Fermi National Accelerator Laboratory

FERMILAB-Conf-85/94
2380.000

THE FERMILAB ADVANCED COMPUTER PROGRAM MULTI-MICROPROCESSOR PROJECT*

T. Nash, H. Areti, J. Biel, G. Case, A. Cook, M. Fischler,
I. Gaines, R. Hance, D. Husby, and T. Zmuda

Advanced Computer Program

June 1985

*Presented at the 1985 Computing in High Energy Physics, Amsterdam, Netherlands, June 25-28, 1985.

The Fermilab Advanced Computer Program Multi-Microprocessor Project

T. Nash, H. Areti, J. Biel, G. Case, A. Cook,
M. Fischler, I. Gaines, R. Hance, D. Husby, T. Zmuda
Fermilab, Batavia, Illinois 60510

Fermilab's Advanced Computer Program is constructing a powerful 128 node multi-microprocessor system for data analysis in high-energy physics. The system will use commercial 32-bit microprocessors programmed in Fortran-77. Extensive software supports easy migration of user applications from a uniprocessor environment to the multiprocessor and provides sophisticated program development, debugging, and error handling and recovery tools. This system is designed to be readily copied, providing computing cost effectiveness of below $2200 per VAX 11/780 equivalent. The low cost, commercial availability, compatibility with off-line analysis programs, and high data bandwidths (up to 160 MByte/sec) make the system an ideal choice for applications to on-line triggers as well as an offline data processor.

INTRODUCTION

The need for computing in high-energy physics has expanded to such an extent that it is no longer possible to do all the computing that is necessary on conventional mainframes. Single experiments amass tens of millions of events, each of which requires several seconds of mainframe CPU time. The turn-on of the new collider experiments like CDF and DO at Fermilab and the LEP experiments, all with large computing appetites, further increases the need for less expensive computing power.

Fermilab has established the Advanced Computer Program (ACP) [1] to research and develop more cost-effective and productive ways for high-energy physicists to use computing. The group's first project takes advantage of the event-oriented nature of the experimental high-energy physics computing problem to produce a powerful and efficient parallel processing system. Based on large arrays of commercial 32-bit microprocessors programmable in Fortran, this will provide very cost-effective computing for data analysis. In this paper we will describe this initial project and its on and off line applications.

DESIGN GOALS

The primary design goal of the ACP multi-microprocessor system is to maximize cost effectiveness, in terms of useful computing /dollar. This goal can be met by using two basic strategies: First, high-volume commercial microprocessors are the computing engines of the system. Such processors provide extremely high cost-effectiveness and are programmable in high level languages. Second, at later stages of the project, the CPUs will be augmented with custom "hardware subroutine" co-processors. The original aim was to achieve, without co-processors, an equivalent computing power of better than 1 VAX 11/780 for $5000. This goal now appears likely to be exceeded by more than a factor of two.

Equally important is the goal of ease of use, or user-friendliness. Special purpose processors for particular applications can be designed to run almost arbitrarily fast, but these are very inflexible and difficult to program and commission. We require that the processors of our system be programmable in Fortran-77, the standard for application programs in high-energy physics. The user interface for program development, debugging, etc. must be at least as friendly as that provided by the VAX VMS operating system, another widespread standard in high-energy physics. Finally, the user must be able to transport a program already working on a VAX or similar uniprocessor system to the ACP system with a minimum of effort, without extensive program conversions.

The third design goal is flexibility and modularity. The ACP system consists of building blocks that can be easily reconfigured to meet the specific needs of different users. Different users may require more or less memory or different amounts of CPU power per I/O bandwidth. The modular approach also guarantees that the system can be easily upgraded as newer and more efficient processors become available. This allows the system to take advantage of the most advanced and cost-effective components available from industry at any time without re-engineering the entire system. This also means that the system should not be totally reliant on any single vendor for critical components (such as the CPU chip or board).

The fourth design goal is that the system should be easily copied and maintained by non-experts. Users anywhere in the scientific community should be able to build and configure a system for their own needs without extensive involvement by ACP personnel. This means that the system should be assembled as much as possible out of commercially available components.

SYSTEM ARCHITECTURE CONCEPTS

We are not attempting to build a general purpose computer to satisfy all computing needs, or even all the needs of high energy physicists. Many computing applications will still require general purpose mainframes with their sophisticated operating systems and full complement of peripheral devices. However, the large majority of computing cycles in high-energy physics are devoted to experimental event reconstruction where the same program is run on many millions of uncorrelated and independent events. The structure of the problem makes a trivial parallelism possible, giving different events to each of a large ensemble of processors with little or no interprocessor communication required.

This is an architecture which maintains its high cost effectiveness because it has no need for complicated shared memory or synchronization mechanisms that a fully general purpose parallel processor would need. The two other high-energy physics problems that will require large amounts of CPU time, accelerator simulations and lattice gauge theory calculations, also have simple parallelism inherent in the problems themselves. These problems are well matched to arrays of simply connected cost-effective microcomputers. In fact, it has been shown [2] that many other scientific computing problems can be well solved with such systems. We will discuss here only the architecture pertinent to the reconstruction problem. However, we intend to take advantage of the flexibility of our system in future research on grid and ring architectures.

This simple event oriented architecture consists of a large array of processing nodes all connected to a single host processor. The host delivers events to each of the nodes. When a node completes processing an event, the host will fetch that event from the node and deliver it a new one for processing. The core of the system is the individual processing node. It consists of a commercial 32-bit microprocessor, floating point coprocessor, and enough local memory (1-16 MBytes) to contain an entire program. Each node sits on a crate global bus and on its own private local bus. The global bus is used to download code and data to

all the nodes, and (in the future) for the nodes to access any crate global resources. The local bus allows each node to access its own memory without interfering with any of the other nodes. Customized coprocessors and nearest neighbor interconnection interfaces for use in grid structures will also be added, in the future, to the processing nodes. The computing power of a node, without special coprocessors, approaches that of a VAX 11/780 for physics problems in Fortran.

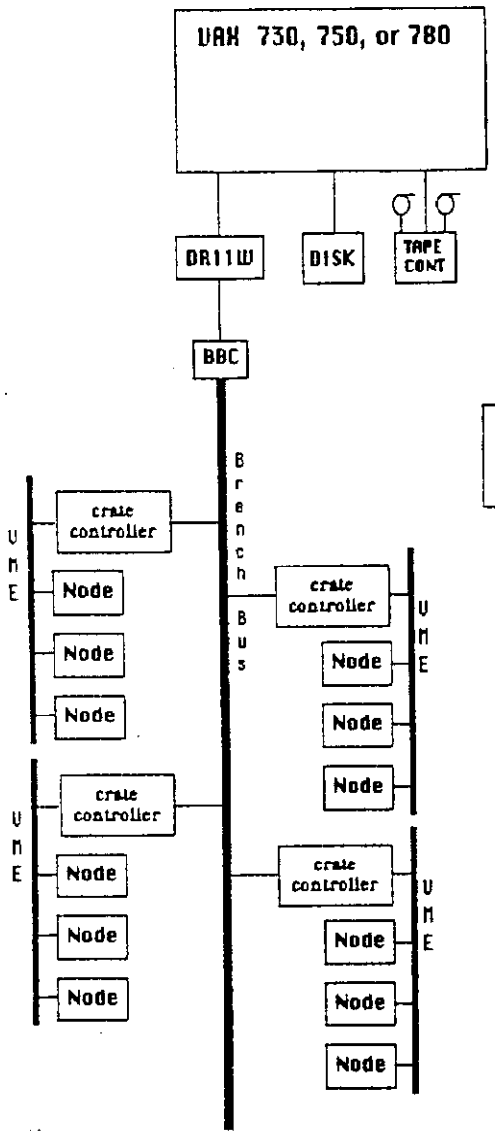

Fig. 1 ACP system controlled by a large VAX as root.

SYSTEM HARDWARE CONFIGURATIONS

The modularity of the ACP software and hardware allows configurations of varying complexity to support different performance and I/O requirements. A standard arrangement of nodes for event oriented data analysis is shown in Fig.1, which also shows other components of the initial ACP system. The nodes reside in standard commercial crates. Each crate has a single crate

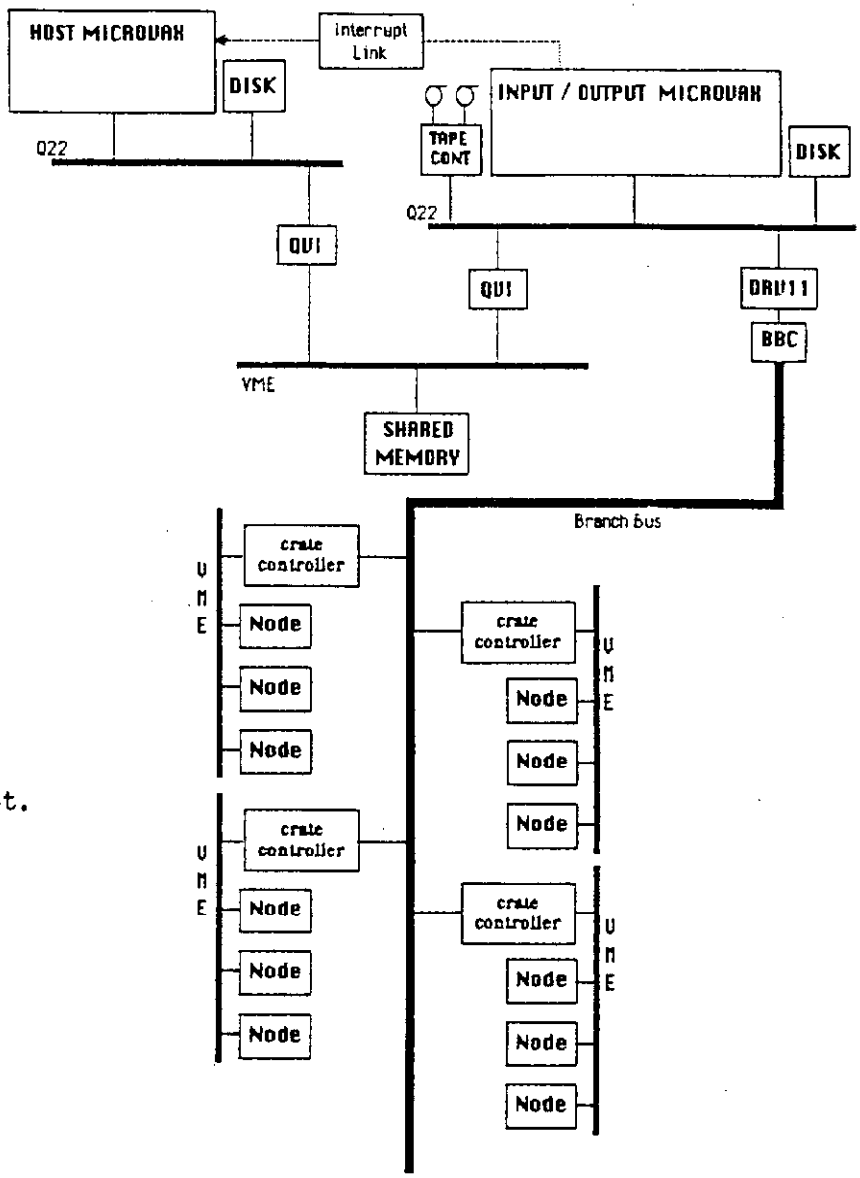

Fig. 2 ACP system controlled by two MicroVAX IIs.

controller which interfaces the crate to an ACP designed Branch Bus which connects all the crates. The first crates will be VME, but Multibus II, Futurebus, BI-bus or other 32 bit buses will be supported with a modified crate controller, when appropriate for a particularly cost-effective CPU. The Branch Bus is a simple single master RS 485 (differential TTL) bus, optimized for high speed 32-bit block transfers. The Branch Bus transmission scheme has been tested at speeds up to 40 MBytes/sec (one word every 100 nsec), and is intended to be operated at 20 MBytes/sec. At the top of Fig.1 is the "root" of the tree-like system, where the user's host program runs. In this simplest implementation example the root consists of a standard VAX with tape drives, interfaced to the Branch Bus through a DR11-W DMA interface and a Branch Bus Controller (BBC). An entry level system of this type, attached to an existing VAX 11/780 or 11/750 installation, will cost less than $50,000 for over 15 VAX 11/780 power in one crate of nodes.

Higher performance versions of the ACP system are shown in Fig.2 and 3. In Fig.2 the root has been expanded. It is here based on two MicroVAXes. One handles tape I/O and Branch Bus communication, allowing the other MicroVAX to devote all its CPU power to the user's host program. The two MicroVAXes

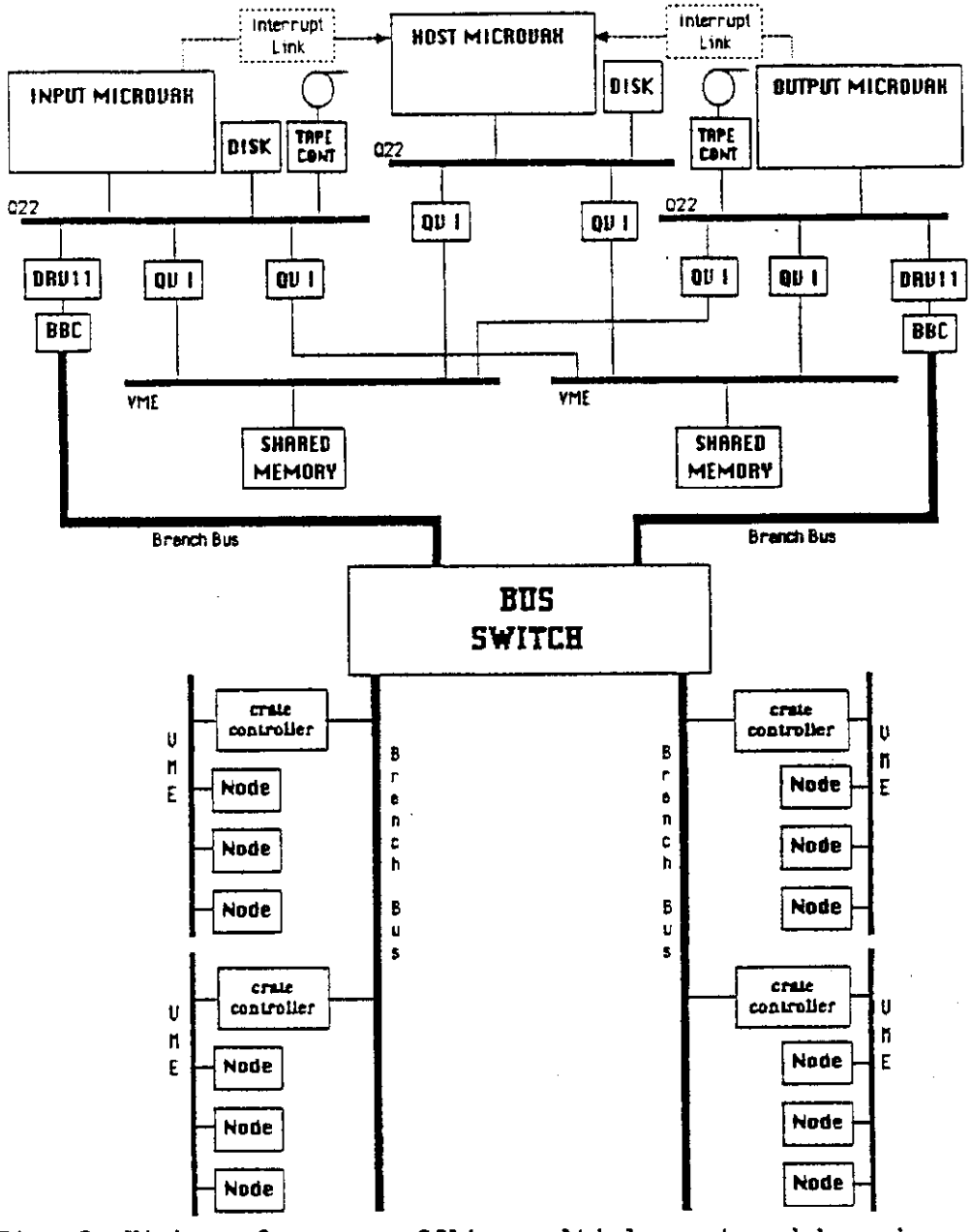

Fig. 3 High performance offline multiple root and branch ACP system with a switch controlled by three MicroVAXes.

communicate through shared Q-bus memory residing in a VME crate using Q-bus to VME interfaces (QVI). The system shown in Fig.3 adds an additional MicroVAX for output, a bus switch, and a second Branch Bus so that an event can be sent to one node on one of the Branch Buses at the same time that a second event is being fetched from a different node on the second bus. The switch allows up to 8 separate Branch Buses to be in simultaneous operation, reaching an aggregate bandwidth of 160 Mbytes per second. Such a system can thus easily support any presently conceived on-line application or off-line applications with future higher speed mass storage peripherals (for example, laser disks). Fig.3 represents the system presently under construction.

SOFTWARE SUPPORT

The user's main software task is to split an existing uniprocessor program into two pieces, one which will run on the host and a second which will run in parallel on all of the nodes. The host portion contains all the input and output: reading raw events from, and writing processed events to, tape, reading calibration files from disk, printing out results, etc. It also collects global statistics from nodes. The node portion contains all the actual event processing, which typically consumes the vast majority of CPU time. By necessity, all big data analysis programs already cleanly separate these host and node tasks.

The user will encounter two important software environments. The first is the development host, a large multi-user VAX system. It has a few nodes attached to it for compiling and testing, and it supports the program development and debugging activities described below. All program and data files reside on disks on the development host. The production host, on the other hand, is a single-user system with a large number of nodes. It is devoted to production running of a single job for an extended period of time. It is networked to the development host for downloading of programs and data.

An extensive package of software has been developed by the ACP which makes the user's task easy. It is described in a full User's Manual [3]. This software is provided in three layers of increasing complexity. The first layer provides sufficient functionality for the majority of users with a minimum of parameters and options. The individual user need only use those parts, if any, of the second layer that are needed for a particular application. Layer 3, which gives complete control over the system, is reserved for experts.

All data passing between the host and the nodes is done by standard ACP subroutines called from the user's host Fortran program. There are three modes of passing data:

1. Event data. Routines SENDEVENT and GETEVENT pass one or more blocks of event data between the host and a single node;

2. Broadcast data. Routine BROADCAST copies a block of calibration data from the host to all the nodes.

3. Statistics data. Routine SUMNODE sums blocks of statistics or histogram data in all the nodes into single corresponding blocks in the host.

The routines referred to above are in layer 1 of the multilayer support. More generality is available in layer 2. For example, the layer 1 version of SENDEVENT sends a single block of data to a fixed common block in any available node. Using layer 2, users can, for example, send multiple blocks of data to the same node, direct the event to a particular node or class of nodes, and send variable length blocks of data.

Having prepared the Fortran source files for the host and for the nodes, the user invokes a VMS command procedure MULTICOMP which performs all compilations necessary to run on the multiprocessor system. This causes node files to be compiled and linked using the appropriate node compilers. The host file will be compiled and linked with ACP subroutines, including code that will download and startup the nodes at run time. The user controls this procedure by entries in a User Parameter File (UPF). This determines which source files are to be compiled and linked and which libraries of routines are to be used. The UPF also contains entries that control error handling and verification (see below), and specify the particular running environment the user desires. The running environment can be simulation within a single VAX, or a small number of nodes for testing attached to a development host, or a full-scale production system of hundreds of nodes under the production host. A simple RUN command starts execution.

The "operating system" in the node is extremely simple and written in Fortran, allowing it to be ported to different types of nodes very quickly. Its job is to support the Fortran run-time environment (but not to allow major I/O operation, which are done on the host), and to provide for data transmission from the host. The user's event processing code is a subroutine in the node. When an event is passed to the node, the system calls the user's code. When processing is complete the user returns control to the system, which automatically informs the host of event completion. The node system also supports error handling.

A multiprocessor presents a number of additional ways to deal with errors beyond those available on a uniprocessor. The ACP system gives the user five options when an error occurs:

1. Ignore the error

2. Continue with new events in the node where the error occured (losing the event with the error)

3. Reload and restart the offending node with a fresh copy of the program, calibration constants, and data arrays

4. Kill the offending node, continuing the job with the remaining nodes

5. Abort the entire job.

The user can specify in the UPF which of these actions are to be taken for each of a number of different errors and classes of errors, and can switch to a more severe action after the occurrence of a certain number of errors. For example, the error handler can ignore the first ten divide checks on a particular node, and then kill that node after any additional divide checks. The user also specifies what type of error log is to be saved for each error and how many nodes must remain alive before the job is automatically aborted. Users can provide special subroutines to be called on certain error conditions such as ABORT or timeout. Defaults are provided so the user need make no entries in the UPF to get standard error handling.

Part of the error handling system is a history file, which keeps track of the events sent to each node. This is used by ACP utilities to reproduce the precise sequence of events that produced a crash in some node so that the user can investigate with an interactive debugger those common situations where an unsuspected problem in an earlier event causes a crash at a later time. Traceback support will be provided on the host based on analysis of memory dumps which may be saved after an error.

An important debugging technique available only on a multiprocessor system is verification. An identical event is sent to two different nodes and result data blocks compared. This technique can catch both subtle hardware failures and software bugs where the answer is incorrectly dependent on the history of previous events processed in a particular node. Verification is automatically performed by the ACP software at a frequency specified by the user in the UPF.

PROJECT STATUS AND SCHEDULE

The ACP project is proceeding in a number of phases. A six-node testbed system with a complete software prototype was constructed and tested last summer. This winter, benchmark boards based on the three 32-bit CPUs and accompanying floating point processors (Motorola 68020/68881, AT&T 32100/32106, and DEC 78032/71032) were built and performance tests carried out. With the completion of these preliminary activities we are now in the process of constructing a full scale 128-node system which is expected to be ready before the end of this year. Future efforts will include incorporation of higher performance CPUs, higher bandwidth I/O, development of custom co-processors, and studies of different interconnect architectures.

The Testbed System consisted of 5 68000 CPUs with 512kB of memory each, and a single 8086/8087 CPU/FPU with 256kB of memory. The processors were in a 16-bit Multibus-I crate, interfaced to a VAX 11/780 through a DR-11W interface. The system was operated during the summer of 1984 with the full complement of software described above. The overall performance of this system was low because of the limited power of the 16-bit CPUs. The system was intended not for performance but primarily to demonstrate the ease of converting large physics application programs to the multiprocessor environment. Several actual track reconstruction programs were run on the multiprocessor after only a few days of program conversion effort. It also demonstrated that it is possible to use more than one type of processor in a transparent way and to make efficient use of multiple processors in the ACP system.

In the next stage of the project test boards for each of the three high performance 32-bit CPU/FPU chip combinations, Motorola, AT&T, and DEC, were benchmarked. The boards used fast static RAM so the processors could be run with 0 wait cycles. A fixed number of wait cycles (dip-switch selectable) could be inserted for all memory references to study performance degradation for slower memory systems. Cache options could also be studied.

The 68020 and 32100 suffered less than 10% degradation when run at 1 wait cycle (total of four CPU cycles for a memory access) with no external cache. Since this 1 wait cycle performance can be easily achieved using low-cost high-density 120 nanosecond dynamic RAM (for processors running at 16.67 MHz), the decision was made to build the full system using 1 wait cycle memory and no cache. Thus all benchmarks for these two CPUs were run with 1 wait cycle. The DEC 78032 CPU, which requires slower memory access times, can be run at 0 wait cycles with existing DRAM and was benchmarked accordingly.

Benchmarks were run using a physics event simulator and track reconstruction code, written in Fortran 77. Our standard for comparison is the VAX 11/780 with floating point accelerator, running Version 3 Fortran. Results were as follows:

CPU	Clock MHz	Compiler	Performance/VAX measured	expected
Mot. 68020	16.67	Absoft	.57	>.8
AT&T 32100	16.67	AT&T	.74	>.9
DEC 78032	40	VMS V4	.89	.89

Both the 68020 and 32100 compilers were unoptimized at the time the benchmarks were performed, while the 78032 compiler is quite highly optimized. New

optimized versions of the first two compilers will become available later this year, with expected performance improvements of 30-70%. Our best predictions for performance at the end of 1985 are given in the last column of the table.

Presently we are designing and constructing the components for a full scale system of 128 nodes, using MicroVAXes as hosts. The node CPU boards will be built on standard VME cards with 2 MBytes of onboard memory. Expansion memory up to a total of 8 MBytes is accomodated on an additional VME card with a local bus connection to the CPU card. This expansion memory can be placed either in the same VME crate next to the CPU cards, or in a separate VME crate located just below the CPU crate. A daughter board connector on the CPU card allows access to all address, data and control lines. This will be used for future developments such as grid interconnections and hardware subroutines.

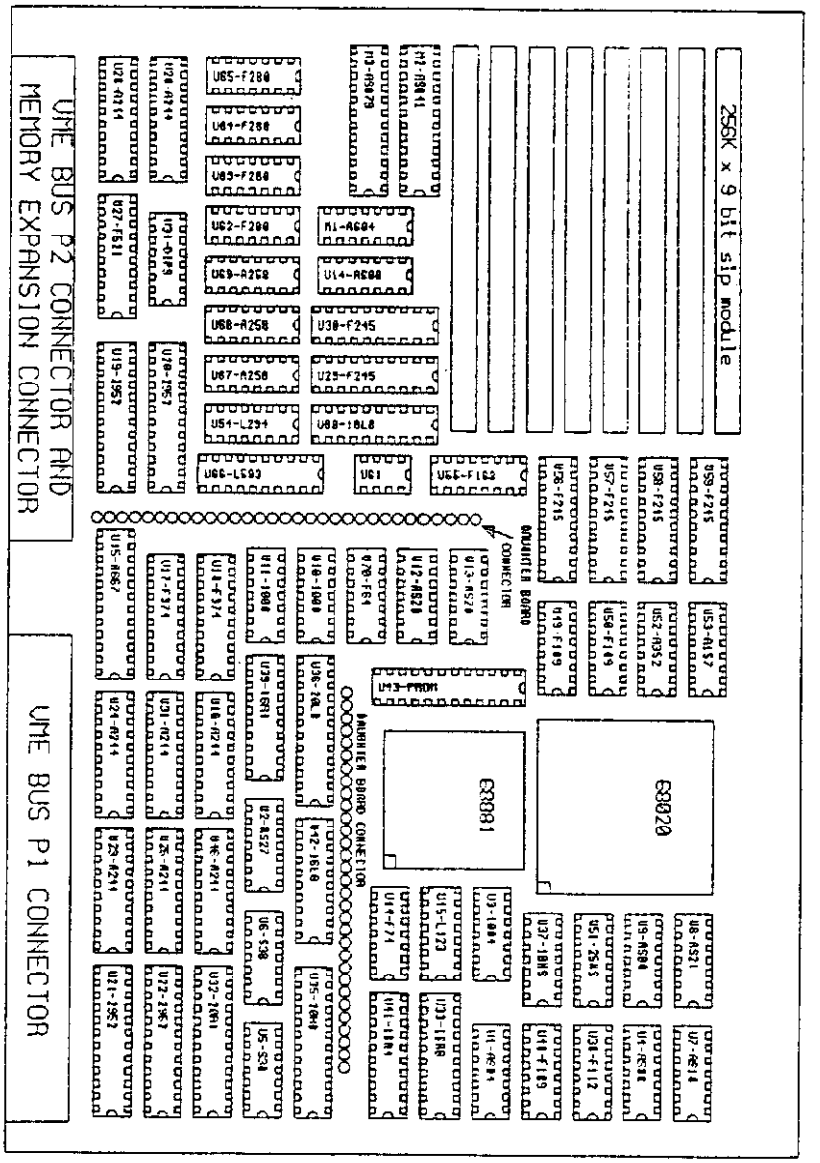

Fig. 4 Preliminary layout of ACP 68020 VME CPU board with 2 Mbytes memory.

The initial system will contain 64 each 68020 and 32100 based processing nodes. Additional processors using the DEC 78032 and/or the Intel 80386 will be added when those chips become commercially available. Fig.4 shows a tentative layout for the 68020 CPU multilayer PC card, which is presently being laid out. The 32100 CPU board will be quite similar. Memory cards have also been designed. The branch bus and crate controller are in final specification stage. The system is expected to be complete, with the full complement of software described above, before the end of this year. It can be readily copied at an assembled cost of about $2000 per node, including crates, production host, and other overhead. Omnibyte Corporation, 245 West Roosevelt Road, West Chicago, Illinois 60185, will sell the 68020 and 32100 boards at prices, including a 2 year warranty, very close to self fabrication costs.

ON-LINE APPLICATIONS

The same features which make the ACP system attractive for use in off-line systems (the low cost per node, ease of programming, and high system performance), also make it suitable for use in on-line applications. Another important factor is the ability to use exactly the same software for both off-line and on-line. The up to 160 MBytes/sec bandwidth of the Branch Bus and switch system exceeds the data collection bandwidth of virtually all experiments. The large amount of memory available in each CPU node provides a natural place to buffer events. The system is thus well-suited for use as a higher level trigger in an on-line environment.

Fig.5 shows an arrangement of ACP nodes in an on-line experiment. Events are delivered to the processing nodes from the experiment's data acquisition bus interfaced to the ACP Branch Bus. Branch Bus interfaces are planned for FASTBUS, Q-Bus and Unibus. Events are read out and logged on tape through a MicroVAX root as in the off-line case. User software controlling the system runs in the Resource Manager MicroVAX, which should be networked, as in the off-line case, to a larger development host VAX where programs are developed, tested, and stored.

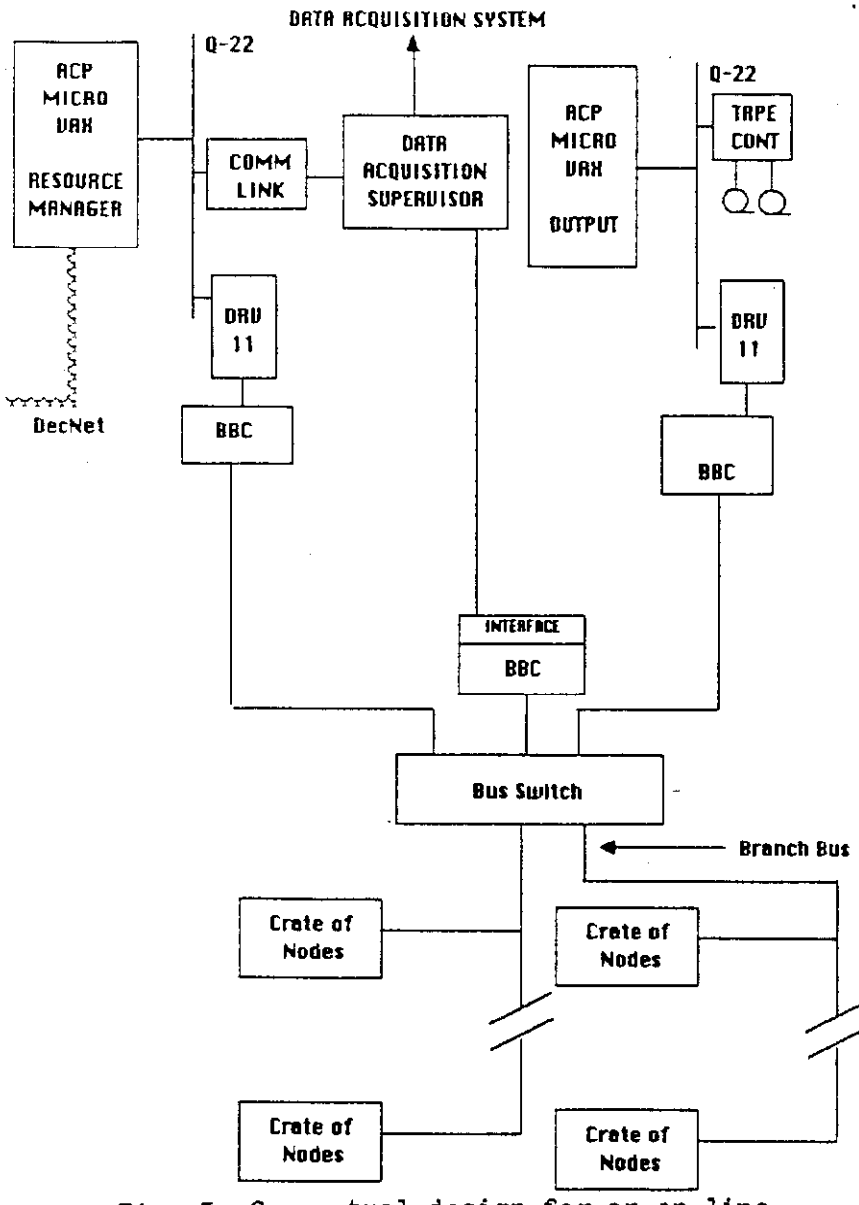

Fig. 5 Conceptual design for an on-line system using ACP components.

There are several possibilities for the control of the processing nodes in an on-line system, depending on system requirements. Simplest is to leave complete control in the hands of the ACP Resource Manager. In this case, the Resource Manager will communicate the address of the next free node to the data acquisition supervisor when the event occurs. The data acquisition simply delivers the event to the correct address. If no nodes are available, the data acquisition system must declare dead time and wait for a node to become free. Alternatively, more intelligence can be vested in the controllers of the data acquisition system. In this case, the ACP resource manager continues to check for errors and monitors event completion. However, the actual allocation of nodes is done by the experimental data acquisition system. This provides for less latency at each event occurance, and allows the data acquisition system to use its judgement to free up nodes when the system gets too busy.

The modularity of the software in the Resource Manager makes it easy to set up a system with control distributed appropriately for a particular experiment. In all cases, some communication path is necessary from the ACP Resource Manager to the supervisor of the data acquisition system. Depending on the application, this communication can be over DECNET, FASTBUS, or other standard buses. Such a system will be able to deliver 100 events per second to the nodes. The system could accommodate as many as 512 nodes with a total CPU power of over 400 VAX 780 equivalents. Output rate is determined by the speed of mass storage devices available, with a single Branch Bus able to run at 20 MBytes/sec for output. The ACP system is expected to be used as the Level 3 trigger for CDF at Fermilab. Details of the interface to the CDF data acquisition will be designed this summer.

In general, online trigger processors are a research area where the precisely correct approach is not yet clear. Particularly for the SSC, new and imaginative methods need to be tried and evaluated. A major strength of the ACP system is its flexibility, which allows tests of a number of different configurations before the best approach to the problem of dealing with luminosities of 10**33 is known.

ACKNOWLEDGEMENTS

S.Bracker made important contributions to the conceptual design of this project. We also thank D.Portier for assembling the manuscript.

REFERENCES

1. T. Nash et al., "Fermilab's Advanced Computer Research and Development Program", Proceedings, Three Day In-Depth Review on the Impact of Specialized Processors in Elementary Particle Physics, Padova, Italy, p. 227, 1983
   I. Gaines et al., "The Fermilab ACP Multiprocessor Project", Proceedings, Symposium of Recent Developments in Computing, Processor and Software Research for High-Energy Physics, Guanajuato, Mexico, p. 183, 1984
   D. Husby et al., "The Fermilab ACP Multi-Microprocessor Project", Proceedings, IEEE Nuclear Science Symposium, Orlando, Florida, p. 195, 1984
2. G.C.Fox and S. Otto, "Algorithms for Concurrent Processors", Physics Today, p. 50, May, 1984
3. M.Fischler, "Software for Event Oriented Processing on Multiprocessor Systems", Proceedings, Guanajuato symposium , p. 175
   Advanced Computer Program, "ACP Software User's Guide for Event Oriented Processing", Rev. Aug. 28, 1984, Fermilab FN-403