Fermi National Accelerator Laboratory

FERMILAB-Pub-87/38
2380.000

Use of New Computer Technologies in Elementary Particle Physics*

I. Gaines and T. Nash
Advanced Computer Program
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

February 20, 1987

*Submitted to the Annual Review of Nuclear and Particle Science

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

CONTENTS

USE OF NEW COMPUTER TECHNOLOGIES IN ELEMENTARY PARTICLE PHYSICS

Irwin Gaines and Thomas Nash
Advanced Computer Program
Fermi National Accelerator Laboratory
Batavia, Illinois 60510

1. INTRODUCTION

Elementary particle physics and computers have progressed together for as long as anyone can remember. The symbiosis is surprising considering the dissimilar objectives of these fields, but physics understanding cannot be had by simply detecting the passage of particles. It requires a selection of interesting events and their analysis in comparison with quantitative theoretical predictions. The extraordinary reach made by experimentalists into realms always further removed from everyday observation frequently encountered technology constraints. Pushing away such barriers has been an essential physicist activity from long before the development of the first practical electronic AND gates as coincidence circuits in 1930 by Rossi (Figure 1) *(1)*. This article describes the latest episode of this history: the development of new computer technologies to meet the huge and increasing appetite for computing by experimental (and theoretical) high energy physics.

The experimenters' computing needs have not been a sudden development. The long term growth in requirements is due to the increase in energy at which experiments are carried out. As the center of mass energy, $s^{1/2}$, increases, so does the complexity and, typically, the amount of data which must be analyzed. (Secondary particle multiplicities, total cross sections and accelerator luminosities all tend to increase with energy.) As the haystack has grown, so has the needle shrunk. With the standard model becoming more established, interesting physics to research enters at lower cross sections.

The problem has deep causes and exposing them gives a hint of the future (see Section 4). Fundamentally, the physicist task of selection and analysis remains the same whether using protractor and ruler for measurement in a cloud chamber or a $100 million detector at a colliding beam interaction point. This observational, pattern recognizing task was conceived by the human brain as an extension of the theoretical pattern recognition that is the creative core of science. Except for the quantity of data now involved, this is an activity better matched to the capability of a brain than that of a computer. We have applied the brute force numerical abilities of computers to very non-numerical problems. One acceptable definition of the fundamental goals of the research field known as Artificial Intelligence (AI) is the development of machines that will attack such non-numerical problems in a more direct manner. Real success toward this key AI ambition has certainly been hard to come by, but its importance for high energy physics (and science in general) cannot be overestimated.

Experiment computing has traditionally been divided into the on line task of trigger selection (see Sections 1.1 and 2) and off line reconstruction and analysis (see Sections 1.2 and 3). However, the distinction between on and off line is becoming softer as parallel processors programmed in high level languages are applied more frequently in real time. Given adequate data rate capacity, a computer does not recognize whether it is being fed from a real time buffer or a magnetic tape at a later time. The major concern on line is in the validity of criteria and programming which throw away data, forever. Intensive real time high level filters are, today, used with reluctance and generally only in later runs of experiments. The need for such sophisticated on line selection will become more and more compelling in the future. Artificial Intelligence concepts, such as program verification, again have fundamental relevance.

Theorists joined experimenters with large computing needs when lattice gauge theory was advanced by Wilson*(2)* as a means for numerically solving Quantum Chromodynamics (QCD). Widely accepted as the theory of the Strong Interactions, QCD cannot be solved perturbatively because of its large coupling constant. With existing algorithms and computers, precise lattice predictions of baryon masses, for example, seem tantalizingly out of reach; estimates like "70,000 Cray years" have been heard. Given their importance, the hope is that improved technology and algorithms will put these calculations on a sensible time scale. In Section 1.2 we introduce the lattice gauge problem from the computing perspective, and in Section 3.4 we describe processor efforts in this area.

Concluding the Introduction in Section 1.3, we discuss the place of elementary particle physics parallel computers in the modern world of computers. What seems like a surprising commonalty in the approaches being taken to computing for experiment and theory is found to be no more than a manifestation of the underlying structural regularity of science.

1.1 The Triggering Problem

As early as 1933, some cosmic ray experiments triggered cloud chambers on coincidences of pulses from Geiger tubes*(3)*. Yet, for many years, triggering was not a common experimenter's concern. Some work was carried out by counting pulses from Geiger and scintillation counters; much of the rest depended on detectors with indiscriminate sensitivity. Emulsions recorded tracks from the moment they were poured until they were developed. There being no correlation of the sensitive time of these detectors with any specific interaction known to be occurring, the events they recorded were pot luck.

The first spark chambers revolutionized physics in 1959*(4)*. The chamber was fired and pictures taken only when an electronic signal indicated that an appropriate interaction was likely to have happened. The "trigger" signals were based on a simple coincidence and or-ing of the pulses coming from a number of photomultiplier tubes attached to scintillating material. The time available for

these decisions was under 300 nsec so that the 10,000 or so volt spark chamber pulse could be formed in time. No longer was the leisurely tenth of a second time scale of cloud chamber triggers acceptable. The simple, but very high speed, electronics required to make these decisions was an outgrowth of the circuits used in a generation of "counter" experiments. By the time spark chambers took on a strong role, the standardization of "fast logic" electronics which could make a coincidence decision in 1-2 nsec was well underway.

Direct electronic spark chamber read out required the first use of on line computers to write the data onto magnetic tape, still the primary data storage medium of particle physics. Spark chambers were typically fired less than ten times per second and the track multiplicities that the direct read out systems could support was limited. The data analysis from these experiments did not overwhelm computer centers already used to the load from bubble chambers. In the 70s this situation changed as energies, multiplicities, and beam intensities increased and new detectors appeared that could support much higher rates. The irony was that, unlike spark chambers, these new detectors, multiwire proportional chambers (MWPCs), were untriggered devices. Yet, triggering was necessary because the on line computers could not keep up.

Triggers have another essential reason in experiments using drift chambers, a close relative of MWPCs in which the time for ions to drift toward a wire is measured to determine precisely the location of a passing particle. The time to digital converters (TDCs) that digitize this data require time reference signals. Similarly, analog to digital converters (ADCs) measure phototube pulse height, proportional to the shower energy in a calorimeter. The ADCs require a gate to define the time over which the phototube signal is to be integrated. The digitization process takes up valuable experiment live time (from a few to usually a few hundred μsec) during which the experiment cannot accept new events. Therefore, the signal to gate the digitizers must be made selectively, only when the probability of an interesting event being present is high.

Triggering electronics appear in large experiments in a variety of data rate environments and time domains as they carry out different functions. The triggers are intertwined with digitization, monitoring and data acquisition electronics in an extraordinarily complicated system. The Collider Detector at Fermilab (CDF), as an example of this scale of effort, is spending over \$1 million on triggers of all kinds, about \$12 million on data acquisition electronics including digitizers, and over \$2 million on large on line data logging and monitoring computers and consoles*(5)*.

Triggers are generally organized in a multi level structure. Passing from each level is only as much data as following electronics, with tasks of increasing complexity, can handle. In this article, we will classify triggers as "low level", "middle level", and "high level". In any specific experiment the distinction between these levels may be fuzzy. The levels may be compressed or expanded or bypassed, subject to the requirements and creativity of the experiment. "Low

level triggers" determine when a basic interaction of interest has been detected at a rate that is appropriate to gate digitizers. An important limitation is that the selection can only be based on parallel information from individual subdetectors, never on global and tracking information which is only available at later stages. A "middle level trigger" is, for our purposes, one which neither is used to gate digitizers nor is programmed in a high level language like Fortran. A trigger programmed in a high level language is here defined as a "high level trigger". The purpose of both middle and high level triggers is to reduce the amount of data ultimately stored for off line analysis.

In sum, the broad reasons why modern experiments require triggers are: to establish digitizer time references; to reduce the experiment dead time due to on line data recording; and to limit the amount of data which must be analyzed off line. Future experiments show no signs of becoming any less dependant on sophisticated real time selection. In fact, for reasons like those affecting data analysis, the requirements for trigger reductions are becoming more severe with time. Data rates are increasing faster than the capacity of digitizing and data recording systems available at acceptable cost.

1.2 The Off Line Computing Problem

An experiment's first off line computing task is reconstructing raw detector signals into useful physics information about each interaction event. The raw data consists of digitized pulse sizes and times as well as arrays of bits indicating whether a wire or counter was hit. This must be transformed into the three momentum, type, and originating (vertex) location of each particle and the error matrices for these quantities. The reconstructed data is the starting point for the real physics: analysis, in terms of theoretical or phenomenological models, of accumulated event distributions over relevant kinematical variables.

The two phases, reconstruction and analysis, pose very different problems. The first demands huge amounts of computation with human activity limited to monitoring of progress and quality. The second requires heavy, hopefully efficient, human interaction and a far more moderate, though significant, computing load. Development of new computers for experiments has been primarily directed at reconstruction. The analysis phase has been handled by commercial computers. Here, too, the situation is in sight of becoming intolerable because of inefficient use of physicist time, and attention is being directed at specialized solutions (Section 4).

The most common attribute of reconstruction software is that almost all such programs have been written specifically for one detector with its special geometry, physics, background and rate considerations (and the whims of its physicists) in mind. Tracks are built up from position coordinates in wire chambers generally in a series of stages. Small curve segments, for example, may be projected into lists of other segments to find matches and produce larger

segments. The process proceeds until all usable detector hits are accounted for. Energy in single calorimeter detector channels is identified with clusters of neighbors showing some energy and with nearby wire chambers. The tracks must be associated with Cerenkov counter and calorimetry information to establish particle type and account for as much interaction energy as possible. Least squared fits throughout the program estimate the values of the required physical quantities.

This whole, extensive, hand crafted process is complicated by the need to consider a variety of subtle, detector specific effects such as noise and inefficiency and ambiguities which result from several particles hitting a small region. The programs, not surprisingly, are complex and long. They are generally unstructured, with frequent conditional branches and other features horrifying to computer scientists. Fixed target experiments with reconstruction programs larger than 25,000 lines are the rule, and for big colliding detectors 100,000 lines are typical. Large memory banks are required just to manipulate lists of temporary data during matching and fitting operations. Calibration constants can consume several Megabytes of memory. A 6 Mbyte memory requirement for a colliding beam detector reconstruction program is no longer considered unreasonable by computer centers.

Reconstruction programs require extensive computation primarily because their brute force pattern recognizing approach requires large and deeply nested loops to test all reasonable combinations of detector hits for possible association into tracks or clusters for large numbers of events. Table I gives representative examples of the average computer time taken per event by a number of experiments, the number of events they have or anticipate having per calendar year of operation, and the total computer time required per calendar year*(8)*. Clearly this amount of computing is not tenable with conventional computers within the limited budget of a basic science. This situation has prompted development of more cost effective solutions by the high energy physics community.

In lattice gauge theory calculations the four space time dimensions are mapped onto a grid of finite spacing*(9)*. Monte Carlo methods are then used to evaluate expectation values of physically relevant quantities using Feynman's path integral formulation. For QCD, products of SU(3) matrices must be evaluated to determine how the action changes at each lattice step. This must be done for each SU(3) gauge field variable -- corresponding to links in the lattice -- to get a new configuration. With conventional algorithms, at least 10,000 such sweeps are required (and, perhaps orders of magnitude more) to insure that the final configuration is statistically uncorrelated with the starting point. The total for a state of the art 16^4 site lattice is about 250 million floating point operations per sweep or at least 2.5×10^{12} per calculation. Depending on hardware utilization factors, this corresponds to 4-8 hours per calculation on a $>$ \$10 million, $>$ 400 Mflops (million floating point operations per second), two processor Cray XMP.

Even this amount of computing provides crude calculations, accurate at best to 10%, and must be repeated frequently as different observables are studied.

The immediate, if limited, goal of this activity is a phenomenological understanding of the theory. Even for this, an order of magnitude more precision is required. In the long run, lattice gauge tantalizes with the opportunity to test with precision a fundamental theory of physics which cannot be tested otherwise. Much larger lattices will be required and quark loop effects will have to be included, both increasing computing requirements enormously. For example, a 1000^4 lattice, which acceptably contains an entire proton, requires a factor of 16 million more time than current 16^4 lattices -- and significantly more yet when including quark effects. The truth is that the extrapolation of these requirements from present understanding is so large that accurate estimates are impossible. It is clear, however, that anticipated hardware improvements alone will not be enough. Much will have to come from better algorithms, and research into these is getting as much attention (and computer time) as phenomenological studies. Already new algorithms (such as the Langevin method*(10)*) are reducing the number of sweeps required to decorrelate. The large computing demand has here also led to the development of specialized processors, many of which have rather restrictive algorithm specific architectures.

1.3 The Parallel Processors of Particle Physics

The very regular structures of the two critical high energy physics problems just described invite parallel computer solutions. Experimenters are using simple architectures with numbers of highly cost effective, stripped down computers to process raw data. Single events are sent to individual processors which pass the results to a host computer to record on tape. Until recently, the individual processors were all of a form called "emulators" because their hardware was arranged to "emulate" the instruction set of a large main frame family of computers (generally IBM). This allowed the users of these machines to take advantage of the software tools developed for the commercial computers, in particular the compilers. A newer approach uses larger numbers of single board computers based on commercial 32 bit microprocessors which are supported by their own Fortran compilers. Theorists have been building parallel computers, also often based on microprocessors in grid architectures, that naturally match their problem.

In order to understand where these particle physics computers fit in the modern world of computers, let's look at a computer taxonomy from a physicist's perspective. It is commonly implied that a computer must be either special or general purpose. A few, such as the low level triggers of high energy physics and military signal processors, are uniquely able to carry out one task. However, any programmable computer can execute different programs with different tasks. In this sense they are all general purpose. Even large commercial

computers are designed to be efficient at specific tasks required by different portions of the marketplace, transactions or vector computations, for example. In this sense they are all special purpose. The physics machines, though driven by a special interest in a particular class of problems, are efficient at a large number of tasks. They fall somewhere in the middle of what is really a spectrum of generality.

Traditional computers are referred to as Single Instruction, Single Data stream (SISD) machines. Big vector computers like those made by Cray are SIMD machines since a Single Instruction operates on elements from Multiple Data streams. Truly parallel machines like experimentalists' computers have Multiple Instruction streams operating independently on Multiple Data streams (MIMD). Some specialized lattice gauge processors, however, operate in lock step with essentially a single instruction stream on all lattice points.

What probably distinguishes physicists' computers most is that they are explicitly parallel with the individual processors primarily having their own local memory rather than sharing a global memory. The direction taken by most parallel computer research outside physics has been toward machines with many processors accessing a common, "global", memory through a complex switch which handles the necessary synchronization. Much computer science effort is directed at supporting implicit, automatic, decomposition of algorithms onto parallel processors which share memory. The idea of identifying the structure of a problem and explicitly mapping it onto a parallel computer architecture has been much easier for physicists to accept than for computer scientists. In an extensive study*(11)*, Fox has demonstrated that most scientific problems can be explicitly mapped onto certain local memory gridlike architectures (hypercubes) so they make efficient (usually greater than 90%) utilization of the hardware. The interconnection of local memory processors is by nature simpler than global switches and, therefore, amenable to larger numbers of low cost processors such as single board computers. The willingness of physicists to accept explicit parallelism has been rewarded with access to what are the most cost effective means of high level language computing presently available, 32 bit microprocessor and floating point arithmetic integrated circuits. In a phrase, the computers of particle physics can be classified as primarily *somewhat specialized, local memory, explicitly parallel computers.*

2. ON LINE SPECIAL PROCESSORS

When designing triggers for most experiments, there is a conflict between two requirements, high speed and flexibility. This conflict is resolved with a hierarchy of increasingly complex triggers. Each successive level makes more detailed decisions requiring greater amounts of time on fewer events. Figure 2 shows the triggering hierarchy for the CERN UA1 experiment*(12)*. Early levels of the triggers, while often blindingly fast and quite powerful, are truly specialized

processors. They lack the programmability necessary to make complex physics decisions or to adjust to a variety of conditions. Often these low level triggers are only understood by a very small group of experts, making the event selection criteria inaccessible to most of the physicists in a collaboration. Nevertheless, the need for making increasingly complex decisions on more and more events has forced the use of such specialized and inflexible processors.

A major challenge will be to develop flexible systems that can be used at earlier trigger stages without sacrificing speed. In particular, middle level triggers function in a data rate environment too high for existing high level triggers. Many smaller experiments do not yet incorporate high level triggers and stop at the middle (or even the low) level. However, the clear tendency is for more experiments to use high level triggers at the earliest possible place in the chain. The reason for this is not only the ease of preparing a trigger in Fortran. More importantly, it is the confidence in the validity and appropriateness of the algorithm that results from the far more extensive testing possible with high level systems. This allows more complex algorithms and deeper trigger reductions which are a requirement becoming more severe with time. Trigger reductions of 10^8 to 10^9 are anticipated for experiments on the proposed Superconducting Super Collider (SSC)*(13)*. It is likely that such requirements virtually eliminate "middle level" triggers as we have defined them. Processors based on verifiable high level language code will have to take on these tasks.

Our coverage of lower level on line processors will of necessity be incomplete. Entire conferences are devoted to this topic, and we refer to their proceedings for further information*(14-28)*. We will almost totally neglect the important use of processors for experimental monitoring, calibration, and data formatting and compaction. Such functions are an important part of all data acquisition systems usually performed by special processors, but space limitations require that we concentrate on the event selecting processors.

2.1 Large Experiment Trigger Hierarchies

The trigger hierarchy is best understood by looking at two large colliding beam experiments. ALEPH*(29)*, a detector now under construction will take data at the Large Electron Positron Collider (LEP) at CERN beginning in 1989. The detector consists of an Inner Track Chamber (ITC), a Time Projection Chamber (TPC), Electromagnetic and Hadronic Calorimeters, and Muon Chambers, with a total of about 500,000 digitizations per event. The beam crossing rate at LEP is 40.5 KHz, allowing 25 μsec for trigger decisions before any dead-time is incurred. The maximum event recording rate is 2 Hz. The triggering system is designed to identify all e^+e^- interactions while reducing background to an acceptable level, all with a minimum of dead time. Other trigger system constraints come from the detector. The TPC cannot be gated faster than 500 Hz, and the electromagnetic calorimeter requires 17 μsec to dump its charge and be ready for

the next crossing. A three-stage trigger scheme has been adopted. The low level trigger using ITC and calorimetry data will reduce the rate to below 500 Hz in about 3 μsec, and will gate the TPC. The middle level trigger will use TPC trigger information to reduce the rate below 10 Hz in 50 μsec. The final high level trigger will be applied only after the entire event has been read out. It will use actual high level language reconstruction programs to bring the rate down below 2 Hz. (ALEPH also uses many special processors besides the trigger devices. These include read out controllers for data reduction, calibration and monitoring, a trigger supervisor, and an event builder.)

UA1*(12)* is a large general purpose detector that has been successfully taking data for several years at the CERN $\bar{p}p$ collider. The bunch spacing is 3.8 μsec, and the interaction rate is expected to reach 150 KHz. The trigger reduces the rate to 5 Hz recorded on tape. An important constraint is the massive amount of data from the Central Tracking Chambers (CTC), which require 25 msec for data reduction and read out. Therefore, the first two levels of triggering cannot use CTC data and must reduce the rate to well below 40 Hz. Here again, a three stage trigger is used (Figure 2).

The low level trigger uses data on hit muon chamber drift cells and analog sums of calorimeter channels. It reduces the rate to 100 Hz at which the full digitized event is stored in a double buffer. The second level trigger uses muon timing and digitized calorimeter data (with best available calibrations) to make more detailed physics selections. It requires 3 msec to reduce the rate to 20 Hz at which the full CTC is read out. A special processor has been designed for the second level calorimeter trigger. The data consists of 20,000 16 bit pulse height words and 20,000 addresses, and the required processing includes pedestal subtraction, calibration in terms of transverse energy, comparison with a threshold, summation of transverse energy over regions of the calorimeter, and a weighted sum of energy over the full calorimeter. The task is split into two phases. Number crunching hard wired special purpose devices do calculations at high speed in parallel. Their results are fed to pattern recognition programs in standard microprocessors which identify detected electrons and jets and trigger on missing and total transverse energy. Finally, the third stage trigger uses six 3081E emulators programmed in Fortran to make the final physics selection.

2.2 Low and Midlevel Triggers

A good example of a modern low level trigger is the Level 1 trigger for the Collider Detector at Fermilab (CDF)*(30)*, a large multi-purpose $\bar{p}p$ detector due to take data in 1987. Typical of such triggers, it is required to make a decision in the less than 3.5 μsec between beam crossings to avoid dead time, and it must use data that is delivered separately from the normal data read out path. The fundamental component of this trigger comes from the electromagnetic and hadronic calorimeters. (Other parts are derived from muon chambers.) It exploits

the projective geometry of the calorimeter by summing calorimeter towers into 15^{o} azimuthal trigger sectors, 0.2 in rapidity wide. There are a total of 24 x 42 electromagnetic and hadronic trigger sectors. The trigger gets signals from dedicated outputs on the front end sample and hold cards via dedicated cables. The signals first go to a "receive-and-weight" card which weights the energy by sin θ, and then to "cluster sum" cards which calculate the number of towers with energies over several preset thresholds. The decision is then based on this number and the total transverse energy. Typical is the use of specialized dedicated hardware and the limited flexibility of this low level trigger. The calculations of calorimeter energy are refined in the CDF middle level trigger, which is more flexible and allows more sophisticated filtering.

Middle level triggers have more time available for complex trigger decisions, permitting systems with some programmability. These triggers still generally do not have the full event data available, since they tend to precede full event read out and process information from a separate path. Since events are usually not buffered while the middle level triggers are working, tight constraints on elapsed time can discourage use of the parallelism seen in high level triggers. They share some aspects of both low and high level processors and include hard wired, microprogrammable, and data driven systems. We describe examples of each type.

An elegant example of a hard wired middle level trigger is the Mark II track finder used at SPEAR and PEP*(91)*. This processor is designed to find curved tracks in a cylindrical geometry detector with an axial magnetic field. A low level pretrigger reduces the event rate to about 1 KHz. The middle level processor must then find tracks in about 30 μsec. Hits in the drift chamber feed a series of shift registers with variable delays, allowing a track with arbitrary curvature to be shifted into a straight line (Figure 3). Multiple curvature modules, each with different shift register delays, can look for tracks of different curvatures simultaneously. The shift register outputs then generate the address of a 2 bit RAM that determines whether or not that hit pattern corresponds to a good track. The data bits are artificially widened to generate a road. Required parameters, such as delays and widths, and RAMs are all programmable.

This device is an example of the commonly used technique of comparing the data from a particular event with a series of prestored patterns. The comparisons are made very rapidly since all calculations have been done in advance. In this case, the pattern of drift chamber hits is compared with prestored patterns for desirable tracks. Typical of middle level triggers, this one provides a limited degree of flexibility (what momentum tracks are accepted, how many missing hits, how wide the roads, etc.) without sacrificing speed.

Greater flexibility for midlevel triggers is possible with explicitly programmable devices. Special computers have been designed because conventional microprocessors are normally too slow. Bit slice technology, for instance, which requires many integrated circuits for a complete CPU, allows higher instruction execution speed. Bit slice processors are programmed in microcode, an extremely

low level language which uses individual bits of instruction words to set the state of specific gates or multiplexors in the processor. Using microcode, a special instruction set may be prepared, tailored to the task at hand. To make the processor more efficient, designers may, for example, restrict precision to 16 bit fixed point.

Examples of such devices are the CERN ESOP and XOP processors*(32)*. The older ESOP processors have been used in several CERN experiments including the European Hybrid Spectrometer, NA11, and R807, while the newer generation XOPs are used by UA2 and the LEP L3 experiment. The XOP processor is optimized for trigger computations of under 4000 instructions on 16 bit integers with execution times up to several msec. XOP uses a very wide (160 bit) microinstruction word, concurrent execution of arithmetic operations, data address calculations, data accessing, condition checking, loop count checking, and next instruction evaluation. It supports such specialized instructions as bit search, population count, and loose compare. All this allows it to do trigger calculations 20 times faster than a 68000 microprocessor (or about 2 VAXes). Another microcoded processor is the M7, built at Fermilab*(33)*.

All these processors share the disadvantage of lacking any high level language in which to program, thereby limiting the complexity of their algorithms. Similar microcoded special instruction set processors find extensive use in read out control and calibration applications. Here the lack of a high level language is less of a problem, since the calibration and read out programs are usually shorter and more stable than triggers programs. Examples are the MX used by CDF*(34)* and the BADC used at SLAC*(35)*.

Data driven processors represent an approach which attempts to combine the speed of hard wired with the programmability of special instruction set processor systems. Examples are the ECL-CAMAC system from Fermilab*(36)*, CERN's MBNIM system*(37)*, and the data driven system developed at Columbia University*(38)*, The latter ambitiously implements a full track reconstruction program with the processor.

These systems allow a trigger to be programmed, yet make maximum use of both parallelism and pipelining to insure that many processing steps take place at once. They also attempt to make each individual processing step as powerful as possible within a short time interval. They use a set of discrete modules that carry out processing steps simultaneously, each doing one operation every 25-50 nsec. Important are general purpose table look up modules which can be reprogrammed (even in Fortran) with the tabulated result of an arbitrarily complicated calculation. A single 50 nsec step can in this way correspond to a huge, precomputed, algorithm. Additional modules include stacks to store raw data and act as buffers between asynchronous loops, and even Fortran-like hardware DO loop indexers. These have been combined to implement powerful track finding algorithms at speeds several orders of magnitude faster than on large main frame computers*(39)*. The only drawback is the difficulty of

"programming" such systems. Rather than working in a high level language, the user programs by recabling modules and loading precalculated results into the tables.

High level triggers typically use the full digitized event after it has been read out and buffered. These systems must make extremely large amounts of computing available so that many events can be examined in detail in a short time. The cheap and powerful high level language programmable processors described in the next section are natural solutions to this requirement, even being able to use the same programs as for off line analysis. We will describe in Section 3.3 how these are installed as high level triggers.

3. HIGH LEVEL LANGUAGE EVENT PROCESSORS

By the mid 1970s, cost effective microprogrammable devices were clearly useful for short trigger algorithms but far too difficult to program for any long trigger or reconstruction program. When it was recognized, probably first by Kunz*(40)*, that the bit slice microprocessor technology used in such processors could be microprogrammed to execute the instructions of a commercial computer family, a new era of particle physics computing was started. Several such "emulating processors" were designed in this period*(41)* including Kunz's 168/E at SLAC which was targeted for an IBM off line environment, the NYU Courant Institute's PUMA which emulated a Control Data 6600, and the CERN MICE processor intended for quick trigger programs and based on the PDP 11 instruction set*(42)*. Some were extremely popular with arrays of emulators computing a hundred times cheaper than big commercial machines. The 168/E, in particular, was followed by improved versions which are still being built today.

In the last year or so the emulators have felt strong competition in cost effectiveness and ease of use from systems with large numbers of single board computers based on 32 bit microprocessors supported by Fortran 77 compilers. Improvements in integrated circuit technology -- and the compilers -- are likely to make the cost performance advantage so strong that such multi microprocessors will take over from the traditionally popular emulators.

3.1 Emulators

Emulators have evolved steadily since their introduction. They were at first limited to important integer operations. The big-machine assembly language this permitted was a large jump from microprogramming. The emulation concept soon allowed another leap to Fortran using the excellent compilers prepared by the computer manufacturers. The 168/E supported most of the instructions required by IBM Fortran except for floating point. Even on line programs for the MICE system were typically half in Fortran. The popularity of MICE and the 168/E

ultimately led to the development of floating point processors for them so they would more completely emulate the instructions required by physics.

Both processors took advantage of the fact that neither trigger nor reconstruction programs require data Input/Output (I/O) during the execution of the kernels of their programs. It is sufficient to supply event information at the beginning of processing an event and retrieve results at the end. As a result, great simplifications can be had by not making direct, formatted Fortran I/O available directly from the processor. All I/O, including program and calibration constant downloading, is handled by a controlling computer. This "host" is connected by an appropriate interface to the processors.

The 168/E was originally designed to do off line reconstruction of the huge data flow from the SLAC LASS spectrometer*(40)*. The key components were four of AMD's 4 bit microprocessor slices, the 2901. The 168/E's ultimate cycle time was 150 nsec and its performance was 50-75% of a 360/168 (or 2-3 VAXes). In 1980 the 168/E was reported to cost about $20,000 per processor fully loaded with 192 Kbytes of memory, including interfacing and assembly (but not testing) in a six processor system*(43)*.

While the 168/E was used for a variety of on line triggers in addition to its original off line applications, the MICE processor was always used only for on line triggering. It's emulation of the PDP11 instruction set was accomplished with Motorola's 10800 series ECL bit slice at a micro cycle time of 105 nsec. It performed at speeds three times the most powerful computer in the PDP family, the PDP11/70 (or about 1 VAX), when programmed in Fortran or assembly language, and two to five times faster yet in microcode which was often used in inner loops. A single processor installation including 4 Kbytes of fast memory and a CERN Romulus interface cost $13,000 in 1980. The MICE processor was ultimately used in at least a half dozen CERN experiments including the neutrino experiment WA1, the ISR experiment R704, and the Omega spectrometer.

The 168/E addressed the off line problem in an ambitious and ultimately valuable way. This led to its becoming the most popular emulator with about 50 processors built and installed at SLAC, CERN, DESY, Saclay, in Toronto and Tokyo and elsewhere. It has been used in "farms" of six or seven parallel machines. The new opportunity for complex high level triggers led to trigger applications at CERN and SLAC, and development of interfaces to CAMAC, Fastbus, and Unibus for activities as diverse as ISR experiments, the UA1 $\bar{p}p$ experiment, and the SLAC hybrid bubble chamber.

All emulators other than the 168/E and its successor the 3081/E have used fixed microcode for the execution of a defined, emulating, instruction set. They emulate at the machine instruction level. The 168/E was unique in employing software translation to microcode. The translator pass is transparent to users who see it only as an apparent delay in the IBM compiler-linker activity. The advantage of this approach is that it eliminates the need for complex hardware

to decode the emulating instruction set into microinstructions. For the new 3081/E there is the added benefit that the translator can set up sophisticated, performance enhancing, pipelines that would otherwise not be possible.

The 168/E's major weaknesses were identified by its designers and included: limited memory requiring extensive software and hardware to support overlays; lack of full 64 bit floating point and byte addressing or manipulation instructions; and lack of adequate testing mechanisms*(44)*. In addition, the 168/E had been designed as a one of a kind installation for LASS. As a typical product of a high energy physics experiment it used very tight timing and "clever", non-modular, design techniques*(45)*. These problems were recognized by Kunz and were professionally avoided in the beautifully modular 3081/E.

Also attempting to improve on the 168/E was the 370/E designed at the Weizmann Institute in Israel. This effort was led by Brafman who had been involved in the 168/E design while visiting SLAC. Here the emphasis was on eliminating the software translation to microcode and emulating the IBM computers at the machine instruction level*(46)*. The 370/E can run untranslated, unmodified IBM object code, including formated I/O, and the IBM operating system. This is felt to be a particularly desirable feature at DESY*(47)* and by others for whom the task of dividing a program into a host and processor part is perceived as too onerous. For highly parallel systems, where the I/O bandwidth can be saturated, the option of doing I/O directly from node processors gives naive users enough rope on which to hang themselves. Nevertheless, for low numbers of processors, as is the case with the 370/E, it is a decided convenience to be able to throw any old program directly into the emulator and have it run.

As may be seen in Figure 4a, the 370/E uses an architecture of multiple busses to which are connected several function units, integer, floating point, multiplication, control, and interfacing. Separating functions in this manner increases the chip count somewhat but greatly reduces design, testing, and debugging time because the potentially complex control logic is much simplified. Although a bit slice 2901B is still used in the integer CPU, more functionality is now obtained with FAST and LS series TTL circuits than was the case with the earlier emulators. A multilevel pipeline prefetches instructions and allows microcoding machine language instructions preparatory to execution. The 150 nsec microcycle is narrowed to 100 nsec during multicycle shift, multiply and divide operations. This emulator attains 60% of the speed of an IBM 370/168 (or 2.5 VAXes) and supports the full IBM address space. A system with two MBytes of memory was reported as costing 45,000 DM (about $18,000) in 1985. A 20% faster prototype was completed at that time in Israel. At least 20 370/Es have been interfaced to a variety of DEC and IBM host computers at over ten locations in Israel, England, Germany, the US, and at CERN for the LEP experiments OPAL and DELPHI. Typical installations involve one or two units, but at Cornell there is a farm of six.

CERN people had seen the 168/E prototype running test programs at SLAC and started work at CERN in spring 1978 on a copy. By the end of 1980 two systems were operational and being tested, off line by the European Muon Collaboration and as an on line filter by the ISR Split Field Magnet group. In 1981 this careful CERN evaluation led to a collaboration with the original designers at SLAC to make a new version, subsequently named the 3081/E after the first of a new series of high performance IBM mainframes. The design work was divided equally between the two institutions.

Like the 370/E, this processor*(45)* is modular with separated functional units on several busses (Figure 4b) and supports a more complete set of IBM instructions, including full double precision, and much larger memory, up to 14 Mbyte with 64K static RAMs, than its predecessor. Unlike their 370/E competition, the 3081/E designers retained software translation into microcode which simplifies the design and can also automatically produce pipelined floating point operations. The 3081/E runs at 1 to 1.5 times the speed of a 370/168 (or 4-6 VAXes), about twice that of the 168/E. Its cost with 4 MBytes is now about $20,000, half attributable to the expensive fast static memory. The designers put a very strong emphasis on ease of building, debugging and maintaining what was expected to be a frequently reproduced processor. Accordingly, the design was much simpler than the 168/E and conservative design rules were followed such as those requiring worst case timing and multiple commercial sources for components.

A Common Interface board was designed to provide a means of communication with all the 3081/E busses and, for debugging purposes, control of its clock and state. Execution may be halted by the interface on various condition traps. This interface has been connected in various installations to an impressive list of different busses including IBM channels, CAMAC, VME, and FASTBUS, using NORD, IBM (mainframes and PCs), Apollo, Motorola 68000, and VAX computers as hosts. Major 3081/E facilities started production at CERN in fall 1986 and a year earlier at SLAC. There are also 3081/Es at Saclay, Harvard, and in Italy.

3.2 Multi Microprocessors

The newly introduced 16 bit Motorola 68000 microprocessor was quickly adopted for particle physics applications in 1980. Though the speed of these devices continued to be a limiting factor, their extremely low cost encouraged their use in parallel systems. The Fast Amsterdam Multi Processor (FAMP) developed by Hertzberger and colleagues at NIKHEF was the most important early multi microprocessor*(48)*. It is still in use for high level triggering, having been a part of the triggers at the CERN ACCMOR spectrometer and UA1 experiments and the DESY JADE $e^{+}e^{-}$ spectrometer. Typically, three to seven processors in two

levels, one processor acting as a supervisor, the rest as slaves, were operated in parallel on data from different detector regions of a single event. Each was a true Single Board Computer (SBC) with CPU and up to 16 Kbytes of memory (with additional 128K extension boards available). Initially, programs were developed in assembly language which is really only suitable for short algorithms. This was remedied with a UNIX^{tm} operating system for FAMP under which high level languages like Fortran and C are available.

The big UA1 colliding beam detector used 60 68000 SBCs for data read out in addition to the seven FAMP CPUs used for triggering. These were designed to be compatible with the 32 bit VME bus standard which was new in 1982 and immediately recognized as appropriate for the UA1 system by Cittolin and colleagues*(49)*. Meanwhile at Bonn a CAMAC Auxiliary Crate Controller based on the 68000 was developed to manage data acquisition*(50)*. As part of this effort von der Schmitt wrote a Fortran compiler (RTF/68K) designed specifically for real time applications.

For some time, such multi microprocessors had only limited applications, mainly in middle level triggers and read out controllers, since the lack of power of the 16 bit CPUs and the incompleteness of the then available Fortran compilers limited the complexity of the algorithms that could be run. However, with the advent of the new generation 32 bit microprocessors and the development of full-fledged Fortran 77 compilers for these micros, the multi microprocessors became competitive with emulators for use as off line and on line processor farms. In fact, the multiprocessors have already passed the emulators in cost effectiveness, and show potential for significant further improvements in cost effectiveness in the near future, as described below.

These developments became the basis for a major new multi microprocessor effort aimed at off line and highest level on line computing by a new group at Fermilab. Named the Advanced Computer Program (ACP), it was formed in 1982 to confront the key particle physics computing problems, which by that time had been generally recognized as critical. The ACP's initial focus was primarily on event oriented multiprocessing. Like most emulator activity individual events were to be handled completely by separate CPUs*(51)*.

There are many commercial producers of SBCs. Although competition has not yet driven these products to the status of a commodity, as has happened for memory chips, the ACP Multiprocessor was developed with an open, eclectic philosophy to take advantage of the strong competition in this area. The high speed, 32 bit ACP Branchbus connects up to 16 crates of SBCs per branch to each other and the host computer. As the fundamental skeleton, the Branchbus and its optional 8x8 crossbar switch are the only features specified to remain the same in future variants of the ACP system. It is intended that, at any point in time, a system should use the most cost effective node CPUs available from high energy physics lab designers or, preferably, from commercial sources. Present ACP systems use VME standard crates interfaced to a Branchbus through a

Branchbus to VME Interface (BVI), as shown in Figure 5. If motivated by SBC product availability in some other, "xBUS", standard, a "BxI" interface module can readily be designed. Because of this competitive, eclectic philosophy, the ACP system software is designed to be primarily resident in the host computer, with only a small node operating system which can be ported with relatively little effort to a new CPU.

The ACP has developed a new CPU based on the 68020, Motorola's 32 bit successor to the 68000, and another on AT&T's 32100, to demonstrate the technical and pricing requirements of nodes for such multiprocessors. Commercial SBC designs are still primarily aimed at controls applications, where they are used in small numbers, and are not yet acceptably cost effective. The ACP boards include two MBytes of memory and the Motorola 68881 or AT&T 32106 floating point coprocessor (FPU) appropriate to the CPU in use. Except for the CPU/FPU used, the boards are essentially identical. They are standard, double Eurocard, VME designs with all normally supported VME single word transfer protocols. Unlike any other SBC available, they also support VME block transfers directly into memory. This permits extremely high rates of data transfer for on line triggers. The system has been tested to transfer data error free at 20 MBytes/sec from a FASTBUS module through the Branchbus into CPU memory for over 48 hours.

These first ACP CPU modules presently cost under $1500 to produce and run Fortran reconstruction programs at about 0.7 VAXes. Including the low crate and interfacing overhead in large systems their cost effectiveness is therefore now about $2000/VAX. Memory extensions up to six MBytes (at under $200/MByte) can be located in slots in a single Eurocard crate immediately below the CPUs. The ACP plans at least one new generation of CPUs and is presently investigating at least eight microprocessor candidates, a much wider choice than was available when the first CPUs were designed. Several of these are Reduced Instruction Set Computers (RISC) and they will depend on good Fortran compilers for realization of their extraordinary performance potential. One, the Fairchild Clipper, has passed the standard ACP reconstruction benchmark at a speed of 3 VAXes in Fortran. Since the cost of the new CPU will ultimately be similar to the present ones, this benchmark demonstrates the possibility of attaining a cost effectiveness of better than $500/VAX in 1988.

Figure 5 shows in block form the first full scale ACP Multiprocessor installed in the Fermilab Computer Center. It is a 140 node system, half based on 68020 and half on 32100 CPUs. Each VME crate contains up to 18 CPUs, a BVI, and a VME Resource Module (VRM) which handles arbitration. The BVIs act as master on VME and slaves on the Branchbus. The Branchbus Controller (BBC) is the Branchbus master and a slave on some other system, here the Qbus of a MicroVAX host computer. A VBBC is under design to allow direct VME control, and multiple mastership, of the Branchbus. The system may be hosted by one, two or three MicroVAXes which share a common memory on a

special VME root crate to which they are interfaced through a Qbus to VME interface (QVI).

An application program destined to run on this system must, as for the 3081/E emulator, be divided into two parts. One, running in the host, handles all I/O. The other, running in each node, does the actual number crunching. The ACP provides system subroutines to communicate between the host and node programs. Routines exist to broadcast calibration constants at the beginning and to sum statistics at the end of a run. Individual events are sent, and results retrieved, asynchronously, by user called send and get subroutines. Automatic floating point and integer conversions, the latter in hardware, are available when required by different host and node CPU standards. The CERN ZEBRA data bank package is supported essentially transparently to the user. Converting programs to meet the multiprocessor requirements is relatively easy, aided by a full, multiprocess simulator which runs on a VAX. A visitor inexperienced with the system was able to bring up the Lund Monte Carlo program on real nodes in two days.

The initial system started running under Fermilab Computer Center operator control in July, 1986, and ran for six months, with no downtime, on a huge backlog of data from the Tagged Photon Spectrometer experiment E691. With 100 processors (the remainder were assigned to other uses), the system performed at more than double the capacity of Fermilab's CDC 175 and 875 computers originally costing some $20 million. Omnibyte Corporation of West Chicago, Illinois, is selling all components of the ACP system at prices including initial testing and two year warranty. By the end of 1986, this company had orders for over 135 processors from at least ten institutions (including SIN, Los Alamos, Brookhaven, Rutgers, Yale, and the Universities of Toronto and Montreal) and had shipped over 100. This did not include an order for an 80 CPU second full system for the Fermilab Computer Center that was in process.

If there has been a weakness in the multi microprocessor approach, it has traditionally been compilers where there still remains an opportunity for a factor of two optimization. Effort in this direction is not encouraged by the commercial market place which has been more interested in compatibility with PCs than performance. It is easier to produce an efficient compiler if the goals are limited as they have been in von der Schmitt's real time compiler. However, for large programs a full Fortran 77 implementation, at minimum, is required. Some, such as the Absoft 68020 compiler, now are reasonably bug free and have proven very satisfactory for writing new codes and for truly portable Fortran 77 programs. Converting from programs prepared under other compiler standards can be more time consuming. This is a problem not unique to microprocessors. It is commonly encountered in large experiment collaborations unless there is an agreement to outlaw exclusive VAX and IBM Fortran dialect features available on some home institution computers.

The ability to use full IBM Fortran is seen as a strongly desirable feature by many emulator users. Similarly many from the VAX environment have a strong preference for using MicroVAXes in a multi microprocessor system even at a significant cost premium because it allows them to take advantage of VAX software. This was stated as a significant motivation by the D0 collaboration in their selection of a multi MicroVAX system developed by Cutts and Zeller at Brown University*(52)* for the highest level trigger on their Fermilab $\bar{p}p$ experiment. (After making their MicroVAX chipset available for prototyping, DEC decided not to release it for external designs, such as open architecture multiprocessors.) A 49 MicroVAX "farm" is planned with each node equipped with a special 256K dual port memory capable of absorbing data from an eight data cable parallel read out at the instantaneous 100 MBytes/sec data rates required by the experiment. Another MicroVax acts as the supervisor. Data is to be read out from the nodes, after event selection, via an Ethernet link to a host DEC 8600 at rates under a few hundred KBytes/sec. Higher rates will require using the dual port memory in write mode.

A 16 CPU farm has been used off line at Brown generating GEANT Monte Carlo events badly needed by the experiment for final design decisions. The DEC ELN software toolkit is convenient for debugging and for handling data at the low rates required by a time consuming large experiment Monte Carlo. Each MicroVAX, including five MBytes of DEC memory and interfacing, costs about $16,000*(53)*. Thanks to the highly optimized VMS compiler, the MicroVAX chip, which is intrinsically not as fast as either the Motorola or AT&T circuits, runs slightly faster in Fortran, about 80-90% VAX. Therefore, the present cost effectiveness of this system is about $18,000/VAX. This is expected to improve by about 20% with the availability of new DEC board level products. Nevertheless, this is a considerable premium over other options, but advocates argue that the costs of "non-VAX operation" should not be forgotten*(54)*. DEC has provided much support, including generous pricing, to this project, and one would expect it will be able to take advantage of successor MicroVAX chips rumored to be in production by 1988.

In another effort involving MicroVAXes, Siskind of NYCB Real Time Systems has used a Department of Energy Small Business Innovation Research grant to develop a FASTBUS MicroVAX which incorporates the DEC board level product as a piggyback in a multiboard, superfast (60 Mbyte/sec) FASTBUS interface*(55)*. The SLAC SLD detector is planning on using 15 of these in a slower (12 Mbyte/sec) and cheaper TTL version for a high level trigger*(56)*.

Each of the approaches to high level language processors has a strong advocacy. Emulators still provide higher performance individual processors, which can be essential in certain real time applications. For those who have the luxury of living within the warm and fuzzy environment of one computer manufacturer, emulators, along with single vendor multiprocessors, allow bit for bit comparison

with popular main frames. On the other hand, open multi microprocessor systems are more flexible, have a lower buy in cost, and have hardware that is easier to make work by nonexperts. They are more readily commercialized in an open competitive market. These systems already deliver at least as much performance per dollar as emulators, with much more cost effective CPUs expected soon. The reader may sense in this summary the flavor of debate that typifies this field. What is important is that this intense interest and activity has led to devices which show promise of being able to handle the computing load of experiments for the foreseeable future.

3.3 On Line Applications: The Interface Problem

Emulators and multiple microprocessor systems have both found application as high level triggers. Each individual processor can run large complex codes written in Fortran. The highly cost effective arrays of such processors pioneered in off line applications can be applied to obtain the massive amounts of processing power needed for on line systems. Problems in applying them on line arise primarily from a lack of standardization by experiments.

CDF is using an array of ACP processors for their level 3 trigger*(57)*. The system is required to accept events at 100 Hz and provide at least 50 VAX equivalents of processing power. The processor farm is fully integrated into the Fastbus data acquisition system, through a Fastbus to Branch Bus Converter (FBBC). The FBBC will be used both for inputting events to the processors (from the Fastbus Event Builders) and outputting from the ACP farm (by a VAX with a Fastbus interface). The system is managed by the Buffer Manager, which controls the data acquisition system, via Fastbus messages to the MicroVAX acting as host for the processor farm. On the other hand, the MEGA experiment at Los Alamos, also using an FBBC to attach ACP processors to a Fastbus system, controls the flow of events to the processors by polling them directly from Fastbus to determine which are ready to accept events*(58)*.

UA1 has used up to 6 168Es*(59)* and recently 3081E emulators*(60)*. The 168Es were originally interfaced to the CAMAC based data acquisition system through the CAMFast interface, but to obtain higher performance a new PAX-Greyhound system was developed. The PAX module was a CAMAC sequencer that optimized CAMAC read out and interfaced to the Greyhound bus, a new bus used to interconnect the 168Es. A new memory board for the 168Es was also developed by the experiment. The 3081Es which have recently replaced the 168Es are interfaced to a new VME system. A VME event builder reads the events from dual port memories and broadcasts them (a non-standard VME feature) to event task units, one of which is the farm of 3081Es. The Mark II at SLAC, on the other hand, attaches 3081Es directly to Fastbus*(61)*. As noted earlier, other experiments such as the D0 multiple MicroVAX system and the experiments using 370Es also have invented their own interfacing schemes.

These examples demonstrate the regretable lack of standardization in the way these processors are interfaced. Experiments frequently find it necessary to develop new modules. It is hard to say whether this derives from the relative newness of these devices or from system designer's wishes to be different. One hopes that the future will bring greater uniformity in the way such on line multiprocessors -- and data acquisition systems in general -- are implemented.

3.4 Lattice Gauge Engines

When confronted with the huge and important computing demands of lattice gauge calculations, theorists, encouraged by their experimentalist colleagues, began building their own parallel computers*(62)*. All the main theory processors are programmed in high level languages, usually Fortran, and many are multi microprocessors. Most supplement the Fortran engine with special devices that use high speed VLSI floating point multiplier and adder chips to compute the intensive kernels of their programs.

Until recently, only communication between processing nodes working on adjacent parts of the lattice was recognized as being required. This naturally led to grid like architectural arrangements. At Cal Tech, Fox, a physicist, and Seitz, a computer scientist joined forces to develop a hypercube of microprocessors*(63)*. The hypercube is a good topology because it embeds the simple one to four dimensional grids typically used by scientific calculations and, in the worst case, communication path distances increase only as log n. The original system, called the Cosmic Cube, was based on 64 Intel 8086/87 16 bit processors and was completed in 1983. It did not emphasize performance but was used for a complete study of the applicability of local memory grid architectures to a wide variety of problems*(11)*. This influential study alerted industry to the broad utility of such systems. It spawned commercial hypercube products from Intel, Ametek, N-Cube, and Floating Point Systems as well a hypercube development effort at The University at Southampton using Inmos Transputer chips, which have excellent communication hooks for such purposes*(64)*. An improved design using 68020s and, ultimately, daughter boards with Weitek floating point chips is being developed by Fox in collaboration with the Jet Propulsion Lab. A 128 node system is planned, and was about half complete at the end of 1986*(65)*.

At the same time as the Cosmic Cube was developed, Christ and Terrano at Columbia constructed a 4x4 toroidal grid of nodes of Intel 80286/87 microprocessors coupled to a microprogrammed floating point vector processor based on the then state of the art TRW VLSI multiplier and adder chips*(66)*. In this processor (Figure 6a) nodes operate in lock step. After identical processing cycles, each transfers data synchronously to its nearest neighbor in a given direction. This is truly SIMD operation appropriate to the homogeneous problems then of interest to the designers. This processor was the first one using the maximally cost effective floating point chips that had just become available.

It is able to reach 16 Mflops per node at a cost under $2500, still a very competitive figure, and has been used for important lattice calculations*(67)*. A 64 node grid incorporating Weitek chips was 75% complete at the end of 1986, and a 256 node machine with 16 Gigaflop performance is planned.

An Italian group led by Cabibbo is developing another synchronous SIMD-like machine (called APE) of a very different architecture (Figure 6b) but also using Weitek chips to compute the lattice gauge kernels*(68)*. Here a 3081/E is used to issue the simultaneous, SIMD, instructions to a bank of floating point units (FPUs) each containing four multipliers and two adders so that the operation y = axb+c on complex numbers is optimized. Each FPU has a maximum performance of 64 Mflops, so the planned 16 unit system could surpass a Gigaflop. Large memory is also important to big lattice calculations, and the design cleverly matches a Gigabyte of dynamic memory through a switch to the high speed FPUs. The switch can connect FPUs only to "neighboring" memory banks, but future design changes may relax this constraint. By the end of 1986 four FPUs were in place and the remainder were anticipated to be complete in 1987. (A similar project is the Space Time Array Computer (STAC) under development at Boston University*(68a)*.)

At present the largest lattice gauge processor project, both in terms of Gigaflops and of dollars, is being undertaken at IBM Watson Labs by Weingarten et al*(69)*. Targeted at eleven Gigaflops, the GF11 (Figure 6c) is also an SIMD machine with one high speed central controller issuing instructions, here to 576 floating point processors again based on Weitek multiply and ALU chips. These 20 Mflop FPUs are flexibly interconnected through a three stage Benes (shuffle) network of 24 x 24 crossbars. Used typically as a 512 node grid for an 8^4 lattice, the extra 64 processors are to overcome the fundamental weakness of all grid processors. Unlike tree structures they are extremely fault intolerant, failing if a single node goes down -- unless there are spares and a reconfigurable switch as in the GF11.

The ACP in a collaboration with Fermilab theorists is developing a lattice gauge processor with the architecture shown in Figure 6d based on ACP system modules. This structure is influenced by the more efficient non local algorithms (the Langevin method) that have recently been proposed for lattice gauge calculations*(10)*. It is just as easy for a processor to communicate to a far away node as to a nearest neighbor in this system. Truly MIMD it also allows asynchronous algorithms which may someday be proposed. The nodes will include microprogrammed "array processor" boards giving the 5 Mflop/$1000 level of cost effectiveness typical of systems based on Weitek chips, but here in a less constraining, fault tolerant architecture. The microcode will be prepared by experts and will look like subroutines to the theorist's Fortran program.

The lattice gauge processor story will not end here. Hopefully, a combination of dramatic new algorithm and hardware developments will make possible real QCD calculations before long. Given a realistic possibility of looking

for fundamental QCD discrepancies or testing predictions of higher unified theories, funding for lattice gauge processors could reach a level that today would be surprising for a theorists' endeavor.

4. UNSOLVED PROBLEMS: THE FUTURE

Technology projections are routinely overturned by unexpected inventions, but one must agree that at least the requirements of high energy physics appear certain and continuing. To date new computer initiatives in physics have primarily applied industry supplied components in suitably cost effective ways. Hardware development will continue, and there is every expectation that it will meet the data rate and trigger reduction needs of experiments on 20 TeV colliding beam machines. What is new is that demand for software innovation, not likely to be adequately met elsewhere, will put particle physics into forefront research in numerical algorithms and artificial intelligence.

At Fermilab the ACP is working on attaching devices based on video technology, like the cheap, mass produced Write Once Read Many times (WORM) optical disks and 8 mm cartridge tapes, directly to nodes within the ACP multiprocessor. This would allow systems originally conceived for reconstruction processing to reduce the analysis turn around for a large experiment from weeks to a half hour. The impact this could have on particle physics is clearly enormous. However, such a dramatic hardware improvement will have to be accompanied by a similarly dramatic change in the way physicists interact with analysis programs. It would hardly be worth developing hardware that cuts the analysis iteration time to this level if, as now, it continues to take many days for an experimenter to prepare the next try.

Analysis programs go through much modification, frequently by many hands, as new ideas, techniques and variables, are tried. They are intrinsically big, messy, unstructured Fortran programs with scattered subroutine calls to histogram and plotting packages with long and obscure call lists. Making changes is, not surprisingly, a time consuming mechanical endeavor. Tests, to make sure everything is right before committing to a full pass through the data, cause further delay. Something has to be done about this situation, and we think that the way is being shown by theorists and businessmen.

The obscurity of Fortran programs for doing science has been recognized in the theoretical context by Wilson et al*(70)*. Their "Gibbs Project" at Cornell is attempting to address the problem by developing syntax for a language at a level higher than Fortran which is more transparent to scientists. Programs will be organized like a text book with chapters defining, in standard mathematical notation, the basic equations, boundary conditions, algorithms (e.g. Simpson's rule), starting values, etc. of a problem. The project is presently using "human compilers" to establish a syntax sufficiently complete for an automatic compiler.

High quality workstations for their research and, later, the science are considered a prerequisite by the Gibbs group.

Even before this effort by theorists, the business community had discovered the benefits of easy to use software for everyday workplace tasks. Spreadsheets and efficient human interfaces, like that on the Apple MacIntosh computers, are now a common part of every office environment. The human interface concepts based on "mouse" cursor control and desktop iconography were developed in AI research at the Xerox PARC Lab in Palo Alto in the early 1970s. We expect successful personal computer software techniques, as well as Gibbs concepts will be combined with a new generation of cheap personal science work stations to produce a spreadsheet friendly front end for experiment analysis. (Early work in this direction at SLAC and CERN was reported at the recent Asilomar Conference on Computing in High Energy Physics*(71)*.) Software that allows an experimenter to request histograms with a few clicks of a mouse and to define the kinematic variables in standard math notation will be an appropriate match to hardware engines that can process a large experiment data base in 30 minutes.

A subtle irony underlies the impressive worldwide work on experiment trigger hardware described in Sections 2 and 3.3. The high level language programmable processors are somehow less trusted, and less utilized, for large trigger reductions than middle and low level devices which are much harder to program. The issue has to do with trust in the correctness of the programs. Because of the difficulty in preparing them, programs for the lower level triggers are simpler and, thereby, more readily tested. High level systems are clearly capable of complex on line programs and selection criteria as severe as off line analysis that ultimately leads to publication of results. However, an event discarded on line is lost forever, and experiments are naturally very careful to test trigger programs thoroughly. Anyone who has wrestled with a large analysis or reconstruction program appreciates how long it takes before anything approaching certification of correctness can be made.

Interaction rates at 20 TeV hadron colliders are expected to be 10^9 per second according to SSC design studies*(13)*. Trigger reductions of 10^8 will be required to get down to data logging rates of 1 Hz. Unbiased triggers that can accomplish this will be so complex that they will only be feasible in high level languages even in environments we now see as middle and low level triggers. Developing high level processors that can handle the rates required will be accomplished by suitably parallel, probably tree structured architectures for data acquisition. In fact, there are high level processors existing today with the ability to handle data rates approaching 200 Mbytes/sec. The real problem will not be rates, but the trigger program confidence issue which is already making itself felt.

Modern software engineering techniques, such as "structured analysis, structured design" (SASD), are beginning to be applied in large experiments*(72)*.

This work is a precursor to what will be essential: establishing techniques for developing and testing certified trigger code. Moreover, it will have to be possible to do this in almost real time, as the need of experiments for triggers that can be changed during a run is not likely to disappear. Rigorous certification of large and critical software projects is hardly a problem limited to high energy physics. In fact, this issue is at the heart of some important political debates about, for example, the Strategic Defense Initiative and nuclear reactor control systems. One can expect, therefore, that particle physics will not have to confront this problem alone. Nonetheless, the issue involves basic and unsolved difficulties of artificial intelligence research. For example, the desire will be to test not only whether the program is in fact carrying out the intentions of the designers, but whether those intentions are correct in the context of the detector parameters, known physics, and a miscellany of other variables.

It is a common trait of scientists to dismiss as "trivial" problems for which the solution path is understood. Development of new hardware for the computing intensive problems of experimental and theoretical particle physics will follow paths well established today, though the effort can hardly be called trivial. We are not as sure about directions for attacking the lattice gauge algorithm, human interfacing, and software verification issues. These are also real and serious impediments to progress in particle physics. Finding approaches which have promise of resolving them is therefore a pressing, and truly nontrivial, concern.

Acknowledgments

We would like to thank our colleagues on the Advanced Computer Program for their stimulating support. M. Fischler and C. Quigg were helpful on areas with which we were not familiar. We thank the many individuals and groups working in this field for responding to our request for information. We apologize that space did not permit us to describe most of the work in any detail. Our choice of efforts to describe was based more on consideration of which would make the clearest examples than on any quality judgement. We thank L. Rauch for preparing the bibliography and manuscript. The authors are supported by funds from the U.S. Department of Energy.

Literature Cited

1. Rossi, B., *Nature* 125:636 (1930)

2. Wilson, K. G., *Phys. Rev. D-8* 10(8):2445-459 (1974)

3. Blackett, P. M. S., Occhialini, G. P. S., *Proc. Roy. Soc.* A139:699-726 (1933)

4. Fukui, S., Miyamoto, S., *Nuovo Cimento* 11(10):113-15 (1959)

5. *Collider Detector at Fermilab (CDF) Activity Report,* Batavia:Fermilab, December 1986

6. Ballam, J., Chairman, *Report of Ad Hoc Committee on Future Computing Needs for Fermilab,* FERMILAB-TM-1230 99 pp. December, 1983

7. L3 Collaboration, *Technical Proposal [for LEP Experiment, submitted to CERN]* May, (1983)

7a. Newman, H., See Ref. 18. In Press.

8. Throughout this article, we use the VAX 11/780 performance as a standard unit (VAXes). Relative performance of other machines are taken from *Computing for Particle Physics, Report of the HEPAP Subpanel on Computer Needs for the Next Decade, August 1985,* DOE/ER-0234 p. 74 (1985)

9. Hasenfratz, A., Hasenfratz, P., *Ann. Rev. Nucl. Part. Sci.,* 35:559-604 (1985)

10. Batrouni, G. G., Katz, G. R., Kronfeld, A. S., Lepage, G. P., Svetitsky, B., et al., *Phys. Rev. D* 32(10):2736-746 (1985)

11. Fox, G. C., Otto, S. W., *Physics Today,* 37(5):50-59 (1984); Fox., G., The Performance of the Caltech Hypercube in Scientific Calculations. In *Supercomputers--Algorithms, Architectures and Scientific Computation,* ed. F. A. Masten, T. Tajima, Texas:University of Texas, (1985)

12. Dorenbosch, J., See Ref. 13, pp. 134-151 (1985)

13. Cox, B., Fenner, R., Hale, P., ed. *Proc. of the Workshop on Triggering, Data Acquisition and Off Line Computing for High Energy/High Luminosity Hadron-Hadron Colliders.* 473 pp. (1985)

14. Michelini, A., Dobinson, R. W., Hoekemeijer, A., Innocenti, P. G., Jones, T., et al., ed. *Proc. Topical Conference on the Application of Microprocessors to High Energy Physics Exp.,* CERN 81-07 614 pp. Geneva:CERN (1981)

15. Istituto Nazionale di Fisica Nucleare Sezione di Padova, ed. *Proc. of Three Day In-Depth Review on the Impact of Specialized Processors in Elementary Particle Physics, Padova, 1983,* 373 pp. (1983)

16. Donaldson, R., Kreisler, M. N., ed. *Proc. Symposium on Recent Developments in Computing, Processor and Software Research for High-Energy Physics, Guanajuato, Mexico, 1984,* 459 pp. (1984)

17. Hertzberger, L. O., ed. *Proc. of the Conference held in Amsterdam, The Netherlands, 1985, on Computing in High Energy Physics,* 431 pp. Amsterdam:North-Holland (1986)

18. Ash, W. W., ed. *Proc. of Conference on Computing in High Energy Physics held in Asilomar State Beach, California,* Amsterdam:North-Holland. In Press.

19. *IEEE Conferences on Real-Time Computer, Applications in Nuclear and Particle Physics, held in Santa Fe,* NS-26(4):4365-678 (1979)

20. *IEEE Conferences on Real-Time Computer, Applications in Nuclear and Particle Physics, held in Oak Ridge,* NS-28(5):3667-928 (1981)

21. *IEEE Conferences on Real-Time Computer, Applications in Nuclear and Particle Physics, held in Berkeley, 1983,* NS-30(5):3721-4024 (1983)

22. *IEEE Conferences on Real-Time Computer, Applications in Nuclear and Particle Physics, held in Chicago, 1985,* NS-32(4):1260-1495 (1985)

23. Verkerk, C., ed. *Proceedings on the 1984 CERN School of Computing, Aiguablava, Catalonia, Spain,* CERN 85-09, 376 pp. Geneva:CERN (1985)

24. Verkerk, C., ed. *Proceeding on the 1982 CERN School of Computing, Zinal, Valais, Switzerland,* CERN 83-03, 350 pp. Geneva:CERN (1983)

25. Verkerk, C., ed. *Proceedings on the 1980 CERN School of Computing, Vraona-Attiki, Greece,* CERN 81-03, 408 pp. Geneva:CERN, (1981)

26. Verkerk, C., ed. *Proceedings on the 1978 CERN School of Computing, Jadwisin, Poland,* CERN 78-13, 256 pp. Geneva:CERN, (1978)

27. Macleod, G. R., Chairman, *Proceedings on the 1976 CERN School of Computing, La Grande Motte, France,* CERN 76-24, 283 pp. Geneva:CERN, (1976)

28. Macleod, G. R., Chairman, *Proceedings on the 1974 CERN School of Computing, Godoysund, Norway,* CERN 74-23, 438 pp. Geneva:CERN (1974)

29. Videau, I., *IEEE Trans. Nucl. Sci.,* 32(4):1484-1489 (1985)

30. Amidei, D., Campbell, M., Frisch, H., Grosso-Pilcher, C., Hauser, J., et al. *A Two Level Fastbus Based Trigger System for CDF, CDF Note No. 510,* Internal Document, Submitted to Nucl. Instr. Meth. (1987)

31. Brafmann, H., Breidenbach, M., Hettel, R., Himel, T., Horelick, D., *IEEE Trans. Nucl. Sci.,* NS-25(1):692-97 (1978)

32. Jacobs, D. A., *Computer Physics Communications* 26:69-77 (1982); Bähler, P., Bosco, N., Lingjaerde, T., Ljuslin, C., van Praag, A., Werner, P. See Ref. 17, pp. 283-86 (1986)

33. Droege, T., Gaines, I., Turner, K. J., *IEEE Trans. Nucl. Sci.,* NS-25(1):698-703 (1978)

34. Drake, G., Droege, T. F., Nelson, C. A., Jr., Turner, K. J., Ohska, T. K., *IEEE Trans. Nucl. Sci.,* NS-33(1):92-97 (1986)

35. Breidenbach, M., Frank, E., Hall, J., Nelson, D., *IEEE Trans. Nucl. Sci.,* NS-25(1):706-15 (1978)

36. Barsotti, E., Appel, J. A., Bracker, S., Haldeman, M., Hance, R., et al. *IEEE Trans. Nucl. Sci.,* 26(1):686-96 (1979)

37. Beer, A., Bourgeois, F., Corre, A., Critin, G., Huber, M. L., et al. *Nucl. Inst. Meth.* 160:217-25 (1979)

38. Avilez, C., Borten, L., Christian, C., Church, M., Correa, W., et. al. See Ref. 16, pp. 45-54 (1984)

39. Martin, J., Bracker, S., Hartner, G., Appel, J., Nash, T. See Ref. 14, pp. 164-77 (1981)

40. Kunz, P. F., *Nucl. Inst. Meth.* 135:435-40 (1976); See also Hungerbühler, V., Mauron B., Vittet, J. P., *Nucl. Inst. Meth.* 137:189-92 (1976) for another project with these ideas at about the same time.

41. Verkerk, C., See Ref. 25, pp. 282-324 (1981)

42. Halatsis, C., Joosten, J., Letheren, M. F., van Dam, A., *Proceeding of 7th Annual Symposium on Computer Architecture, 1980, La Baule, France; New York:IEEE publication* 80CH1494-4C (1980)

43. Kunz, P. F., Fall, R. N., Gravina, M. F., Halperin, J. H., Levinson, L. J., et al. *IEEE Trans. Nucl. Sci.* NS-27(1):582-86 (1980)

44. Lord, E., Kunz, P., Botterill, D. R., Edwards, A., Fucci, A., et al. See Ref. 14, pp. 341-54 (1981)

45. Kunz, P. F., Gravina, M., Oxoby, G., Trang, Q., Fucci, A., et al. See Ref. 15, pp. 83-100 (1983)

46. Brafman, H., Fall, R., Gal, Y., Yaari, R. See Ref. 15, pp. 71-81 (1983)

47. Notz, D. *A Data Processing System Based on the 370/E Emulator,* DESY 85-046, 17 pp. Hamburg:DESY (1985)

48. Gosman, D., Hertzberger L. O., Holthuizen D. J., Por, G. J. A., Schoorel, M. See Ref. 14, pp. 70-82; and See Ref. 14, pp. 83-90 (1981)

49. Cittolin, S., Demoulin, M., Haynes, W. J., Jank, W., Pietarinen, E., Rossi, P. See Ref. 16, pp. 413-27 (1984)

50. Mertens, V., von der Schmitt, H., See Ref. 15, pp. 257-76 (1983)

51. Bracker, S., Nash, T., Gaines, I., See Ref. 15, pp. 277-301 (1983); Gaines, I., Areti, H., Atac, R., Biel, J., Cook, A., et al. See Ref 18. In Press; Biel, J., Areti, H., Atac, R., Cook, A., Fischler, M., et al. See Ref. 18. In Press.
[A similar concept was conceived at UCLA and CERN where the VIRTUS processor was to use a CERN 68000 FASTBUS module, a combination which is not competitive in this application. See Ellet, J., Jackson, R., Ritter, R., Schlein, P., Yaeger, D., Zweizig, J., See Ref. 17, pp. 235-39 (1986); Müller, H., See Ref. 17, pp. 240-46 (1986)]

52. Cutts, D., Hoftun, J. S., Johnson, C. R., Zeller, R. T., Trojak, T., van Berg, R. See Ref. 17, pp. 287-91 (1986)

53. DØ Management Plan and Cover Agreement Fermilab, Internal Document, 76 pp. (1985); and see Johnson, T., Durham, T., see Ref. 54, p. 410 (1986)

54. Johnson, T., Durham, T., *Parallel Processing: The Challenge of New Computer Architectures* p. 410, England:Ovum Ltd (1986)

55. Siskind, E. J., See Ref 16, pp. 281-84 (1984)

56. Sherdan, D. J., *IEEE* NS-32(4):1479-1483 (1985)

57. Beretvas, A., Carroll, J. T., Devlin, T., Flaugher, B., Joshi, U., et al., *Proceedings of 3rd Pisa Meeting on Advanced Detectors, Castiglione della Pescaia, Italy, 1986;* In Press.

58. Oothoudt, M., Naivar, F., Smith, W., *Use of the Fermilab Advanced Computer Project (ACP) for MEGA On Line High-level Triggering and Off Line Data Analysis,* MEGA internal document, Los Alamos, November 1, 1985; and Oothoudt, M., private communication.

59. Carroll, J. T., Cittolin, S., Demoulin, M., Fucci, A., Martin, B., et al., See Ref. 15, pp. 47-70 (1983)

60. Cittolin, S., See Ref. 17, p. 278 (1986)

61. Rankin, P., Bricaud, B., Gravina, M., Kunz, P. F., Oxoby, G., et al., *IEEE Trans. Nucl. Sci.,* NS-32(4):1321-325 (1985); Paffrath, L., et. al., *IEEE Trans. Nucl. Sci.,* NS-33:793-96 (1986)

62. Pearson, R. B., Richardson, J. L., Toussaint, D., *Comm. of the ACM* 28:385-89 (1985)

63. Seitz, C. L., *Comm. of the ACM* 28(1):22-33 (1985); Brooks, E., Fox, G., Johnson, M., Otto, S., Stolotz, P., et al., *Phys. Rev. Lett.* 52(26):2324-327 (1984); Otto, S. W., Stack, J. D., *Phys. Rev. Lett.* 52(26):2328-331 (1984)

64. Hey, A. J. G., Jesshope, C. R., Nicole, D. A., See Ref 17, pp. 363-69 (1986)

65. Rogstad, D. H., *AMPlifier (Jet Propulsion Laboratory, Pasadena, California)* 1(1):5-6 (1986)

66. Terrano, A., See Ref. 15, pp. 135-153 (1983); Christ, N. H., Terrano, A. E., *IEEE Trans. Comp.* C-33(4):344-50 (1984); Christ, N. H., private communication

67. Christ, N. H., Terrano, A. E., *Phys. Rev. Lett.* 56:111-14 (1986)

68. Bacilieri, P., Cabasino, S., Marzano, F., Paolucci, P., Petrarca, S., et al., See Ref. 17, pp. 330-37 (1986)

68a. Brower, R. C., Giles, R. C., Maturana, G., See Ref. 17, pp. 339-344 (1986).

69. Beetem, J., Denneau, M., Weingarten, D., *J. Stat. Phys.* 43:1171-183 (1986)

70. The GIBBS Group (includes Bergmark, D., Demers, A., Gries, D., Lepage, P., Moitra, D., et al.) See Ref. 16, pp. 89-96 (1984)

71. Burnett, T., See Ref. 18. In Press; Brun, R., Bock, R., Conet, O., Marin, J. C., et al., See Ref. 18. In Press.

72. Kellner, G., See Ref. 18, In Press; Palazzi, P., Brazioli, R., Fisher, S. M., Zhao, W., et. al., See Ref. 18. In Press.

TABLE 1 Representative computing intensive experiments

Experiment	Location	Events/CYr[c]	VAX-sec/Event	VAX-yrs[d]/CYr[c]
E691	Fermilab fixed target	1×10^8	7	30
UA1(6)	CERN $\bar{p}p$	5×10^6	60	20[b]
CDF[a](6)	Fermilab $\bar{p}p$	10^7	200	50-100
L3[a](7)	CERN LEP	4×10^6	> 120	48-60
Anticipated SSC Experiment(7a)		10^7	1000	1220

[a] Estimated

[b] Constrained by limited computer resources

[c] Calendar Year (CYr) includes typical beam on and off times

[d] Includes analysis and simulations

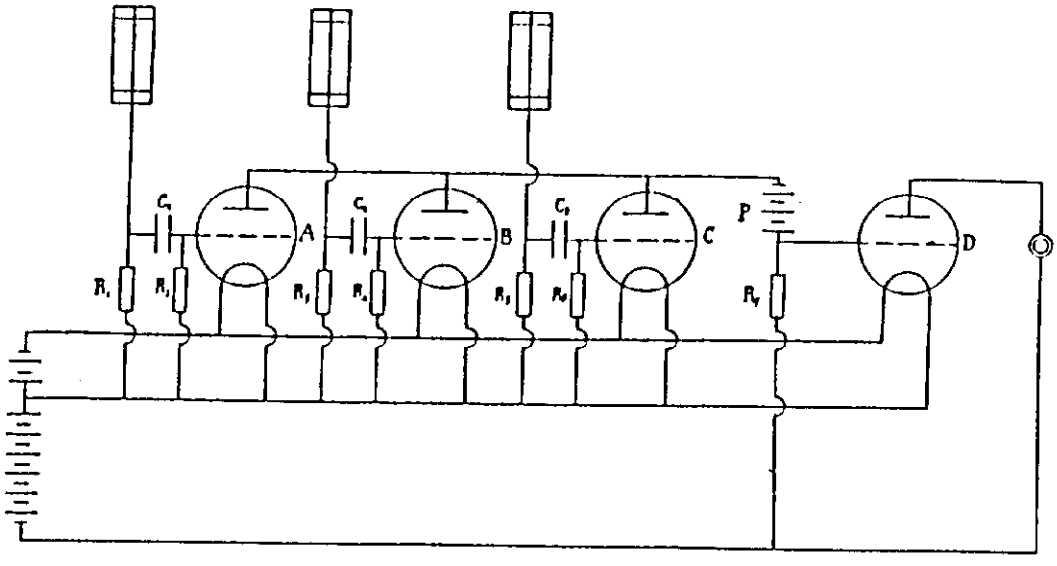


Figure 1. Particle physics computer technology in 1930: The first practical electronic AND gate, a Geiger Counter triple coincidence circuit. Pulses were "detected by a telephone" and scaled manually.

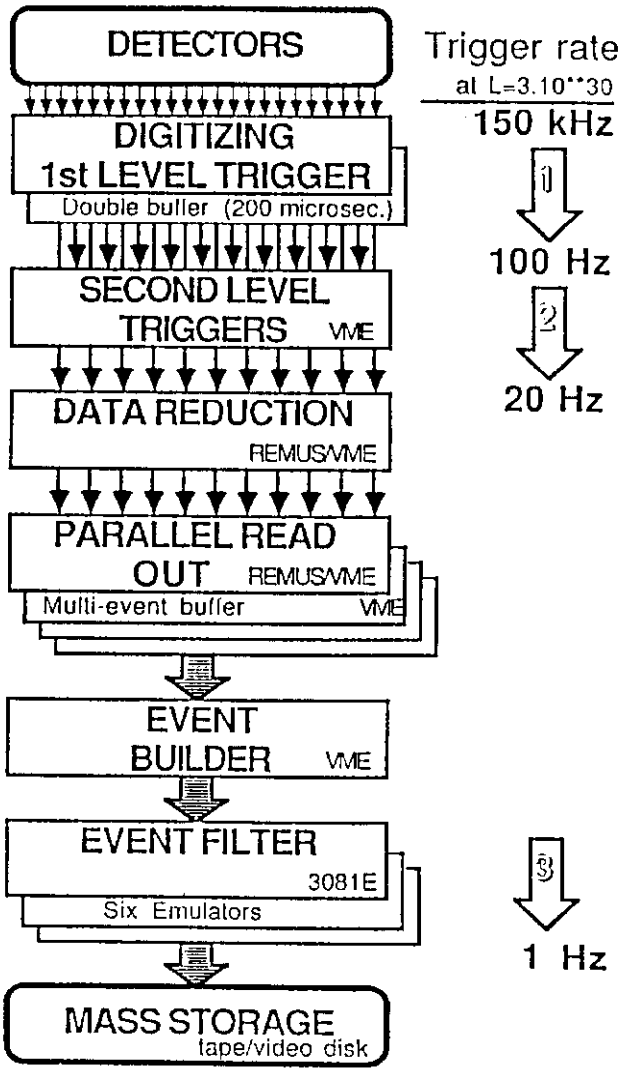


Figure 2. Overview of the UA1 multilevel trigger and data read out system. The numbers at right give the surviving event rate after each level of the trigger. Dead time is < 10%.

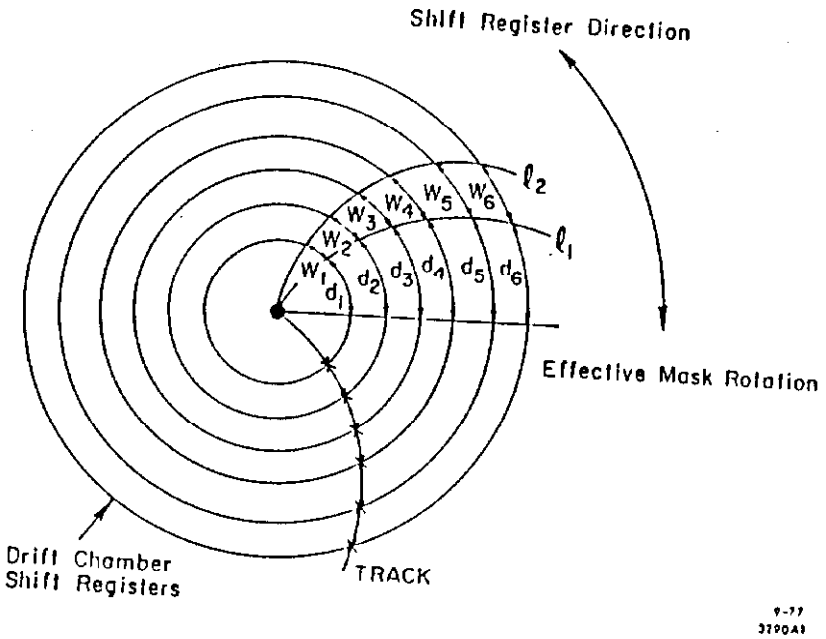
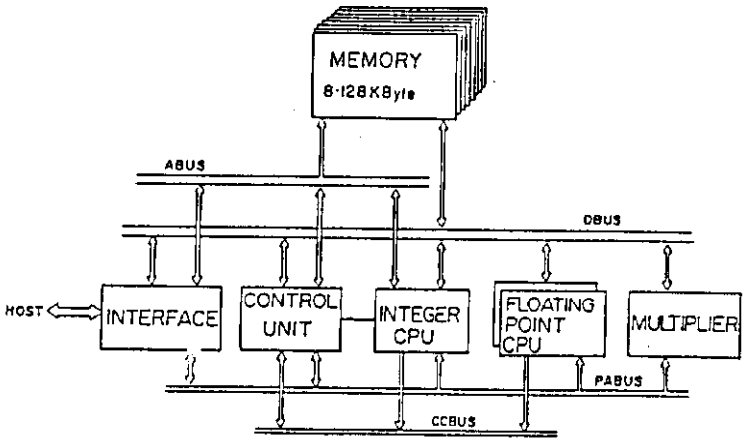
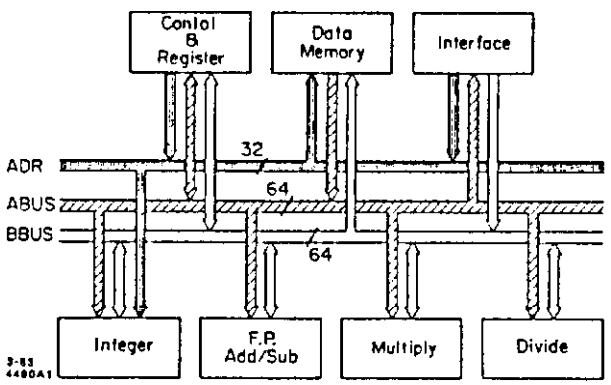


Figure 3. Mark II track finder mechanism. The effective mask for accepting a track is defined by the delays (d_i) necessary to shift the curved track into a straight line. Road widths are defined by the w_i.



(a)



(b)

Figure 4. Block diagrams of two modern emulators.
a) 370/E;
b) 3081/E.

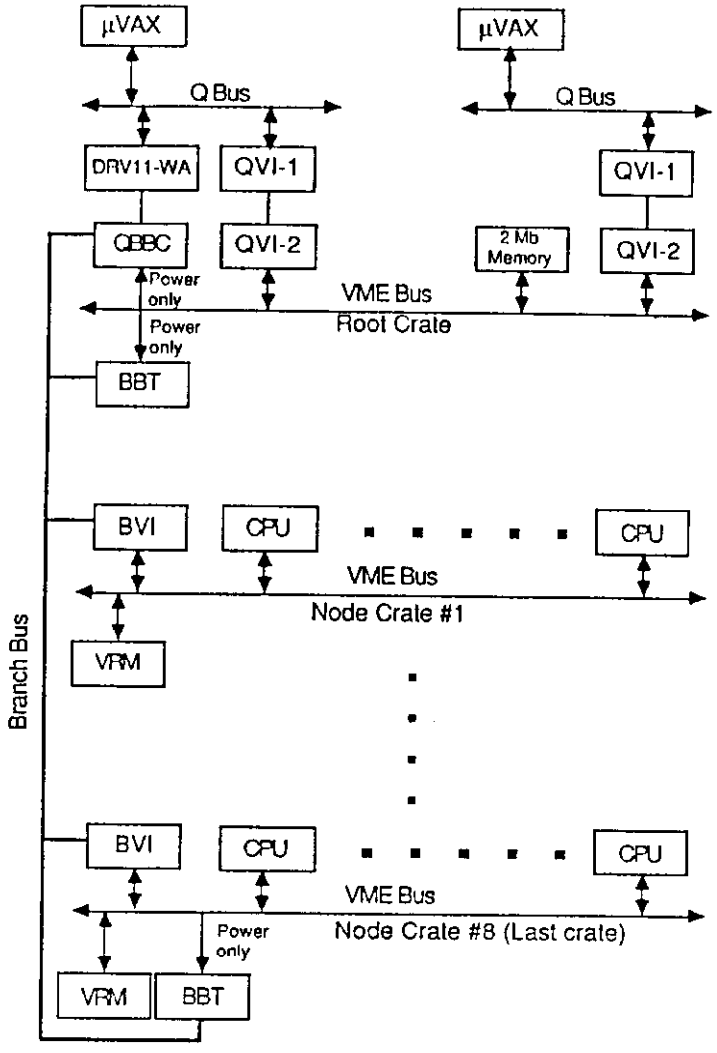
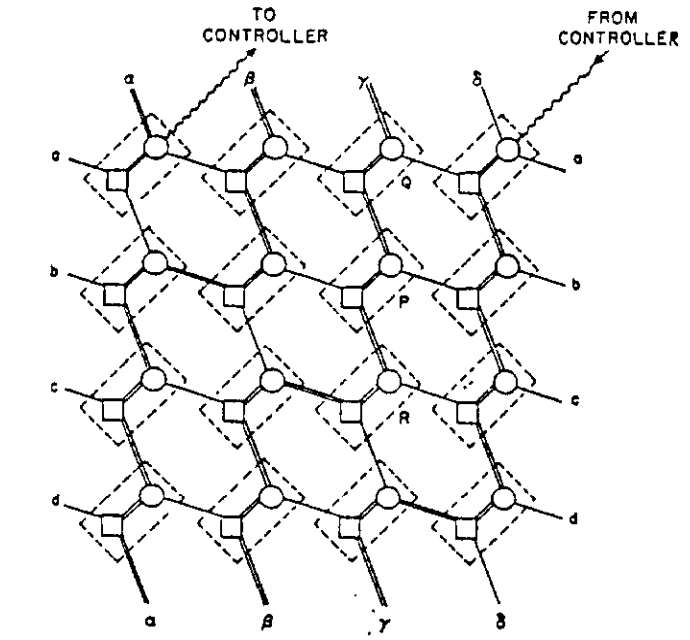
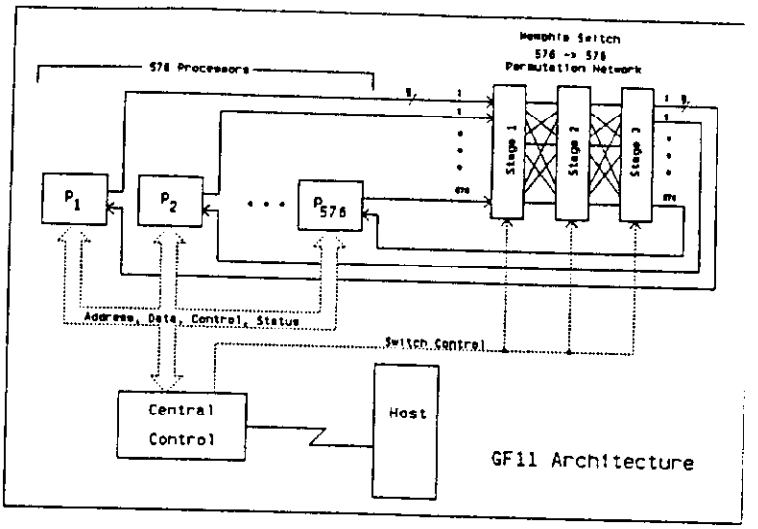


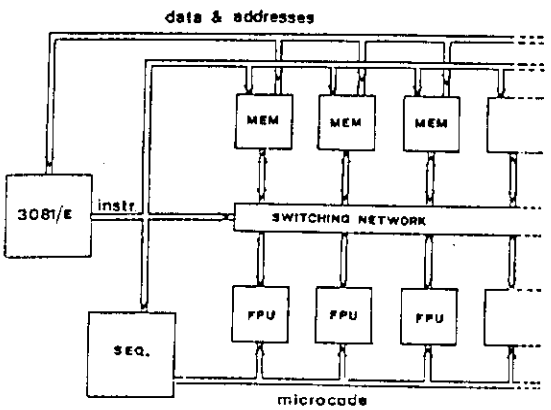
Figure 5. Block diagram of first ACP Multiprocessor in the Fermilab Computer Center.



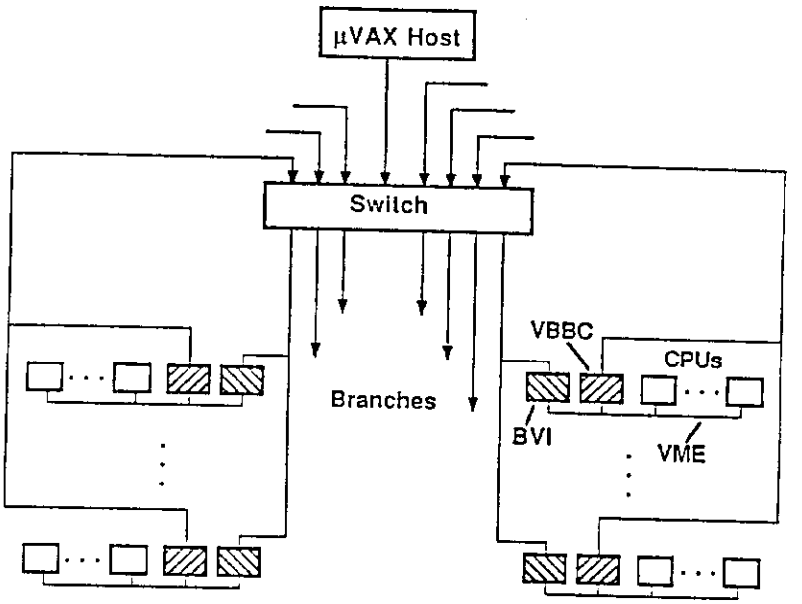
(a)



(c)



(b)



(d)

Figure 6. Lattice gauge processor architecture.
a) The Columbia 4x4 toroidal grid;
b) The Italian APE machine;
c) IBM's GF11;
d) ACP Multiprocessor.