

The QCDOC Project

P. Boyle^{a b *}, D. Chen^c, N. Christ^b, M. Clark^a, S. Cohen^b, C. Cristian^b, Z. Dong^b, A. Gara^c, B. Joo^a, C. Jung^d, C. Kim^b, L. Levkova^b, X. Liao^b, G. Liu^b, S. Li^b, H. Lin^b, R. Mawhinney^b, S. Ohta^e, K. Petrov^b, T. Wettig^{ef}, A. Yamaguchi^{bg}

^aSchool of Physics, The University of Edinburgh, Edinburgh, UK

^bPhysics Department, Columbia University, New York, USA

^cIBM Thomas J Watson Research Center, Yorktown Heights, NY, USA

^dBrookhaven National Laboratory, NY, USA

^eRiken-Brookhaven Research Center, NY, USA

^fInstitute for Theoretical Physics, University of Regensburg, 93040 Regensburg, Germany

^gPhysics and Astronomy, The University of Glasgow, Glasgow, UK

The QCDOC project has developed a supercomputer optimised for the needs of Lattice QCD simulations. It provides a very competitive price to sustained performance ratio of around \$1 USD per sustained Megaflop/s in combination with outstanding scalability. Thus very large systems delivering over 5 TFlop/s of performance on the evolution of a single lattice is possible. Large prototypes have been built and are functioning correctly.

The software environment raises the state of the art in such custom supercomputers. It is based on a lean custom node operating system that eliminates many unnecessary overheads that plague other systems. Despite the custom nature, the operating system implements a standards compliant UNIX-like programming environment easing the porting of software from other systems. The SciDAC QMP interface adds internode communication in a fashion that provides a uniform cross-platform programming environment.

1. Background

The QCDOC project is an international collaboration to develop a massively parallel supercomputer architecture tailored to the needs of QCD simulations. The project has been centred at Columbia University with significant contributions from the RIKEN-BNL research center, from the UKQCD collaboration and from the Thomas J Watson IBM Research Center.

Over the last four years, the project has developed a custom system-on-a-chip (SoC) application specific integrated circuit (ASIC) containing almost all of the components required for a self contained compute node on a single chip.

First silicon for this ASIC was received in June 2003, and the process of bootstrapping up to very large working prototypes has been carried out over the subsequent year.

Three very large machines are now under con-

struction: one for each of UKQCD, the RIKEN institute, and the DOE SciDAC project [7]. These will contain 12,288 processors each (with additional spares kept used in smaller satellite machines). A sustained performance of over 5 Teraflop/s will be delivered by each of these machines on double precision QCD code, depending on application efficiency.

2. High Performance QCD

Dynamical lattice QCD simulations at light quark masses unavoidably require large amounts of computational power to be applied to the evolution of a single lattice.

This is most practically obtained by using many microprocessors in parallel on the problem. The memory and internode communication must be capable of keeping the floating point units supplied with data. Less capable systems fail to deliver the “quality Teraflop/s” that enable the dynamical evolution of practical lattice volumes

*Presented by P. Boyle at Lattice 2004

at high speed, despite delivering “Linpack Teraflop/s”.

The principal goal is to maximise Krylov solver iterations per second on a sensible lattice volume. This is the basic currency with which we can choose can either purchase lighter quarks, or more configurations as our physics research dictates.

3. Latency considerations

Ultimately, a large machine cannot perform more matrix multiplies per second than dictated by the latency characteristics of its interconnect. For MPI cluster interconnects around 5-10 μ s is state of the art. With a switch style interconnect, messages are typically sent consecutively, and the latency paid 8 times per Dirac application.

By eliminating message passing software overhead and slow bus structures, QCDOC reduces latency to a fraction of a microsecond - competitive with large shared memory machine. As we shall see QCDOC further alleviates the impact of latency by overlapping the latency of many messages with hardware support for simultaneous launching of many transfers.

We note that the latency of interconnects is fundamentally harder to improve than bandwidth, and does not significantly improve with Moore’s law. However, achieving the speed up required to run five dimensional formulations a factor L_s faster than on QCDOC is will prove much easier than obtaining the same speed up for a 4d formulation at fixed volume. Thus, in the future, the fifth dimension in Domain Wall and some formulations of the Overlap operator will come for free, in the sense that it only costs money and bandwidth.

4. Design Goals

The nature of QCD simulations may be exploited to simplify the design constraints for a QCD machine compared with a supposedly “general purpose” massively parallel supercomputer:

1. The problem is naturally Cartesian allowing for a simple Cartesian mesh network with nearest neighbour communication used for parallel transport of fields in coordinate space.

2. The only common non-nearest neighbour communication is global summation.

3. Scaling at fixed problem size is required (hard scaling). A small amount of high speed memory on each node will accelerate the problem when many nodes are used.

4. Both communication and memory access patterns are deterministic and repetitive.

In addition to performance on QCD, we must satisfy constraints on price, power consumption and system density to minimise purchase price, running costs, and floorspace.

The approach taken in the QCDOC design is to make use of the enormous investment made by the computer industry wherever possible, and add to this only those features that push the advantages described above. We use a low power embedded processing core and add our own on-chip memory and 6d torus network. This enables a very dense and scalable system.

5. Hardware Architecture

The QCDOC ASIC integrates a number of standard IBM Micro-electronics IP blocks with some custom components developed for this project. It has been fabricated in a 0.18 μ m mixed logic and DRAM process allowing us to place a large memory on chip, with a 1024 bit wide high bandwidth access. While it has been described at previous lattice conferences [1–5], we briefly summarise the functionality.

The processing core is an IBM PowerPC 440 integer processor with floating point co-processor, implementing the BookE PowerPC instruction set. There is an on chip bus heirarchy. The fast on-chip Processor Local Bus (PLB) connects the DDR memory controller to the PPC 440 core, and can be thought of as analogous to the frontside bus on a personal computer. The somewhat slower on-chip OPB (on-chip peripheral bus) may be thought of as analogous to the PCI bus in a conventional system.

The QCDOC team integrated the bus structure and a full set of standard peripherals around the CPU. This includes interrupt controllers, the bus arbiters and bridges, the on-chip single channed DDR memory controller, a 100Mbit ethernet controller and DMA engine, general purpose I/O, and an IIC interface.

The custom parts used in this project include

a simple machine wide global interrupt tree, serial communications unit (SCU), on chip memory controller (PEC). An Ethernet-JTAG device provides debug and boot support over ethernet.

5.1. Serial communications network

The six dimensional torus network is implemented in the serial communications unit. This logic block is connected to the rest of the chip via the PLB. Thus, our communications network is integrated at a very high-level in the bus hierarchy, giving with access to the memory controllers that is equal to that of the CPU. This in particular helps the QCDOC network deliver both the low latency and aggregate high bandwidth requirements of QCD compared with, say, cluster interconnects implemented on PCI cards.

The electrical signalling technology is a 500Mbit low voltage differential signal, driven by IBM High Speed Serial Link transceiver cores. 24 differential wire pairs provide bidirectional communication for a six dimensional mesh.

The logical protocol was implemented at Columbia, and has a fixed length packet size with 8 bytes data and 1 byte command header and parity. This keeps the protocol overhead low. Each packet is acknowledged and parity errors are negatively acknowledged, triggering a retransmit. Header options support partition wide interrupts, and an interrupting out-of-band data path called "supervisor" packets. Three outstanding packets are allowed on each link before progress halts awaiting acknowledgement. This covers the maximum round trip latency, and allows the links to progress at wirespeed.

The serial communications unit provides a bidirectional pipe for communicating data with nearest neighbour. The unit manages the 24 DMA engines. Each of these DMA engines has 16 programmable descriptors for the block strided moves appropriate to pulling faces from multidimensional arrays. An arbitrary subset of these DMA engines can be started using a single PowerPC instruction.

Global reduction assistance is implemented via a pass-through mode in the serial communications unit. After a modest synchronisation overhead on entering the mode, the hardware will implement the distribution of a value from each

node on a given axis to all nodes in that row. In this way all nodes can obtain data from all the other nodes. Any reduction operation can be performed using the CPU with this hardware acceleration of the communications pattern.

5.2. Prefetching eDRAM Controller

The PowerPC memory system has been enhanced with a custom component inserted between the processor and the PLB. The Prefetching Embedded DRAM Controller (PEC) provides access to the 4MB on-chip memory at full CPU clock speed, while passing normal memory requests to the PLB. The PEC provides independent interfaces to both data cache read and write ports over the processor data bus (PDB), and provides an PLB port for use by the DMA engines in the serial communications unit. The internal interface to the embedded DRAM arrays is more than adequate to cope with the maximum load placed on the PEC by the PDB and PLB ports, and this decoupled architecture allows the overlapping of communication and computation without any contention. Wide 128 byte lines are read and written from the DRAM arrays by the various read and write ports, and each port hardware prefetches two independent read streams, and double buffers write streams. Thus computational kernels with up to two sequentially stored input streams and one output stream (which can move arbitrarily in memory) will use the PEC optimally. This optimisation strategy is shared with many modern processors such as the Intel, AMD, Sparc, and Power chips which have hardware prefetch and write buffering, and should not be considered a QCDOC specific optimisation.

5.3. QCDOC System Integration

The QCDOC system is composed of three custom printed circuit boards, plugged together as building blocks. These are the daughterboard (holding two QCDOC compute nodes), the motherboard (holding 32 daughterboards), and the backplane (holding four motherboards).

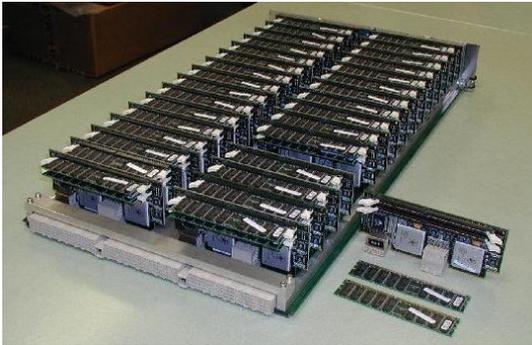
5.3.1. Daughterboards

The QCDOC daughterboard integrates two almost independent compute nodes on a single board. The board has independent DDR memory systems for each ASIC, and an ethernet sys-

Figure 1. Signalling stability during Wilson Dslash operation of the serial communications obtained by overlaying many transitions aligned to a reference clock.



Figure 2. 32 QCDOC daughterboards assembled into a QCDOC motherboard.



tem including ethernet transceivers and repeater yielding a single off board ethernet link.

One of the six torus network dimensions is wrapped periodically on a board, and the remaining five dimensions come off board. Signalling quality is excellent, and we include a sample "Eye diagram" demonstrating the timing quality of the High Speed Serial Links in Figure 1.

5.3.2. Motherboards

Figure 2 displays a populated QCDOC Motherboard. This board provides a very high node density enabled by our low power consumption system-on-a-chip. It is air cooled, with air blown in the channels between daughtercards. Edge connectors can be seen, which carry serial communication and ethernet links off board.

Three dimensions remain local to the motherboard, one being local to the daughtercard, and the remaining three dimensions come off motherboard, creating a long range cabling problem that is only three dimensional. Each motherboard carries a total of 768Gbit/s of internode communication bandwidth.

Each motherboard contains eight separate ethernet repeater systems (four daughtercards in

each), with 800 Mbit external ethernet bandwidth. In large systems the aggregate bandwidth is highly satisfactory: in fact throttling code has been inserted to prevent the medium sized QCDOC systems from accidentally carrying out denial of service attacks on the host.

5.3.3. Backplanes and cabinets

The large machines utilise watercooled cabinets containing 16 motherboards each, grouped in four slot backplanes. The ASICs are air cooled with recirculated airflow passing over chilled water heat exchangers. Two water cooled cabinets may be stacked giving a low footprint 2Teraflop/s machine.

All internode links are passed straight through the backplane to cables, and are driven from one ASIC through motherboards and cables to a neighbouring ASIC without redrive.

A single clock source drives the entire machine. The clock is fanned out but not phase aligned, and the QCDOC ASIC dynamically locks on to the phase of incoming data from its' neighbours and synchronises it with its own clock domain.

5.4. Host computer and disk system

The host computer for QCDOC is an IBM pSeries 8 processor SMP server running AIX. This machine and O/S combination was selected by benchmarking the network I/O bandwidth of a number of options. The host obtains excellent performance at delivering ethernet packets to the QCDOC nodes over its multiple Gigabit interfaces. Switch trees are contained within the rack for the boot, diagnostic, and I/O ethernet network. The top level switches are Gigabit, connecting the 100Mbit motherboard links to both a host computer with multiple interface cards, and to an array of network attached storage (NAS) boxes embedded in the switch tree to make up a high bandwidth parallel file system.

The packet download required to boot a 1024 node machine can be performed in as little as 12 seconds over a single interface, at 20% of wire speed. This is substantially less than the DRAM self test phase. Using multiple interfaces, the large 12 cabinet machines under construction will boot from cold in no more than a few minutes.

Figure 3. 4096 node prototype at Columbia.



5.5. Hardware Status

A 4 cabinet 4096 QCDOC machine, Figure 3 is installed in the machine room and cabled as four 1024 node machines for shakeout. One machine is fully debugged and running an initial test Staggered AsqTad RHMC evolution at a 420MHz cpu clock speed.

A further two 1024 node machines have nearly completed shakeout, and will commence production on DWF simulations soon. The boards to populate fourth machine have been received and it is undergoing initial debug. The first cabinets for the UKQCD machine have arrived at Brookhaven and are undergoing assembly, cabling, and electrical tests.

6. Software Architecture

The QCDOC software environment is composed of three key parts. Management software on the front end computer, the operating system running on the nodes, and the user run-time support libraries that present the programming interface to physics simulation code.

6.1. Qdaemon

The host software principally consists of a multi-threaded program called the "qdaemon". The qdaemon implements all the QCDOC commands that users execute, and provides scalable high bandwidth access to boot and run code on the QCDOC hardware. An enhanced shell, qcsh, is used to access the qdaemon and this simply forwards all qcdoc related commands directly to the qdaemon in a secure way.

The qdaemon provides full featured diagnostics of the machine and has been continually refined to identify all common failure modes hit during ma-

chine debug. It can divide a machine into many partitions and handle multiple user connections simultaneously to different partitions of the machine.

The boot process uses the on chip EthernetJTAG to download a small boot kernel directly to the instruction and data caches of each ASIC. This kernel performs carefully logged chip initialisation and self test, DRAM bring up and test, network driver initialisation, and then runs a simple bootstrapping protocol to allow the QCDOC node kernel to be down loaded at high speed.

A number of different precedents exist for the behaviour of input and output streams on parallel computers. On QCDOC, by default circular print buffers on most nodes, with node-0 printing to the console. Qdaemon commands print kernel and user print buffer logs. If this quirk is unpalatable, arbitrary subsets of the nodes may be selected as printing to the console, or library code may be used to setup each node as having standard input, output, and error connected to files on the host disk system via the UNIX-like "dup2" call.

6.2. QCDOC node kernels

The QCDOC kernel architecture is very advanced for a custom developed platform. It contains a flexible interrupt stack, multiple kernel threads and one user process, an NFS client, an RPC server and portmap daemon, and various device drivers including a network driver and kernel sockets implementation.

System calls are used to safely access privileged functionality, and provide a standards compliant UNIX system call environment (missing process support since there is only one process!) even though the operating system has been written from scratch by members of QCDOC team.

The PowerPC virtual memory hardware is used to protect the kernel space, but not no non-trivial virtual memory translation is used. This makes it easy to implement zero copy DMA direct from user space, and avoids TLB miss overhead that plague UNIX based HPC nodes. Time quanta are not used, and so scheduler induced slow down cannot occur. A sleep/wakeup event mechanism is used to schedule kernel threads only on key events, such as ethernet packet arrival, or inter-

node interrupts. The kernel RPC server and portmap thread implement the protocol to allow the host computer to control the node, and load and start programs.

Several programmer friendly features have been implemented, including PC sampling based profiling, debugger support, automatic symbol table parsing to identify the subroutine in which an exception occurs, and a dump of the processor context on program termination.

The ASIC has many self diagnostic error detection mechanisms and if any of these are triggered the program is terminated and cause identified. Post-mortem diagnostic mechanisms can analyse a "hung" communications pattern, and identify both programming errors and errant hardware.

The kernel's ground up NFS client implementation is used to implement the disk system, and greatly enhances the usability of this machine.

6.3. Disk system

The QCDOC disk system has been implemented by writing a custom NFS client as part of the node kernel. A simple model for the QCDOC disk system is used.

Firstly there is a disk system shared between all nodes, and easily accessible by users on the host system. This disk is known to the nodes via the directory `"/host"`. A file on the `/host` filesystem is globally visible to all nodes, and to the host computer.

The second disk system is logically a per-node private parallel disk system, known as `"/pfs"`. This may be thought of as analogous to clusters having a "scratch-disk" on every node. Note that the QCDOC nodes are diskless, and implement the pfs by NFS mounting a number of network attached storage boxes. In fact, each NAS is shared by as many as 256 nodes, each using a per-node unique subdirectory. SciDac software will be provided for moving ILDG data to and from this distributed file system, allowing QCDOC to have very high performance I/O with the slow operations of serialising the I/O carried out offline.

6.4. Partitioning

The six dimensional torus network of QCDOC is put to good use by allowing partitioning of the machine in software. The machine may be sliced

Figure 4. Folding a two dimensional slice of a machine into one periodic application dimension restores the toroidal nature and reduces the dimensionality by one.

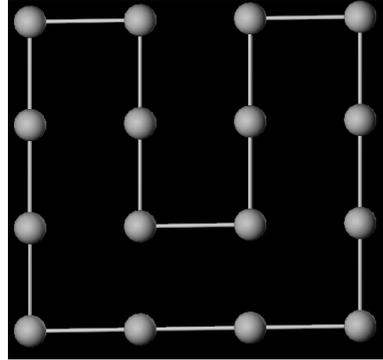
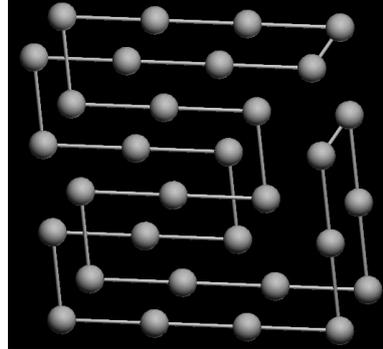


Figure 5. The folding process may be iterated allowing partitions of one to six dimensions. We show the remapping of a three dimensional machine to form a one dimensional application torus.



by software into six dimensional hypercuboids along hyperplanes. Each partition is rendered periodic once more by folding together several of the machine dimensions to form each application dimension which meander non-trivially through these machine dimensions using only SCU links internal to the partition. Figure 4 displays folding a 4×4 machine slice to form one periodic application dimension of length 16. This process may be iterated in orthogonal dimensions and a slightly less trivial case of folding together 3 machine dimensions to form a single periodic application dimension (essentially the space filling loop) is given in Figure 5.

6.5. Application Software

Several common QCD application codes have been ported to and extensively run on QCDOC.

These include the Columbia Physics System, Chroma, and MILC code bases.

6.5.1. Programming environment

We support both the GCC tool chain, and the link compatible IBM xlC compiler suite. C++ templates, iostreams etc... are all supported, and "new" and "malloc" heap allocation requests default to the transient DDR memory discussed below. We have ported the Cygnus "newlib" embedded libc and libm and implemented the O(30) system calls to support this library in our custom operating system. This provides a standard compliant UNIX-like environment for single processes. This single process programming environment is sufficiently standard that few porting issues have occurred.

As is common for MPP machines the QCDOC programming model extends the Unix like environment with libraries for message passing. The message passing extensions take the form of both a C++ interface known as the SCU library, and a SciDac QMP compliant wrapper for this [6,7,2]. The one non-standard area lies with memory allocation, and reflects both the directly addressable embedded DRAM hardware and the noncoherent L1 cache in QCDOC. A "qalloc" routine takes an additional argument specifying memory type. The level-1 cache of the PPC440 CPU is partitioned into transient and normal regions and both communications buffers and large streaming arrays should be allocated as transient for best performance.

7. Performance

Optimised Dirac operators have been implemented in the context of the CPS by UKQCD [8] (Clover, Wilson, DWF) and by SciDac/Columbia [6,7,2] (AsqTad and naïve staggered). These will all be made available in various ways, and the optimised Wilson code has already been interfaced to Chroma/QDP++ [9]. The UKQCD kernels and linear algebra routines are generated by a cross platform assembler generator and scheduler which will be made publically available and will be the subject of a forthcoming publication.

Hybrid MonteCarlo Clover simulations have been run on large machines, both 512 node and 1024 node. Performance as high as 47.5% of peak

Table 1

Double precision performance of QCDOC. Figures are as a percentage of peak. Current machines are running at 420 MHz for 840Megaflop/s peak.

Action	Volume	CG Perf	Dirac Perf
Wilson	2 ⁴	32%	44%
Wilson	4 ⁴	38%	44%
Clover	4 ⁴	47.5%	54%
DWF	4 ⁵	42%	47%
AsqTad	4 ⁴	40%	42%

has been obtained in the inverter for this action during the full HMC. Over 30% is achieved when running from the DDR memory, and the HMC has been run for periods in excess of 48 hours on the recently debugged 1024 node machine. Code optimisations may yield an increase in performance in the future.

The final clock speed has not yet been determined: the current clock speed is 420MHz which reflects a 60MHz improvement that has been obtained since lattice conference by tuning DDR memory timings, and SCU PLL options. A similar additional speed up is possible in the future, and 420 MHz is likely a lower bound. Table 1 displays the current performance of some popular actions, and additional results can be found in [6].

8. Roadmap

The UKQCD and RBRC machines will finish construction in late October. The DOE SciDAC Machine will be completed in March 2005. They all have over 12k nodes, and will sustain over 40% of peak with at least 840 peak Megaflop/s per node. The current 420MHz may be increased as manpower is freed up from machine debugging.

REFERENCES

1. Nucl.Phys.Proc.Suppl.129:838-843,2004
2. eConf C0303241:THIT003,2003
3. Nucl.Phys.Proc.Suppl.119:1041-1043,2003
4. Nucl.Phys.Proc.Suppl.106:177-183,2002
5. Nucl.Phys.Proc.Suppl.94:825-832,2001
6. C. Jung, these proceedings.
7. <http://www.lqcd.org/scidac>, www.scidac.org
8. P. Boyle, in preparation.
9. R. Edwards and B. Joo, these proceedings