required for comparison with experiment is a challenging theoretical problem. The theory is made suitable for computations by representing the usual space and time by a four dimensional discrete grid or lattice of points. Using finer grained lattices requires ever larger number of points; hence, the computational demands of the problem are enormous and require large scale supercomputers.

In this paper, we discuss the work of the QCDSP collaboration to build an inexpensive Teraflop scale massively parallel computer suitable for computations in (QCD) (see references *[1]*, *[2]*, *[3]*, *[4]*, *[5]*). A 0.4 Teraflop 8192-node machine is being built at Columbia University for a cost of about $1.8

million. A 0.6 Teraflop 12,288-node machine is being built for the RIKEN Brookhaven Research Center for also a cost of about $1.8 million. The cost per flop is lower for this machine since the price of various components (including memory) have lowered over time. There are some other smaller machines, including a 50 Gigaflop version at SCRI, Florida State University which is operational now.

The organization of this paper is described as follows. We first describe some of the design considerations. We then describe the architecture of the machine. Details of the hardware components and assembly are presented, and the software is described. Finally, there are some conclusions.

Design Considerations

Because lattice QCD is described by a four dimensional regular grid, the computational methods are particularly suited for grid-based data parallelism. In particular, fine grained parallelism is suitable. The issue is then to make the cost of communications small enough to efficiently use a small number of sites per node, and a large number of nodes to achieve a large aggregrate speed.

In Figure 1, a two-dimensional example of the type of grid mapping is shown. The amount of data to be communicated is proportional to the surface area of the subgrid on each processor. The limitation of scalability is the communication to computation ratio. Namely, the amount of computations is falling faster than the amount of communications when the number of processors is increased for a problem which takes the same total elapsed time to solve.

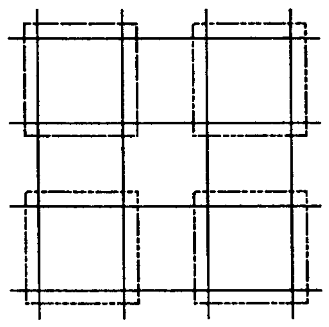

Figure 1: A two-dimensional 4x4 grid showing the 2x2 virtual grid on each of the four 2x2 physical nodes. A communication involves each node sending the face of its virtual grid in some chosen direction.

The effect of latency of communications can be lowered because intensive parts of the computation can be overlapped with communications. Similarly, if suitably wide communication wires are used the bandwidth is not issue. The main limitation for scalability is the packet size and the overhead to initiate a communication. For the type of problems we would study on a Teraflop scale computer, we typically need to exchange only a few hundred bytes between neighboring processors for the communication of a face.

It was not cost effective to design and build a microprocessor, so an off the shelf processor was used. Modern microprocessors rely on reuse of data via on-chip caches to compensate for lack of off-chip bandwidth. However, for fine-grained parallel problems, the cached data becomes stale too rapidly, so the performance of the processor is limited by its off-chip memory bandwidth.

Distributed memory for individual processors was used instead of a large shared memory model since in the latter memory coherence between processors becomes a bottleneck. The same issue arises for a collection of symmetric multiprocessor (SMP) computers, of which each SMP has it's own memory. While the cache coherence conflicts for a large grained problem are rare, the bus becomes a bottleneck for more processors and/or very fine-grained problems.

Requiring the cpu to handle off processor communications is a bottleneck. However, we can offload the face communications in Figure 1 to a seperate DMA engine with *block-strided move* capability.

The precision of calculations is always a concern in numerical computations. We decided to use 32-bit single-precision for the time consuming parts of calculations, and higher precision when needed for operations like global sums. Work has shown (see for example *[6]*) that this precision is sufficient for this scale machine and the type of problems studied.

Design choices for the QCDSP Teraflop Scale Computer

The computer is a collection of nodes connected in a four dimensional toroidial grid with bit serial communications. This grid is the main communications network for computations. An other network exists for diagnostics, booting, and disk communication. The main elements of a processing node are as follows:

- Texas Instruments TMS320C31 50MHz Digital Signal Processor (DSP) capable of performing a floating point multiplication and an accumulation in a single 25MHz cycle,
- Two Megabytes of DRAM with a 39-bit data bus (32-bit data word with 7 bits for Error Detection and Correction).
- The Node Gate Array (NGA), the cornerstone of our design, is a custom Application Specific Integrated Circuit (ASIC). The NGA provides a controller for the serial communication protocol, arbitrates memory accesses, handles recovery from errors in DRAM and serial wires, and includes a buffer that balances data transfers between the DSP and DRAM.

A fully assembled node costs about $81 in quantity. They are designed to be easily replaced upon failure.

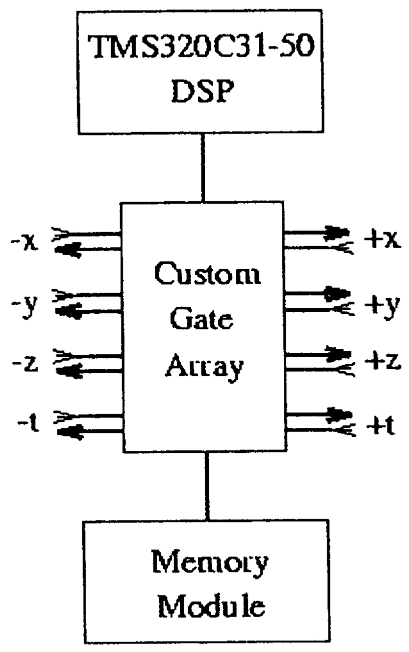

Figure 2: Block diagram of a node with the DSP, and the NGA which acts a communication controller for the four-dimensional grid network and a memory controller for DRAM.

There are 63 processing nodes, called *daughter boards*, on a motherboard. A motherboard holds 63 daughter boards attached to it through SIMM connectors. A 64th node, called node 0, is soldered directly to the motherboard. From the user's programming point of view all 64 nodes are treated equivalently, node 0 only behaves specially for booting, diagnostics and I/O operations. Eight motherboards are in a crate and four crates are stacked in a rack with water-cooled heat exchangers between each crate.

There are 2 main networks in the machine. There is the four dimensional serial communications network and a control/diagnostic network.

- The four-dimensional network uses programmable 25Mhz or 50Mhz bit-serial four dimensional communications between the processors. Sending or receiving data can be carried out in both forward and backward directions in all four dimensions simultaneously. Supported are nearest neighbor transfers as well as pass-through operations like add, max, and broadcast to support global sums. On each motherboard, the processors are configured in a 4x4x2x2 grid, with the first dimension periodic on the motherboard. The remaining three dimensions connect to a backplane via edge connectors. This is the intended mode of communication for user programs.
- The node 0 of each motherboard has two narrow differential SCSI chips to form a SCSI network. Each motherboard also has a DSP serial network. A SUN workstation hosts the QCDSP computer and communicates to it over SCSI. The SCSI network allows the host workstation to communicate with node 0 on any motherboard. The DSP serial network allows node 0 on a motherboard to broadcast to all 63 daughter boards or read and write to one daughter board at a time. The SCSI tree plus DSP serial network provide a reliable, relatively slow network for boot-time I/O and hardware verification.

Figure 4 shows these two networks, plus an example of a SCSI tree connection which links motherboards to each other and the host workstation. Node 0 on each motherboard has 8 Mbytes of local DRAM, while the daughter boards have 2 MBytes of DRAM. Node 0 has access to the single PROM on the motherboard and the electronics driving the off-board serial communications network connections. The additional memory of node 0 can be used for buffering I/O from the processor to the front end. Used in conjunction with the four dimensional network to communicate data to node 0, disks attached directly to the SCSI network of the QCDSP provide a high speed storage subsystem.

More complicated communication patterns for operations like an FFT are not directly supported by special hardware.

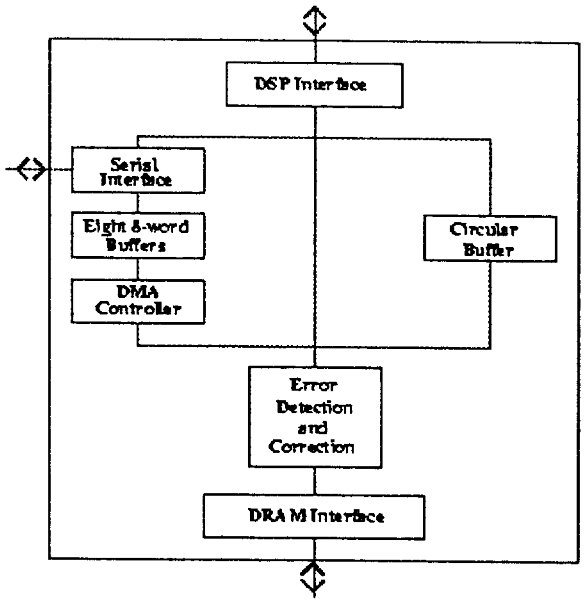

Figure 3: A diagram of the functional blocks of the Node Gate Array (NGA), a custom designed ASIC chip that is a joint communications and memory controller for a node.

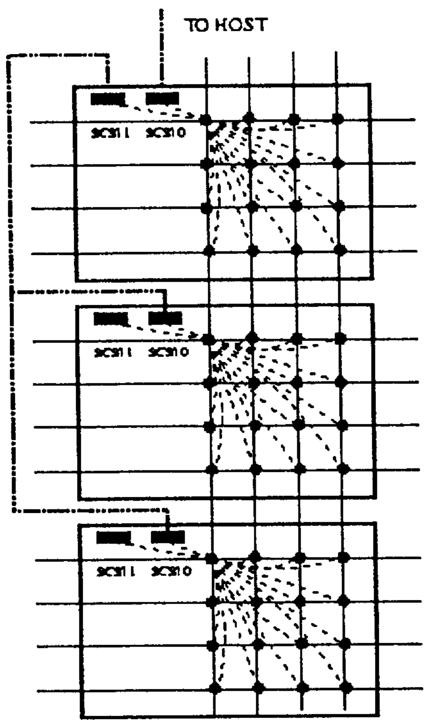

Figure 4: A diagram of the various communications paths in QCDSP. Here the nodes are depicted as located in a two-dimensional mesh (for ease of exposition) rather than in the four-dimensional mesh which is actually implemented. The filled circles represent processing nodes, the thin straight lines the four-dimensional nearest neighbor network, the thin dashed lines the DSP serial network on the motherboards and the thick dotted lines the connections in the SCSI tree. The thick solid lines making up boxes give the physical boundary of the motherboard.

HARDWARE COMPONENTS

PROCESSING NODES

The TI TMS320C31 DSP is a widely used processor for embedded systems and costs about $40. It is a word based microprocessor, with a 32 bit word for instruction and data. The DSP has a 2K word on-chip memory section that can be used for data or code in addition to a 64 word instruction cache. Performance-critical sections of code can be loaded into the on-chip memory for increased performance.

The TI DSP has a mature, user friendly, development environment. Amongst other tools, there is a C and C++ cross compiler and linker for the SUN. The linker allows great flexibility in how code is mapped into memory with support for run-time loading of code from main memory into the DSP on-chip memory.

A daughter board has seven surface mounted integrated circuit chips on a 1.8''x2.7'' 6 layer printed circuit board; one DSP, one NGA and 5 DRAM chips. The NGA can drive two different types of physical chips (either 512k by 8 bit DRAMs or 256k by 16 bit DRAMs) to help insulate us from changes in the available configurations of memory. From a programmer's perspective, the type of physical memory used is irrelevant.

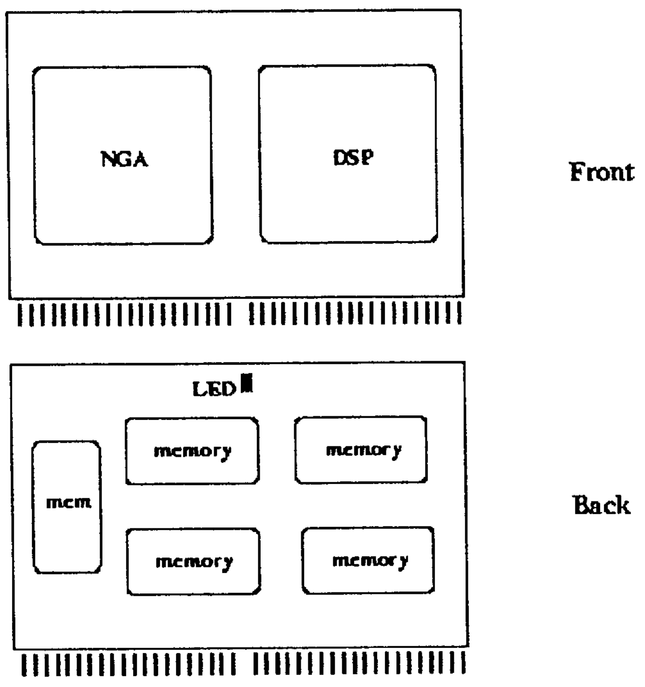

Figure 5: Diagram of a daughterboard showing locations of the DSP, NGA, and DRAM chips. A 1.8''x2.7'' 6 layer printed circuit board.

There are a large numbers of chips in the Teraflop scale computers, and we thus expect failures over the life of the machine. The majority of these chips are in the daughter boards. The DSP and DRAM have failure rates quoted at 100 million hours Mean Time Between Failure (MTBF). With an 8K node machine and 7 chips per daughter board, we expect roughly 1500 hours MTBF, or 5 failures per year. The SIMM socket connectors for the daughter boards allow easy replacement in the field. There is no provision in the machine for the dynamic reassignment of failed nodes with spares via software.

Shown in the NGA are the functional units corresponding to the Serial Communications Unit (SCU) and the Circular Buffer, each of which have independent access to the DRAM memory interface.

- The Serial Communications Unit provides an independent DMA engine for each off-node, communication direction. Thus, there are eight independent DMA engines that control a block strided communication. There are eight autonomous DMA engines controlling the forward and backward communications along the 8 nearest neighbor links of a node. The direction of transfer along a wire is decided in advance, so in any one communication the send unit of one node and the corresponding receive unit of the partner are programmed for operation. The communications across a link are independent of other links and effectively synchronize the machine.

  The communication transfers along a link have a word level protocol using single bit parity for error detection. In a transfer, two guard bits are sent, then the 32 bit word, followed by the parity bit. Upon receipt of the 32 bits, the receive buffer is flushed and it sends a 3 bit acknowledgement. Upon parity failure, the transfer is redone. Thus, if a receiver has not reached the point of code where it is programmed, the sender will fill the receive buffer, but no acknowledgement will be sent until the receiver is ready. Each send/receive unit has a buffer for the transfer, and an 8 word deep queue of words for storage into memory. Access to the internal memory bus is arbitrated with the Circular Buffer and Bus Interface.

The send/receive units of the SCU are memory map programmable. Each SCU DMA engine has a separate 32 bit control register that has fields for the number of blocks, the length of a block, and the stride between blocks. Writing the starting address of the source or target to the corresponding DMA unit starts the transfer. The corresponding send and receive do not have to have the same block/stride, only the total number of words must be the same. Status registers are available for polling a specific wire, or combinations of wires. Error status registers are also available.

Communications can be overlapped with computations in an efficient manner. Also provided are pass-through operations like add, max, and broadcast operations to support arbitrarily high global sums.

- To increase memory bandwidth, there is a programmable fetch-ahead unit called the Circular Buffer. The 32-word Circular Buffer can be instructed to prefetch a certain number of words from DRAM which can be read by the DSP in zero wait state mode later on. The Circular Buffer has an embedded protection against accesses to invalid data locations unless the programmer specifically turns this protection off. The size of the Circular Buffer comfortably accommodates important data elements in our computations.

MOTHERBOARDS

A sketch of a motherboard is shown in Figure 6. The motherboards are 14.5'' by 20.5'', 10 layer printed circuit boards laid out at Columbia and manufactured by Hadco. Node 0 has a total of 8 MB of DRAM and is soldered directly to the motherboard. There are 63 SIMM sockets for the daughterboards. The signals for the serial communications network, SCSI, global interrupts, clock, power and reset go through the edge connectors to the backplane.

One of the important features of the DSP is its built in serial port and boot loader. A pin can select booting from an on-chip ROM code, the serial port, or an external PROM. On a motherboard, the node 0 boots from an external PROM. The 63 daughterboards are set to boot off their serial ports. On reset, they wait for boot code from node 0 via a PAL switch. The node 0 PROM has driver code to enable it to receive commands over the SCSI network. A SCSI reset command will reset all 64 nodes.

Since we can only have 7 target devices on the SCSI bus, having two SCSI chips allows us to support a tree of SCSI connected motherboards. In addition to communicating with the host front end or other motherboards, the SCSI network Figure 6 can also be used to connect to disks or tape drives.

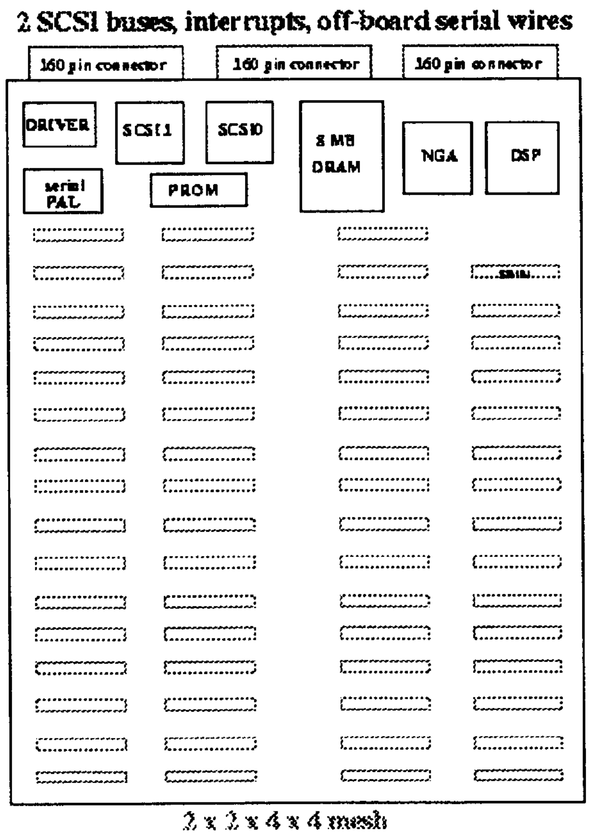

Figure 6: A diagram of a motherboard used in QCDSP. Shown are the SIMM sockets for the daughterboards. Node 0 with its larger memory is soldered directly on the motherboard. There are edge connectors to the backplane that include the communications and SCSI lines for the two onboard SCSI chips.

BACKPLANES AND CRATES

The backplane design includes an equal-time, 50 MHz clock fanout to all the crates in the machine. This is important since nearest neighbors in the four dimensional network will generally be in different crates. An equal time reset signal is also distributed. Even though the machine is self-synchronizing on nearest neighbor links, it can be useful for debugging purposes to have all processors come up from reset synchronously. Another vital reason for a synchronous reset is so all nodes will agree on the correct phase for a 25 MHz signal derived from the 50 Mhz clock. When the SCUs are running at 25 MHz, all nodes must agree on this signal.

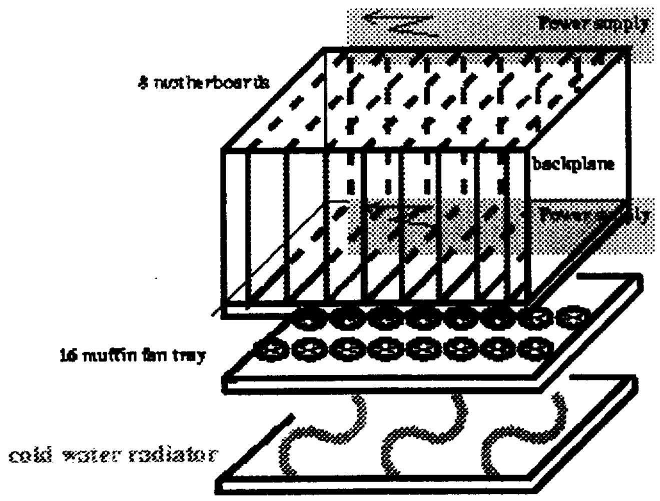

Figure 7: Each crate has 8 motherboards (512 nodes) plugged into a backplane which provides each motherboard with connections for power, serial communications, SCSI, and other signals.

The backplane also handles the distribution of three global interrupts linking all nodes. One interrupt gives the global synchronization signal, the second a non-recoverable error signal, and the third a recoverable error signal.

The backplane circuit boards were laid out and assembled by Bustronics. The crates were designed and built by Elma Corporation, with the first single crate arriving in December 1995. Also manufactured are racks of four crates. Each crate requires a separate 20 amp, 220 volt circuit and a single crate is cooled by a tray of muffin fans. The four crates that are stacked into a rack have water-cooled heat exchangers between each crate.

For a given physical configuration of the crates, the four dimensional configuration of QCDSP is determined by external cables connected to the backplane. In particular, the cabling can produce a machine of dimension 4x4ix2jx2k where *i*, *j* and *k* are integers. Two and three dimensional arrays are also possible by only altering the external cabling.

Software

Hardware Verification

A significant amount of work has gone into software verification of the hardware. The NGA was entirely designed and tested on a PC and a Sparc-10 workstation at Columbia using the Viewlogic software tools. Software models of the DSP and DRAM where supplied by Logic Modeling Co.

Code to test the NGA, called *test vectors*, where developed. These tests, a quarter million cycles worth of code including the actual physics code of a conjugate gradient method to solve a linear system of equations, were first run at the logic level, and then on the gate level to account for derating of the chip. The NGA manufacturer, Atmel, would insure all the NGA's passed these tests.

A single node prototype machine was initially developed to test the daughter boards. These tests were so successful that the same basic set of tests were packaged together and used to test the daughter boards at the manufacturer thus eliminating debugging time for the motherboards. A single motherboard machine was constructed to test full motherboard operation at the manufacturer.

TWO NODE PROTOTYPE

Early on, a two node prototype machine was constructed for testing of the daughter boards. It was used for low level software development such as the kernel, and also user level physics code development.

QCDSP OPERATING SYSTEM SOFTWARE

We should point out that the QCDSP computer does not have a conventional kernel, like a UNIX kernel. There is no memory protection scheme via a memory controller unit, and the machine does not time share among different processes. Also, the machine can not be divided to run two different processes among different blocks of the machine. However, the machine can be time multiplexed between successive processes with a run-time scale of minutes or hours.

The basic level of the QCDSP operating system, called the boot kernel runs at power on or reset. The node 0 of each motherboard boots off its PROM, and a boot code is broadcasted by the DSP serial line via the PAL switch to the on-chip memory of the 63 daughter boards. At no point is DRAM used.

The kernel on nodes 1-63 services a set of fundamental requests that arrive at the DSP serial port, such as write, read and execute. The kernel on node 0 is more complex since it has to control the two SCSI chips and the PAL switch. It must also handle the routing of SCSI packets and service requests along the SCSI tree as well as along the DSP serial port tree. The node 0 kernel expects to receive requests according to a predefined communications protocol. This protocol allows recursive routing from say the host front end to several layers down in the tree.

After the initial booting, the motherboard is ready for additional commands via SCSI. Diagonistic programs are run on each node 0 to verify its DRAM is operating properly. Full diagonistics are then run on the entire machine. At this point the full boot kernel is downloaded. User code can be downloaded to each motherboard node 0 and then broadcasted to each daughterboard. For higher bandwidth, the code can be broadcasted from the host to a single node then ''rotated'' around the machine via the four dimensional wires.

The kernel of the DSP provides basic system services, like run-time I/O to the front-end accessed by the usual fprintf, fread, etc., possible disk accesses to back-end disks, global interrupt control handling, error status and handling. The global interrupt wires can be programmed to generate interrupts if a double bit memory error is detected or global sum communications fail. By not enabling interrupts on the DSP, the global interrupt wires can be used as global flags which are useful for constructing software barriers and synchronization.

System service libraries are available for rotating data from around the grid via the four dimensional communication wires to the single node 0 of the machine or to the node 0 of each motherboard. This provides a high bandwidth way to write out information to the front-end or to back-end disks. Also provided are basic four dimensional communication calls, and routines for global summations of data.

The SUN host front-end has a custom SCSI device driver for communicating with the QCDSP. A user level program or shell is run on the front-end to communicate with the QCDSP. Diagnostics can be run, code cross-compiled on the SUN can be downloaded and executed, and memory and status of the DSP's can be read. This host front-end program is responsible for servicing requests from the QCDSP. A typical mode of operation is as follows: the SUN, being the initiator of the SCSI bus, sends a packet to a connecting motherboard indicating to run a program. After the program starts, the SUN sends down a packet with a protocol asking if there is anything the QCDSP requests. When I/O buffers on node 0 become full, the motherboard will respond that it wants data to be printed to selected files, possibly standard output. The SUN services the request and then submits a new packet asking for requests. Upon an error, the QCDSP can be reset via a SCSI reset. The contents of memory are preserved and diagnostics can be run.

PHYSICS SOFTWARE

A great deal of effort has gone into developing high level user software. The most time consuming portions of the typical physics code is solving large sparse linear systems of equations by the conjugate gradient method. Optmized versions exists sustaining 30%. Typical user code (using optmized basic primitives) sustains 20%. Performance scales quite linearly with the size of the machine (down to about 4 virtual sites per node) since nearest neighbor communications are the dominant communications pattern. This decent performance is in accordance with the design philosophy of the very low cost of the machine and is close to the design target.

One of the most important aspects of the physics code is the Monte Carlo evolution of the "gluon" fields of QCD using the optimized conjugate gradient codes. There are then various calculation of quantities like the mass of a proton using these gluon fields. High level C and C++ code exists for these applications. The Tartan C++ compiler has double precision library support which allows checks of stability (albeit slowly) under increased precision. C++ provides a convenient way to describe the parallel nature of the machine and is being used extensively.

The SCRI based programming environment, called *SZIN* has been highly optimized for the QCDSP. This system is an object-oriented macro-based C code for QCD. It allows user level programs to port between a large variety of computers using the same high-level problem-oriented source code. The system not only allows different code optimizations for different architectures, but also allows completely different data organization, and thus has been able to achieve very good performance on a wide range of completely different architectures including the QCDSP, CM-2, ETA10, Cray YMP, a variety of workstations, a cluster of workstations, and an SMP workstation.

Conclusions

The QCDSP is a simple elegant inexpensive design for a Teraflop scale supercomputer. We would like to stress that it is **not** a special purpose computer for solving QCD. Namely, no special features of QCD used, like multipling complex numbers in hardware. Rather, the approach was to design a machine to solve efficiently (at least) the more limited class of regular grid based problems with data parallelism.

A 50 Gigaflop version of the QCDSP is operational now at SCRI, Florida State University. A 0.1 Teraflop advanced prototype should be operational at Columbia by the end of August. This is part of the 0.4 Teraflop machine which is under construction now, and should be operational by the end of 1997.

The 0.6 Teraflop 12,288-node machine being built for the RIKEN Brookhaven Research Center is intended to be operational by the end of January.

Given the low cost of the design, the cost per flop makes the QCDSP an attractive computing platform.

References

1. Norman H. Christ. **A 0.5 Teraflops Machine Optimized for Lattice QCD**. Nucl. Phys. B (Proc. Suppl.) 34 (1994) 820.

2. Igor V. Arsenin. **Architectural Choices for the Columbia 0.8 Teraflops Machine**. Nucl. Phys. B (Proc. Suppl.) 42 (1995) 902. hep-lat/9412093.

3. Robert D. Mahwinney. **The Status of US Teraflops-scale Projects**. Nucl. Phys. B (Proc. Suppl.) 42 (1995) 140. hep-lat/9412068.

4. Igor V. Arsenin, *et. al.* **Status of the 0.8 Teraflops Supercomputer at Columbia**. Nucl. Phys. B (Proc. Suppl.) 47 (1996) 804. hep-lat/9509075.

5. Robert D. Mahwinney. **QCDSP: The First 64 Nodes**. Nucl. Phys. B (Proc. Suppl.) 53 (1997) 1010. hep-lat/9705028.

6. Robert G. Edwards, Ivan Horvath, Anthony. D. Kennedy **Instabilities and Non-Reversibility of Molecular Dynamics Trajectories**. Nucl. Phys. B484 (1997) 375-399. hep-lat/9606004.