Presented at 1983 Conf. on Real-Time Computer Applications in Particle and Nuclear Physics, Berkeley, Calif., May 16-19, 1983

CONF-830576--4

PARALLEL PROCESSOR FOR FAST EVENT ANALYSIS

D. C. Hensley
Oak Ridge National Laboratory*, Oak Ridge, Tennessee 37830

CONF-830576--4

DE83 014050

Summary

Current maximum data rates from the Spin Spectrometer of ~5000 events/s (up to 1.3 MBytes/s) and minimum analysis requiring at least 3000 operations/event require a CPU cycle time near 70 ns. In order to achieve an effective cycle time of 70 ns, a parallel processing device is proposed where up to 4 independent processors will be implemented in parallel. The individual processors are designed around the Am2910 Microsequencer, the AM29116 μP, and the Am29517 Multiplier. Satellite histogramming in a mass memory system will be managed by a commercial 16-bit μP system.

Introduction

The Spin Spectrometer[1] at the Holifield Heavy Ion Research Facility is a 4π γ-ray detector system consisting of up to 72 NaI detectors of equal solid angle. Heavy ion reactions of current interest produce more than 20 γ rays on the average and from 2 to 12 neutrons. Consequently, the average number of detectors involved in a single event may be 30 or more, and the number of bytes per event readily averages 250 or more. The Event Handler driven CAMAC based data acquisition system[2] is currently limited by ADC conversion time and data read-out time to event rates of about 5000 events/s.

The present possible data acquisition rate exceeds our ability either to transmit data to the computer or to spool it thence onto tape. Thus, any preprocessing which might reduce the data transmission rate without reducing the event rate is highly desirable. Further, various levels of processing of the data are required before detailed analysis can be done, and ways of speeding up these processing phases are clearly needed. To perform the standard first pass processing, approximately 3000 CPU operations per average event are needed as a minimum, and more detailed processing would definitely be desired. A rate of 5000 events/s with 3000 operations/event implies an operation time of ~66 ns. Although many applications of such a processor will be for significantly lower data rates, a CPU cycle time less than 70 ns seems a desirable goal, since the analysis might be correspondingly more complicated.

An effective CPU cycle time less than 70 ns can most easily be achieved for this application with the use of parallel processors since the data stream has natural divisions at event boundaries. The processing of a particular event can be handled essentially independently of that for another event. If it is assumed that the necessary analysis programs are both relatively simple (no floating point calculations) and relatively small (<4k instructions), a CPU cycle time near 200 ns is reasonable, and a set of 4 parallel processors would achieve an effective cycle time near 70 ns.

Overview of the System

The parallel processing system will consist of the following modules (see Fig. 1): A commercial

*Operated by Union Carbide Corporation under contract W-7405-eng-26 with the U.S. Department of Energy.

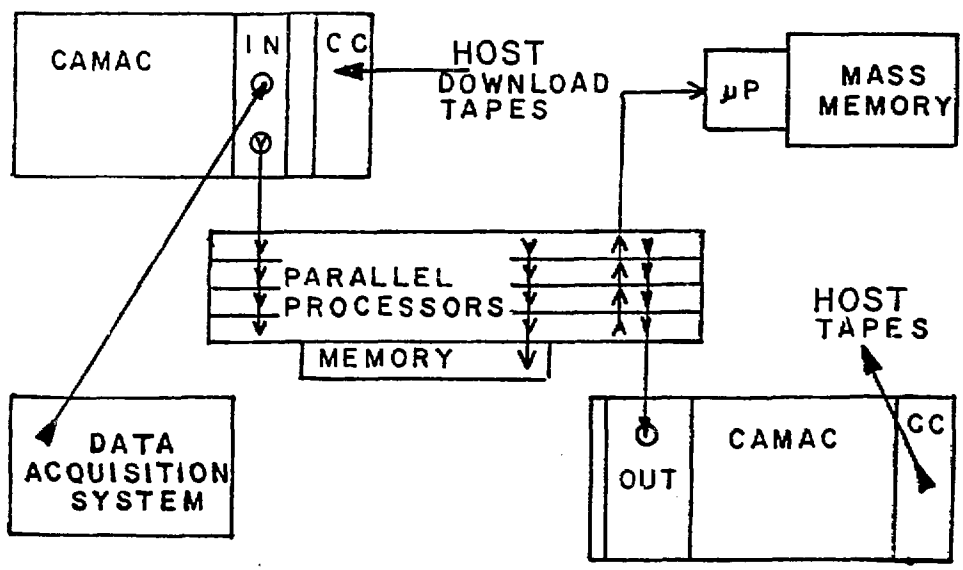


FIG. 1

crate controller(s) for interfacing to the host computer(s). There will be an INPUT module and an FIFO-OUTPUT module in CAMAC. In addition to the parallel processors (PP), there will be a shared MEMORY module, and a random-time access OUTPUT module interfacing to a mass memory and μPs system.

The crate controller permits the host computer to download programs into the processors and to place constants/data into the processors and the memory. When the processor system is running, the host computer uses the INPUT/OUTPUT modules to effect block transfers of data with the processors.

The INPUT module has two distinct functions, downloading the processors and memory and distributing entire events to different processors. The event stream is extracted from an incoming FIFO data stream.

Downloading the processors requires that the INPUT module be able to select one or more processors, disable it, and transfer both addresses and program code--memory verification would be done using the same interface. In the usual case where each processor uses the same program, all processors could be downloaded in parallel. The INPUT module has no direct connection to the data storage of the processors, so it must use a processor program to transfer data through an input register to these memories.

Data for processing is input through a FIFO memory of at least 4k words. Input to this memory can be from the host computer via the CAMAC dataway or from the data acquisition system (DAS) through a front panel input. The INPUT module handles the processor selection and the transferring of an entire event to an available processor. Data transfers can be up to 24 bits wide, but the processors will typically analyze only 16 of the bits.

The OUTPUT module also contains a FIFO buffer memory of at least 4k words. The OUTPUT module controls which of the processors it will service, but it generally takes an entire event of data from a processor before it tries to service another processor. The module services the processors in a mixed mode of first-come-first-served and round-robin--this mode ensures that no processor can be locked out by the other processors. Note that the order of events out need no longer be the same as that in, but it should be reasonably close. The FIFO memory can be accessed via the CAMAC dataway or via a front panel FIFO output port.

The memory module is to contain 64k words (16 bits) of fast memory which can be accessed via a ribbon cable connected to the processors. Access on a request/grant basis will give the processors a large memory capability. The memory can be used to store memory-mapped-gates, free-form-gates, or other common information too extensive to be stored in the individual processor memories. It can also contain common information which is being changed during the course of the processing, such as a software stabilized gain. It can also serve as a histogramming memory for the processors.

The processor section of the parallel processors will be described below. The CAMAC link to the processors through the INPUT module allows the program and data memory to be written/verified by the host computer. The processor can be disabled/enabled through the INPUT module. The processor is able to generate both an "available" LAM and an "output ready" LAM, and FIFO block transfers of events is accomplished under processor program control through the INPUT and OUTPUT modules. No special provision has been made to allow processors to have different programs, but if one processor is programmed to be independent of the input event stream, it can run independently.

Details of the Processors

The processors in the parallel processor system will be optimized for 16 bit I/O and data. Even though CAMAC possesses a 24 bit capability, in practice at Holifield Heavy Ion Research Facility only 16 bit data acquisition is performed, and a 16 bit limitation seems quite acceptable.

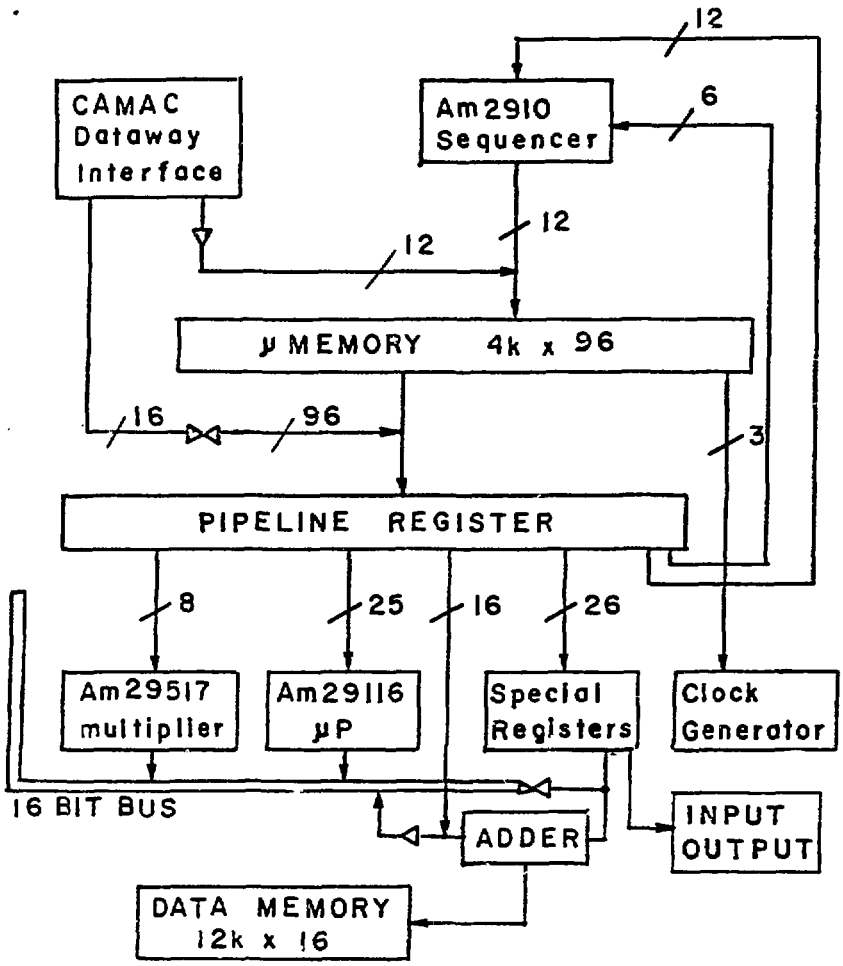
The program memory will be limited to 4k of microcode (96 bits maximum). Most programs should use far fewer operation locations than this for their execution, particularly since I/O will be limited to quasi-DMA interchanges with the INPUT and OUTPUT modules. The program memory will not be alterable by the program.

The work memory will consist of from 4k words (16 bits) to 12k words. Part of this space will be working space and part will consist of predetermined constants and data. The predetermined constants would include energy calibrations, ADC pedestals, etc., and other fixed constants such as reciprocals for performing fast divisions.

The processor will be built around three different devices, the Am2910 Microprogram Controller, the Am29116 16-bit Bipolar Microprocessor, and the Am29517 16 by 16 bit Parallel Multiplier as shown in Fig. 2. All of these devices feature cycle time capabilities of at least 100 ns, so that with 50 ns access-time memory an overall cycle time near 200 ns seems feasible. These devices will be driven in parallel by horizontal microcode as shown in Fig. 2. In principle the Am29116 and Am29517 could be working in parallel, but much code will not be able to use this feature because these devices share the 16-bit I/O bus. On the other hand, the Am2910 and the special registers will be running largely independently, and some parallel processing capability with them will be possible.

The Am2910 features a 12-bit address output, 4 address sources, a 5 deep subroutine stack, a conditional test input, an internal loop counter, and 16 microinstructions. Basically it allows absolute jumps, conditional jumps, and full subroutine capability, and it requires up to 11 controls in addition to a 12 bit address input and 12 bit output.

The Am29116 is a 16 bit, 100 ns cycle time processor with arithmetic logic and an accumulator, up to 15 bit rotations or shifts, bit oriented operations (set, mask, and clear), priority encoding, 32 internal registers storage, an input latch, and extensive condition code generation. It requires up to 26 control inputs in addition to its 16 I/O lines. It can use as its data sources various combinations of the internal registers, the accumulator, the input latch, the external inputs, or even the control inputs. Its result can be routed to the internal registers, the accumulator, the input latch, or to the output lines. The status word can be read or a simple test condition can be requested and sampled while the Am29116 is executing an instruction.



PROCESSING UNIT

Fig. 2

The Am29517 is a fast (~80 ns) 16 by 16 bit multiplier with the full 32 bit product multiplexed at the output. It can handle TWO's complement, unsigned, or mixed operands. It requires up to 12 control lines and has three separate 16 line input/output ports.

In addition to these three devices, there are many registers which perform various internal and interface functions. There will be address/index registers which allow data to be accessed at separately specified locations. These registers can be incremented at the end of a cycle in parallel with any of the main devices, and zero detection will be available for conditional testing. A random number register will allow smoothing of modified ADC information, and a shift/priority register will enhance the utility of the Am29116. A buffer register for the multiplier chip may be desirable.

Input/output will also be handled through registers. There will be an input and an output register for FIFO-like data transfers, and there will be a 32 bit wide register for output to a histogramming device. Finally a pair of registers will handle I/O with the memory module.

Satellite μ Processor

It is proposed that a satellite μP system accompany the parallel processor system. This would be a system built around a device comparable to the MC68000 μP and would include a mass memory (at least several MBytes), interfaces to the host computer and to the output module of the parallel processor system, and some very modest display capability. Its main use would be as a fast, large histogramming memory, or it could be used for first stage histogramming for very large histograms.

We typically use histogramming for two different phases of our experiments. The first phase is the monitoring of online data acquisition. Histograms of most of the parameters will be generated in order to check the gain, threshold, resolution, integrity, etc., of the parameters. This is done during the setup of the experiment and during the course of the experiment. Generally these histograms are not used in the final data reduction. In a few cases where the experimental system is reliable and relatively simple, these histograms may constitute all or most of the final data acquisition.

For more complicated data acquisition systems, the data are spooled directly onto magnetic tape, and the tapes are replayed to generate the final reduced data. In many cases, these replays generate very large histograms, but if the available histogramming space is large enough, the replaying of the tapes can be minimized.

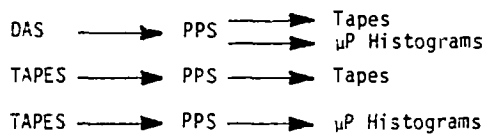
In either case, histogramming in the main computer can be unsatisfactory because it is both memory and CPU intensive. If the main computer does histogramming during data acquisition, this activity may limit the data rate or provide a too small sample of the data stream. During processing of data tapes, the histogramming function ties up the CPU and memory from other users and may cause the tape processing to proceed too slowly.

If the parallel processor system generates histogram addresses which it passes on to the satellite system, the entire histogramming load is removed from the main computer which is left free for tape I/O, interactive displaying, and data analysis.

In the event that the histogram space exceeds the size of the mass memory, the system can still be used effectively as a "prehistogrammer." Random add-one access to a 16 megachannel array on disk can proceed at an average rate near 10 kHz. If the satellite system had 4 MBytes of memory, it could assign 2 bits per channel to the histogramming and reduce the input rate to the main computer by a factor of 4. Then an average updating rate near 40 kHz could be achieved.

Use of the System

There are several intended uses of this parallel processing system (PPS). The event source can be either the data acquisition system (DAS) (1.3 MBytes/s maximum) or data tapes read by the main computer (0.7 MBytes/s maximum). The output destination can be either tapes on the main computer, histograms in the main computer, or histograms in the satellite μP system. The following are the main configurations to be used:



In all of these configurations, the main computer is required to perform block I/O which it does very efficiently, almost effortlessly, and it is not required to make calculations and decisions which it does powerfully but slowly.

Why This System

While it is reasonably clear that parallel processing is desirable, why design this particular special processor? The approach of the "MIDAS"[3] project at Berkeley has been to use a modern CPU, stripped of extras, while SLAC has developed the 168/E,[4] which emulates a subset of the IBM 360/370 instruction set. Both laboratories run their processors in parallel to achieve high computing power, and both systems can handle complicated calculations. A special strength of these systems is that they can use software which runs on the host computer. For a large class of experiments in nuclear physics, these approaches seem more powerful than necessary and hence more expensive and complicated. Thus, while they may present one of the best solutions to many problems, they may not be the best match to the present problem.

The current class of 16 bit microprocessors certainly presents another alternative. They have the advantage of relatively low cost, powerful instruction sets, and growing software support. On the whole, however, they seem to be mismatched to the current problem; their effective speed would probably be at least a factor of four too slow.

The proposed processor has certain weaknesses. It and its software have to be designed and debugged. It does not directly have a divide function. Floating point calculational capability has been ignored in its design. However, more than sufficient numerical precision can be achieved with 16 bit integers for all of the proposed calculations. Seldom is there a need to divide, and fast division can and will be achieved through multiplication by a reciprocal. The horizontal nature of the microcode appears to allow useful parallel operation that further enhances the processor's speed. On the whole, the proposed processor should in fact be especially effective in processing the Spin Spectrometer data, either online or offline.

References

1. D. G. Sarantites et al., Journal De Physique, Colloque C10, Supplement au n° 12, Vol. 41, C10-259 (1980).
2. D. C. Hensley, IEEE Trans. Nucl. Sci. NS-28, No. 5, 3720 (1981).
3. C. Maples, W. Rathbun, D. Weaver and J. Meng, IEEE Trans. Nucl. Sci. NS-28, No. 5, 3746 (1981).
4. P. F. Kunz et al., "The LASS Hardware Processor," Proc. 11th Annual Microprogramming Workshop, SIGMICRO Newsletter 9, 25 (1978).

DISCLAIMER