QCDPAX — A Parallel Vector Processor Array for Lattice QCD

QCDPAX collaboration
Yoshio Oyanagi
Institute of Information Sciences, University of Tsukuba
Yoichi Iwasaki, Tomoteru Yoshié and Kazuyuki Kanaya
Institute of Physics, University of Tsukuba
Tsutomu Hoshino and Tomonori Shirakawa
Institute of Engineering Mechanics, University of Tsukuba
Tenno-dai, Tsukuba, Ibaraki, 305 Japan

Shingo Ichii
KEK, National Laboratory for High Energy Physics, Oho, Tsukuba, Ibaraki, 305 Japan
Toshio Kawai
Department of Physics, Keio University, Hiyoshi, Kohoku, Yokohama, 223 Japan

ABSTRACT

We have constructed a highly parallel processor array QCDPAX with 432 processing units for the numerical simulation of lattice gauge theory. The QCDPAX adopts the MIMD, two-dimensional nearest neighbor connection and the common bus architecture. Each processing unit has a 32 bit microprocessor MC68020, a floating-point chip L64133, and an LSI for vector operation specially fabricated by the gate array, 2MB of fast memory, 4MB of slow memory, etc. The peak performance of the QCDPAX is 12.25 GFlops and the link update time for subspace heat bath method with 8 hits is 1.8 μsec. The multiplication of Wilson fermion matrix and the spinor vector takes 0.42 μsec/site.

1 Introduction

Computational methods now stand at the forefront of scientific research. Numerical approach naturally requires enormous computer resources. Over the past ten years physicists have employed numerical simulations of lattice gauge thoery on vector supercomputers. The inherent parallelism, however, enables one to perform these calculations on parallel processors.

2 Architecture of QCDPAX

QCDPAX[1] is a highly parallel processor array dedicated to numerical simulation of lattice gauge theory and was designed in University of Tsukuba and manufactured by Anritsu Corp. It is the fifth of PAX series[2].

The QCDPAX project is funded by the Japanese Government with about two million dollars from 1987 to 1989FY. QCDPAX with 432 processing units (PU's) is completed and is running. The full scale QCDPAX with 480 PU's will be completed in a couple of months.

The global architecture of QCDPAX is shown in Fig. 1. 480 (at present 432) identical PU's are interconnected in a toroidal (periodic) two-dimensional nearest-neighbor-mesh, 24×20 (at present 24×18). It is a homogeneous, non-clustered array of PU's.

2.1 2d NNM connection

Since the lattice gauge model is four-dimensional, it might be natural to construct the parallel computer of four-dimensional connection. However, it would require eight communication paths to each processing unit and make it difficult to keep the wide bandwidth of each communication path. Moreover, four-dimensional connection is difficult to implement in our three-dimensional space. QCDPAX adopts the two-dimensional nearest-neighbor-mesh (2d NNM) connection like as the former PAX architecture. It is straightforward to map the four-dimensional lattice onto the two-dimensional array of processors.

2.2 MIMD architecture

QCDPAX is a modified Multiple-Instruction Multiple-Data (MIMD) machine. The MIMD architecture has a variety of advantages over Single-Instruction Multiple-Data (SIMD) architecture. First, we note physical models are commonly full of conditioal branches even in a uniform model like lattice gauge theory. This merits the MIMD structure. Secondly, algorithms are flexible in MIMD machine. Especially, hyperplane incomplete LU preconditioning[3], which is performed in a wave front fashion, cannot be implemented on SIMD machines. Lastly, machine clock need not be synchronized among the PU's in MIMD machiens, where the clock skew gives no problem.

2.3 Floating point operations

The simulation of the lattice gauge theory requires huge amount of floating point operations such as multiplication of complex matrices and random number generation. Each PU has a specialized high speed floating operation system tuned for the lattice gauge simulation.

3 Hardware of QCDPAX

3.1 System configuration

The system configuration of QCDPAX is shown in Fig. 1. It consists of PU array, a host-computer Sun-3/260, a color graphic display and the interface between the host and the PU array (HPI). The host is used to compile and assemble the source program, load the object program into the PU array, start the parallel task, transfer and receive the data to and from the PU array.

A PU shares a two-port RAM with each of its four nearest-neighbors to form a torus. It also shares a two-port RAM(ferry) with the host through the HPI. A common bus connects all PU's and the HPI. The memory of each PU is mapped on the address space of the host-computer. The HPI serves to select which PU(s) is/are mapped to the address space of the host-computer.

The computational results from each PU are outputted through the two-ported RAM and the common bus. The HPI is connected to the graphic display with a video processor to diplay the result on line.

3.2 Processing unit

The single PU of QCDPAX is a single board microcomputer based on the Motorola's MC68020 microprocessor(25 MHz) and LSI Logic's L64133 as an FPU. The local memory is 4 MB with 100 ns 1 Mbit DRAMs and the floating point data memory is 2 MB with 35 ns CMOS static RAMs.

The synchronization register (SYNC) is a 6 bit register. At the synchronization point in the program, each PU writes a code in the SYNC and halts itself. If HPI detects the coincidence of the contents of all SYNC's, it lets all the PU's to resume the operation.

3.3 Floating-point operation system

The QCDPAX utilizes the floating-point processing unit (FPU) L64133 on the market. The LSI Logic's L64133 (run in 69.8 ns) has the peak performance of 28.65 MFlops. It comprises a 32 bit arithmetic logic unit and a 32 bit multiplier concurrently workable. It is a scalar processor, unlike other chips such as Weitek's, which operate in a pipline mode.

We have developped an FPU controller using a gate array chip (LSI Logic's) as the interface between the CPU and the FPU, the sequencer for the fundamental arithmetic calculation and the address generator for vector operations. It also supports microprogram operation to calculate elementary mathematical functions such as $1/x$, $\sqrt{x}$, $\sin x$, $\cos x$, $\exp x$, $\log x$, $\arctan x$, etc. as well as special operations such as the multiplication of 3×3 matrices, random number generation by Lewis-Payne method etc. The writable control storage (WCS) is 32 bit $\times$ 2Kwords high speed (25ns) BiCMOS static memory.

3.4 Installation

The PU array is installed in six cabinets, forming a right hexagon. Each cabinet contains five modules, and each module contains sixteen PU boards, which are connected in a 4×4 array. One PU is installed on a six-layer print board of 367 mm $\times$ 400 mm. The whole PU array forms 24×18 lattice. When boards are added, the array size will become the full size 24×20.

4 Software

4.1 Development of programs

The user of the QCDPAX should prepare two programs: one for the host-computer, the other for the

PU's. The former is written by the language c. The latter is written by a newly developped language psc (parallel scientific c).

The psc program is compiled on the host-computer to a qfa program. The qfa (quick floating assembler) is an assemply language, specially designed for our PU. The user can optimize the code at the qfa level. The qfa program is assembled to the usual assembler and then to the machine code for MC68020. The host program loads the machine code into the PU array before starting the PU's.

While the host-computer supports multi-user operation, only one job can be executed in the PU array. If, while one job is being executed in the PU array, some other job instends to load a program, it is rejected.

4.2 psc

The psc is a c-like language for nodal program in a PU and can describe explicitly the vector processing in a PU as well as the parallel operation by the CPU and the FPU. The vector processing is described by vfor-do statement. An example is given here:

```
vfor( i=0 ; i<n ; i+=1) {
   a[i]=a[i-1]*a[i+1] + b[i-1]*b[i+1] ;
   p[i]=q[i-1] + r[i+1];
} do {
   for( j=0 ; j<m ; j+=1)
     l_ferry[j]=l[j];
}
```

In the psc program listed above, the block after vfor is executed by the FPU and the block after do is executed by the CPU concurrently. The real or complex arrays a, b, p and r are located in the SRAM area. It should be noted that such recurrence iteration is supported in the vfor statement. The do block describes a data transfer from the DRAM area to the ferry. These two blocks are executed in parallel. In the vfor-do statement, either the FPU or the CPU waits until the other finishes the operation.

4.3 qfa

The psc compiler generates the object code both for the CPU and the floating point operation system in terms of the qfa language. The assembler of qfa translates the qfa source program to the assembly language of the MC68020. The qfa code proper to the CPU is nothing but the assembly language itself. On the other hand the code for the FPU is translated to the CPU code (sequence of movl instructions) which writes the instructions of the FPUC into the WCS.

The qfa program for the FPU has better readablity than the translated CPU code, and allows the user an easier tuning of the PU program.

5 Evaluation

5.1 Performance of a single PU

Since the FPU can execute one addition and one multiplication in 69.8 ns, the peak performance in the vector mode is 28.65 MFlops. Nearly maximum performance of 28.3 MFlops was actually measured for the sum of squares of vectors with 500,000 elements on each PU without communication by employing a sophisticated coding. This corresponds to 12.25 GFlops peak speed. The realistic performance, however, is limited due to the following factors.

1. Vectorization ratio of the program.
2. Start-up time of the vector operation.
3. Load/store time around the FPU.
4. Data transfer between registers in the FPU.
5. Data transfer between the DRAM and SRAM.

Table 1 lists the performance of various basic loops. The scalar performance, limiting vector performance (extrapolation to infinite vector length) and the half performance vector length $n_{1/2}$ [4] are given. A Tuning is added to the vector operation at the qfa level.

Basic loops	scalar	vector	$n_{1/2}$
z[i]=x[i]+y[i]	0.176	4.39	156
z[i]=a*x[i]+b	0.402	9.56	82
s=s+x[i]*x[i]	0.388	22.67	267

Table 1. Performance of basic loops (MFlops)

The elementary functions are microprogrammed on the WCS. They can be called both in scalar and vector modes. Table 2 lists the execution time of elementary functions in the vector mode. For comparison, the data for Sun-3/260 compiled with optimization and 68881 option are also given. Note that double precision results are obtained in the Sun case while our FPU has only single precision operations.

functions	QCDPAX PU	Sun-3/260
$1/x$	0.908	14.0
$1/\sqrt{x}$	1.421	40.0
$\sin x$	2.114	52.0
$\cos x$	2.097	52.0
$\arctan x$	2.529	56.0
$\exp x$	2.235	56.0
$\ln x$	1.885	40.0

Table 2. Execution time of elementary functions (μsec)

5.2 Performance of parallel processing in lattice QCD

Parallel processing inherently includes the following overheads.
1. Synchronization time.
2. Communication overhead between the PU's.
3. Idle time due to uneven workload.

These overheads should be sufficiently small. For the QCDPAX, the time for the synchronization of all PU's is 2.5 μsec. The broadcast of 1024 byte data from one PU to all PU's is 560 μsec. The speed of data transfer between neighboring PU's was measured to be 4.23 MB/s.

The link update time for the quenched approximation on $48 \times 48 \times 24 \times 18$ is 1.787 μsec/link at $\beta = 5.69$ by the subspace heat bath method with eight hits. The communication overhead in this case is measured to be 0.443 μsec/link, which is about 25% of the total time.

The multiplication of the Wilson fermion matrix and the quark field vector on the same lattice size is 0.42 μsec/site, which corresponds to 5 GFlops effective performance.

The QCD programs are written in psc language with the help of appropriate preloaded functions.

Acknowledgement

This work is partly supported by the Grand-in-Aid for Specially Promoted Research of Ministry of Education , Science and Culture of the Japanese Government (No. 62060001).

References

[1] Y. Iwasaki, T. Hoshino, T. Shirakawa, Y. Oyanagi and T. Kawai, Computer Physics Communications **49**, 449 (1988). T. Shirakawa, T. Hoshino, Y. Oyanagi, Y. Iwasaki, T. Yoshie, K. Kanaya, S. Ichii and T. Kawai, "QCDPAX — An MIMD Array of Vector Processors for the Numerical Simulation of Quantum Chromodynamics", pp. 495–504 in *Proceedings of Supercomputing '89*. Y. Iwasaki, K. Kanaya, T. Yoshie, T. Hoshino, T. Shirakawa, Y. Oyanagi, S. Ichii and T. Kawai, "Status of QCDPAX", talk presented at the International Symposium of Lattice Field Theory LATTICE 89, Sept. 1989, Capri, Italy.

[2] T. Hoshino, IEEE Computer **19**, No.5, 68 (1986). Further references therein.

[3] Y. Oyanagi, Computer Physics Communications **42**, 333 (1986).

[4] R. W. Hockney and C. R. Jesshope, *Parallel Computers* (Adam Hilger, Bristol, 1981).

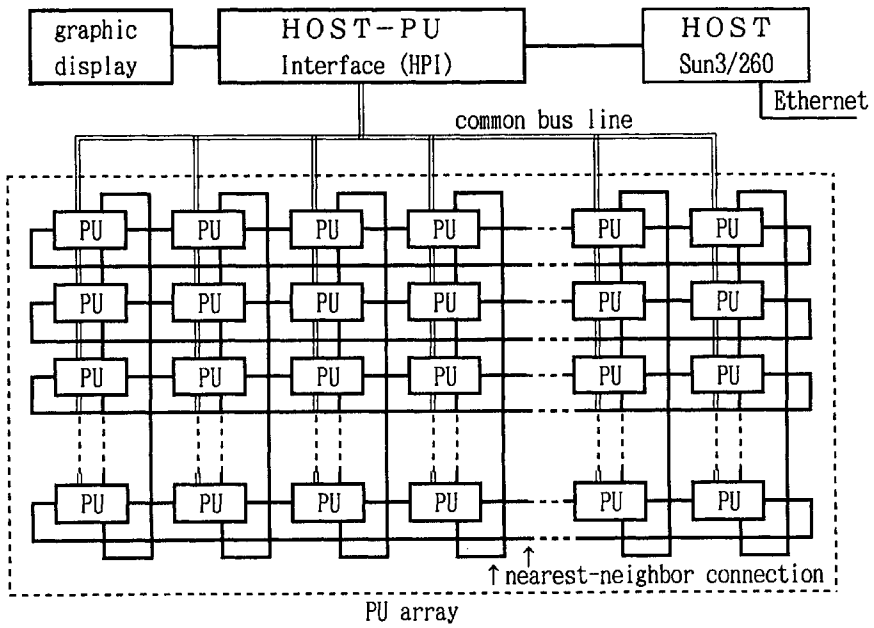

Fig. 1 The architecture of QCDPAX

Q. **W. J. Zakrzewski** *(Univ. Durham)*: Will this machine be able to run several jobs at the same time (share CPU)?

A. **Y. Oyanagi**: No.