Fermi National Accelerator Laboratory

FERMILAB-Conf-89/217

General Purpose Computers in Real Time *

Joseph R. Biel
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Illinois 60510

September 18, 1989

* Presented at ECFA Study Week on Instrumentation Technology for High Luminosity Hadron Colliders, Bellaterra, Barcelona, Spain, September 14-21, 1989.

Operated by Universities Research Association, Inc., under contract with the United States Department of Energy

General Purpose Computers In Real Time

Joseph R. Biel
Fermilab Computing Division
September 18, 1989

Introduction

I see three main trends in the use of general purpose computers in real time. The first is more processing power. The second is the use of higher speed interconnects between computers (allowing more data to be delivered to the processors). The third is the use of larger programs running in the computers. Although there is still work that needs to be done, I believe that all indications are that the online need for general purpose computers should be available for the SSC and LHC machines.

More Processing Power

The history of computers is a history of vast increases of computing power with a simultaneous decrease in price. At present, the greatest contributor to the continuation of this trend is found in the development of RISC (Reduced Instruction Set Computers). One manufacturers projections of RISC performance are shown in figure 1.

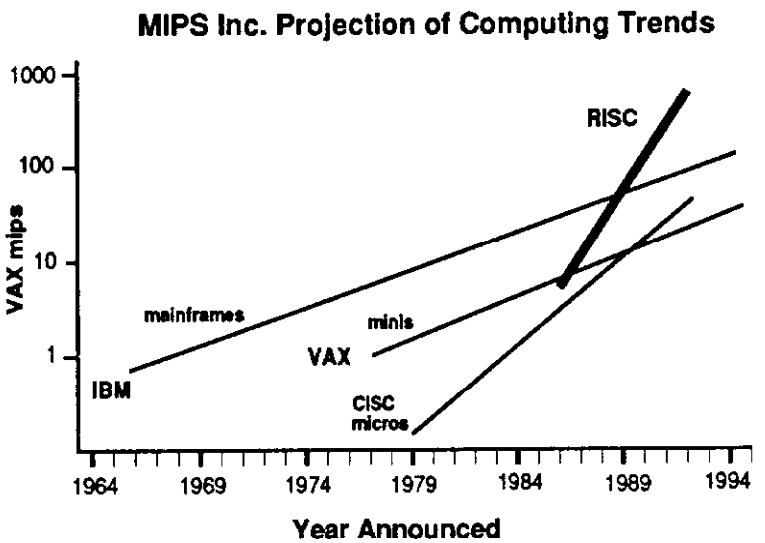

Figure 1. MIPS Computer Systems, Inc. Projections

The basic ideas behind RISC are to construct a microprocessor with

- A relatively small set of machine language instructions
- Machine instructions that provide needed functions only with no redundancy
- An instruction set that allows a simple, high speed implementation
- A highly optimized compiler well matched to the instruction set

A wide variety of RISC microprocessors have been designed. Among those available commercially are the MIPS R2000/R3000, the Sun SPARC, the Motorola 88000, the AMD 29000, the Intel 860, and the Intergraph Clipper. Among those companies with proprietary chips are Apollo, IBM , and Hewlett-Packard. Approximate performance values for some of these chips are

- MIPS R3000 (25 MHz clock) 20 VAX MIPS
- SPARC (16 MHz clock) 10 VAX MIPS
- Intel 860 (40 MHz clock) 14 - 17 VAX MIPS
- Motorola 88000 (20 MHz clock) 14 - 18 VAX MIPS

where a "VAX MIP" is the power of a VAX 780. Some of the complete computer systems using RISC microprocessors are manufactured by DEC, SUN, Apollo, and Silicon Graphics. It is rumored that IBM will soon make some major RISC product announcements. As a specific example of a RISC computer system, consider the Silicon Graphics model 4D/280S. One possible configuration for this machine is

- Eight cpu boards each containing a MIPS 25MHz R3000 RISC chip
- 64 MBytes of memory
- 256 KBytes local cache memory for each processor
- VME bus (10 MB/sec now, 30 MB/sec early 1990)

This configuration has a list price of about $230,000 which corresponds to about $1,400 per VAX MIP.

The Fermilab Computing Research and Development department is working on a RISC VME module. Like the Silicon Graphics machine, it also uses the MIPS R3000 microprocessor. It contains 8MB of memory, 32KB of both instruction and data cache, 256KB of EPROM, and a VME master/slave interface. It is constructed on two boards which together fit within a single width VME slot. A secondary data bus designed at Fermilab, called the "X bus", is also included. This provides an alternate DMA path into memory at a transfer rate of 40MB/sec. This can, for example, be used to implement a high bandwidth level 3 trigger. In such a system, a set of processors are sent events in parallel over their X buses. The total bandwidth for this event filling process is thus 40MB/sec times the number of processors being used. Each module examines the event stream that it receives and rejects the vast majority of them. If an event passes the trigger requirements, that event is read out over the much smaller bandwidth VME bus.

Plans for future enhancements of MIPS chips (not including gallium arsenide plans) include

- MIPS R3000
  - 33 MHz version in 1989
  - 40 MHz version in 1990
  - 60 MHz version in 1991
  - 100 MHz version in 1992
- SPARC
  - 33 MHz version in 1989

40-50 MHz version by 1990

- Intel i860
  50 MHz version soon
- Motorola 88000
  100 MIPS 5-chip set by 1991

Several gallium arsenide RISC implementations of RISC are also being worked on. TI and CDC have a chip that runs at 68 MHz and are working to get it running at 200 MHz. McDonnel Douglas Astronautics Corp. has a chip that runs at 60 MHz and are working to get it running at 200 MHz. Prisma, a startup company in Colorado, is working on a GaAs RISC system that will run at 250 MHz.

The TI/CDC and McDonnel Douglas chips both require external GaAs cache memories to allow 1 memory access every 5ns. Both also require clever compilers to keep pipelines full despite branches.

Higher Speed Interconnects

Computers constructed with the high speed RISC chips already being developed or being developed must also be faster than those currently in use. Fortunately, there are a number of new interconnection mechanisms that are being developed. The Scalable Coherent Interface (SCI) is a standard being developed to allow a 1GB/sec connection to be made between processors. The standard will allow up to 16K processors in a system. Future Bus is a new bus standard being developed. Current plans are to support transfers of 200MB/sec with a 32-bit wide data path and to support higher rates with wider data paths. Fermilab is working on an "event builder switch". This device has as its inputs, multiple streams of data -- each stream from a subdetector in an experiment. It has as its outputs multiple streams of assembled events -- each event sent directly into the memory of a processor. These processors are general purpose computers used as an upper level trigger farm.

Future Online Use Of General Purpose Computers

The expected event rates in the next generation of hadron colliders must be massively filtered to get to an acceptable event recording rate. One scenario for achieving this is shown in figure 2. This design uses a high level processor farm to provide a factor of 10 to 100 reduction in the event rate. The farm would accept 10,000 events per second which at an estimated event size of 1MB requires a bandwidth of 10GB per second. Current estimates for checking if an event passes the filtering requirements are 10 to 100 seconds on a processor with a power of 1 VAX MIP. The farm must, therefore, have a computing power of from 100,000 to 1,000,000 VAX MIPS. This is an impressive amount of general purpose computer power. There is at least one project already underway, the Intel/DARPA Touchstone project, to build a general purpose computer with a power of 100,000 VAX MIPS.

Even though it is theoretically possible to design a complete high luminosity hadron collider trigger without a large general purpose computer farm, there are important advantages in having such a farm. The programs running in the farm can examine the entire event using the full power of FORTRAN (or

any other desired high level language). These programs can exactly duplicate algorithms tested in offline computers. In fact, the experiment can use an offline farm that uses the same RISC microprocessor. In this case, moving a program between offline and online is especially easy. Another advantage is that the filtering algorithms can be discussed with the entire collaboration in terms of a FORTRAN program rather than some obscure trigger processor programming technique.

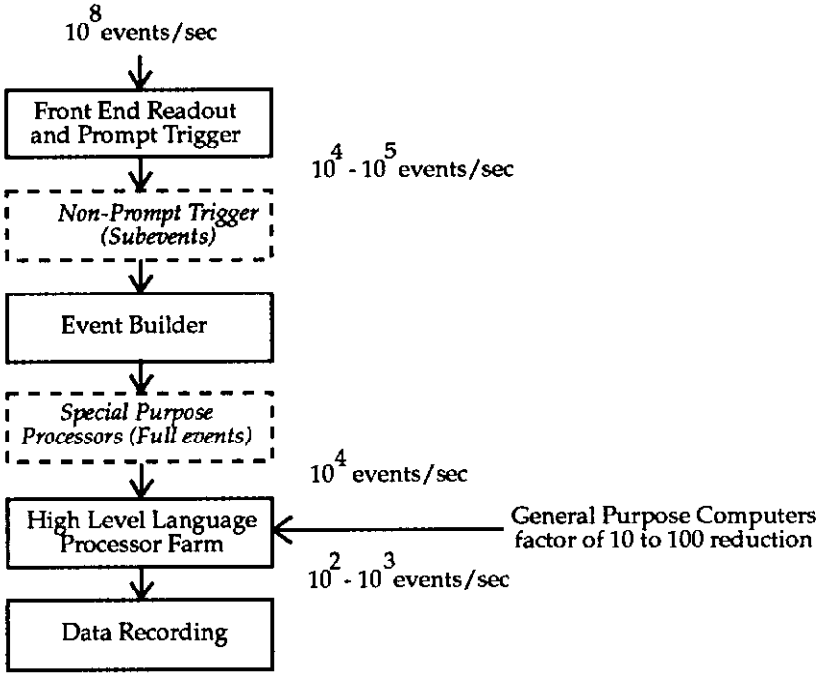

Figure 2.

In order to have a online farm of fully general purpose computers, it is necessary that the programs running in the computers have a complete operating system environment. In particular, the programs should have the usual FORTRAN access to disk files and terminal input/output. This allows program development to be done for an online farm the same way it is done for an offline computer. Initialization data files can be read from ordinary disk files using FORTRAN OPEN and READ statements. Programs can be debugged on the actual farm hardware by running a normal terminal session with a symbolic debugger. If a program crashes, the operating system can make its usual crash dump file. This file can then be examined later with a crash dump analysis program.

It is not necessary to have serial lines and disk drives connected to each processor. If the processors run UNIX, it is easy to make peripheral connections over a network. Serial connections can be established with telnet and disk connections can be established with NFS and ftp. For processors that share the same high speed bus (e.g VME) can make a network connection directly over that bus. For processors that do not share such a bus, a network connection can be established over Ethernet or FDDI.

There are some potential disadvantages to running a full operating system on the online processors. First, the processor modules will be somewhat more

expensive because they may need more hardware features to run the operating system. These features range from the simple (i.e. perhaps an onboard time-of-day clock chip) to the complex (e.g. a VME bus interface in order that VME disk and Ethernet controllers can be accessed). Second, more memory will be needed to hold the operating system. Third, the booting of the farm is likely to be more complex with a full operating system. Fourth, the operating system itself must be "ported" to the processor board. This includes getting the appropriate license for the operating system.

Larger Programs

The filtering programs running in an online processor farm will probably be longer and more complex than those being used now. How to write these long programs is a very important challenge. It will become more important that programs be properly designed. The tools of structured analysis and structured design (SASD) are one way to help write the programs. Another possibility is the use of new programing techniques such as "object-oriented programming" (OOP).

FORTRAN is still the most used language in HEP, but it may be of great benefit to switch to a more modern language. If UNIX is the operating system being used, the C language is a good choice because UNIX is written in C. Historically speaking, it has been easier to get a good C compiler for a new microprocessor than a good FORTRAN compiler. The use of C also allows a natural progression to be made to the object-oriented languages C++ and Objective C.