ANL-HEP-CP-85-64

CONF-8506158--1

ANL/HEP-CP--85-64

DE85 014988

THE ANL/STAR PROJECT: A NEW ARCHITECTURE FOR LARGE SCALE THEORETICAL PHYSICS COMPUTATIONS

A. M. Rushton
Argonne National Laboratory
Argonne, Illinois

INTRODUCTION

Research in many areas of theoretical physics, such as lattice gauge theory, is increasingly limited by the cost and availability of computing power. Existing machines approach speeds on the order of 1 Gigaflop at a cost of tens of millions of dollars; hence many research groups lack the resources to acquire such a machine, or the problem cannot be run in a sufficiently short time (on the order of six months). Also, the rate of increase in uniprocessor speed has been slowing as we approach the limits of scaling down silicon chip technology. These factors have led our group at Argonne National Laboratory to explore a different approach to supplying increased computational speed and cost-effectiveness for such problems - one which uses tightly coupled multiple processors with shared memory and a sophisticated array processor as the node element.

OVERVIEW

Our project consists of two phases, each of which has goals of substantial physics content on its own. In Phase I, we have selected Star Technologies' ST-100 as the array processor for the prototype coupled system and have installed one on a Vax 11/750 host. Our goals with this system are to institute a substantial program in computational physics at Argonne based on the power provided by this system and thereby to gain experience with both the hardware and software architecture of the ST-100. In Phase II, we propose to build a prototype consisting of two coupled array processors with shared memory to prove that this design can achieve high speed and efficiency in a readily extenaible and cost-effective manner. This will implement all of the hardware and software modifications necessary to extend this design to as many as 64 (or more) nodes. In our design, we seek to minimize the changes made in the standard system hardware and software; this drastically reduces the effort required by our group to implement such a design and enables us to more readily incorporate the companies' upgrades to the array processor. It should be emphasized that our design is intended as a special purpose system for theoretical calculations; however it can be efficiently applied to a surprisingly broad class of problems. I shall discuss first the architecture of the ST-100 and then the physics program being currently implemented on a single system. Finally I will present the proposed design of the coupled system.

ST-100 ARCHITECTURE

The architecture of the ST-100 is shown in Fig. 1. The array processor actually contains three separate computer subsystems: the Control Processor (AP), the Storage Move Processor (SMP), and the Arithmetic Control Processor (ACP). Data is transferred between the host computer memory and the array processor main memory. The SMP controls data movement between main memory and the data cache; it is also used for integer and logic operations, such as random number generation. All floating point arithmetic operations are controlled by the ACP and the data flow is from cache to the Arithmetic Logic Unit and back to cache. The ALU contains two adders, two multipliers, and a divide/square root section. The adders and multipliers are three stage pipelines requiring three clock cycles; however the divide/square root section

is non-pipelined and requires 13 cycles. So the old lesson from other machines also applies here - multiplication is fine, but division really slows you down! The Star uses a 32 bit word size and a 40 nsec cycle time. Main memory can be extended to 8 Megawords, although cache memory size is fixed at 48 Kwords. Our machine presently has 2 Megawords of main memory.

Writing software for the ST-100 requires a significant logical reorientation by the programmer. The Star itself is programmed in a subset of Fortran-77 called APCL; the program is termed a "process". However, most of the computation is performed by "macros" called by the process, which are essentially assembly language subroutines written for either the ACP or the SMP. Star Technologies' supplies libraries of standard macros for both processors; however in a typical physics problem, the user must write macros to efficiently implement his own algorithm. In effect, one must write much of the program in assembler, although this works out in practice much better than one might think. The processes are called from a host program running on the host computer and computations which are performed only a few times in a program are normally done in the host program. Only the computationally intensive portions of the program are done on the array processor. Also, the macros encourage a modularization of the program structure which facilitates the reuse of code in similar problems. In practice, support for running processes on the ST-100 requires only 1-2 per cent of a host Vax 11/750.

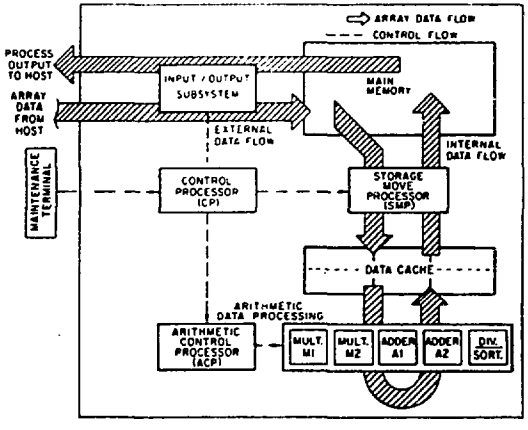

Figure 1. ST-100 Architecture and Major Data Flows.

The most serious obstacle to production use of an ST-100 is the difficulty of writing programs in microcode. The optimal choice of algorithm is changing sufficiently rapidly in some areas such as lattice gauge theory with fermions that it consumes significant manpower to keep programs current. Hence the user groups have expended considerable effort on developing software tools to aid in code development. Star has also contracted for the development of a Fortran compiler for the ST-100, and this may change the situation markedly.

PHYSICS PROGRAM FOR THE SINGLE ST-100

Our array processor has been operational since approximately last March 22. Since then, our initial computational program has consisted of problems in the

areas of lattice gauge theory, condensed matter physics, and molecular modelling. (As you see, we are not confined strictly to high energy physics problems, although the driving force behind the project has been the computational needs of lattice gauge theory.) We are also working with a collaboration based mainly at the University of California at the Santa Barbara and San Diego campuses. They also has an ST-100 and we share software tools and utility programs; some of their projects are also run on our system. Our host computer is connected to a Decnet with approximately 120 nodes, so most users are able to work conveniently while 2,000 miles from the array processor!

Our principal lattice gauge theory projects deal with both pure gauge fields and with fields with coupled fermions. The largest effort is being undertaken by Don Sinclair in collaboration with John Kogut. They are studying two problems: the calculation of the low lying hadron mass spectrum by calculating the long distance behavior of gauge invariant propagators in channels with the desired spin and flavor, and secondly, the thermodynamics of lattice QCD fermions present. In the conjugate gradient routines for matrix inversion (which are central to the computation), Sinclair estimates that he achieves a speed of slightly greater than 50 Mflops; he expects comparable results in the rest of the program.

The California contingent (D. Touissant, R. Sugar, S. Gottlieb, and others) work on both condensed matter and lattice gauge theory problems, which emphasizes the similarities of the formalism in the two areas. In the lattice gauge area, they have studied the phase behavior of both a pure U(1) gauge field (for electromagnetism) and also an SU(3) gauge field at finite temperature. Using the Metropolis method with a pure SU(3) field, they can perform 10 updating hits on a link in 100 microseconds, with additional hits requiring 4 microseconds each. This speed is comparable to a Cray-1 programmed in Fortran. Due to our relatively small (2 Megaword) memory, they use our machine to test algorithms on small lattices and do production runs on their 4 Megaword machine.

They have also studied the behavior of a superconductor in an electromagnetic field (using a modification of an Abelian Higgs model program) and are currently studying a model of a polymer, in which which fermions are free to move along a chain in one dimension, and the chains are coupled by Coulomb interaction with the neighboring chains.

In a completely different area, Clausing, Hagstrom, and McConnell have been developing software for molecular modelling on the ST-100. The problem is to determine the equilibrium configuration of a large molecule (on the order of 100 to 10,000 atoms) by calculating the internal forces starting from an arbitrary initial configuration and numerically approximating the motion of the atoms until the molecule reaches a minimum energy configuration. They consider the forces between non-bonded atomic pairs up to an arbitrary cutoff distance using a Lennard-Jones potential and the forces due to bending of chemical bond angles (between triplets of atoms) and torsional forces. They have obtained the timings shown in Table 1 for the indicated standard calculations using programs written in microcode.

The problem of comparing relative speeds of machines with very difficult architectures is a well known one; one reasonable method is to compare computational times for standard problems, and by that standard, these values are several times faster than similar calculations reported on a Cray-

Table 1

TIMING	TASK
0.97 sec/random pair**	Non-bonded pair interaction
1.98 sec/random triplet*	Bonded three-body interaction
5.14 sec/random four*	Proper dihedral four-body interaction
5.26 sec/random four*	Improper dihedral four-body interaction
0.23 sec/atom**	Verlet integration time step
3.20 sec/random pair (est.)	Bonded pair length correction
0.16 sec/pair (est.)	Create non-bonded interacting pair list

*24.3 Mhz clock; **25.8 Mhz clock

1. PROPOSED MULTIPROCESSOR SYSTEM

The preceeding gives you the sense of what can be done with a single array processor; however, we propose to build a far more powerful system composed of an ensemble of array processors, each of which communicates via shared memory with its eight nearest neighbors in a three dimensional array. As we saw earlier, the arithmetic unit of the ST-100 works on data transferred to local cache memeory, so the shared memory modules are used only for sharable data and program storage. We propose to use modified Dataram 32 Mbyte memory modules for these units to reduce costs. The architecture is shown schematically in Fig. 2, which is drawn for the two-dimensional analogue of our architecture; the reader should extend this to three dimensions.

SCHEMATIC OF PROCESSOR AND MEMORY
CONNECTIONS FOR 2-DIMENSIONAL MACHINE

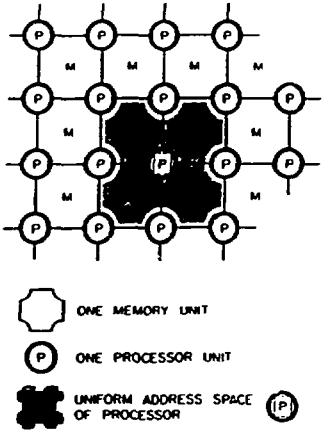

Figure 2. Schematic of Proposed Multiprocessor System.

This architecture incorporates a number of unique and/or noteworthy features to deal with the major problems of multiprocessor systems:

a) The shared memory modules form a uniform address space over the 8 nearest neighbors. Hence access to data belonging to an adjacent node is as fast as references to the local node.

b) This architecture is a true multiple instruction stream, multiple data stream (MIMD) system. Each node is capable of executing independent programs and each has a separate connection to the host computer for down-loading and synchronization.

c) Each processor can block access by other processors to an adjacent memory module; hence read-modify-write cycles can be indivisible. This prevents the "read before writing" problem which can occur in many systems, where one processor may read values from memory before they have been up-dated by another processor. Access requests to a locked module are queued by the hardware and serviced when the module is released.

d) A unique and potentially powerful aspect of our system is the reconfigurability of the mapping of the node processors' uniform address space into the physical addresses of the eight adjacent memory modules. Three bits are selected by the user as "key bits"; they will determine the mapping between the address space of one processor and the entire address space of 8 adjacent memory modules. This gives the capability of accessing regular patterns of non-adjacent memory locations as fast as adjacent locations.

We propose initially to build a 2 node prototype with this architecture and are presently negotiating a joint venture agreement with Star Technologies' to do this. This will demonstrate all of the hardware and software changes necessary for a multiple node system and could be easily extended with minimal changes to as many as 64 nodes. (This is actually a rather soft limit.) Our experience with this will be used as a basis for larger systems.

At this point, you may well wonder what efficiency we anticipate from the additional processors. Let's consider the specific case of lattice gauge theory, which is of particular interest. Here, we have a link variable, typically an SU(3) matrix, associated with each site in a 4 dimensional lattice. For the 3 dimensional geometry of our system, we can divide the data space for N processors by dividing the set of (x,y,z) points into N subsets. With each such point is then associated all of the t axis sites. In calculating the properties of pure gauge fields, one needs for a given lattice site, only data for that site and sites one unit away in any dimension; if all processors execute the same patterns of memory reference, the efficiency of additional processors is essentially 100%, and this will also apply to the broad class of problems where such local data can be read from one array, processed by the CPU, and written to an adjacent array. However, when we add fermions to the lattice, the problem becomes much more difficult. It is possible to integrate out the fermion fields by hand, leaving an integrand which involves the inverse and determinant of a large matrix for the bosonic integrals. The dimensionality of this matrix is at least as large as the number of spacetime points in the lattice. The techniques for solving such large sparse matrices efficiently on ensemble systems are an area of active study, but, unfortunately I can't yet quote results for our architecture.

In summary, we have obtained effective speeds from a single ST-100 array processor comparable to a Cray-1 on problems of substantial physics interest and plan to continue the use of this machine as a production facility. We will also be developing software tools to make coding for it easier. We further propose to build an easily extensible 2 processor prototype of the ensemble system to demonstrate the potential of this architecture for achieving cost-effective computing for large problems.