TM-747
2500.000
Oct. 1977

"THE M7 - A HIGH SPEED DIGITAL PROCESSOR FOR SECOND LEVEL TRIGGER SELECTIONS"

T. F. Droege, I. Gaines, K. J. Turner
Fermi National Accelerator Laboratory, Batavia, Illinois 60510

Summary

A digital processor is described which reconstructs mass and momentum as a second level trigger selection. The processor is a five-address, microprogrammed, pipelined, ECL machine with simultaneous memory access to four operands which load two parallel multipliers and an ALU. Source data modules are extensions of the processor.

Introduction

As High Energy Physics experiments become larger and more complex, the amount of data collected per experiment has become so large that central computer facilities can no longer process the data as it is collected. Larger computer centers do not appear to be the whole answer since even higher data rate experiments are foreseen. Often preliminary screening by a relatively simple program will greatly reduce the quantity of data to be analyzed, since some experiments have only a very small fraction of interesting events. Every effort is made through the use of fast trigger electronics in the experimental design to select only good events for storage (on tape usually) and later analysis. Often the fast trigger electronics is too complex to be physically realizable and there has been no alternative but to write tapes which are mostly full of junk for later selection at the central computing facility.

Experiment #400 at Fermilab (Figure 1) is a particle search experiment using a dimuon trigger. Based on previous experiments, it is known that the dimuon spectrum is dominated by low mass dimuons of no particular interest. E-400 will trigger on only high mass and high transverse momentum muon pairs so that the experiment can be run at high enough intensity to search down to theoretically interesting cross sections. The M7 processor has been designed to quickly analyze the tracks from a pair of x-y muon drift chambers of the delay line type[1] and to provide fast determination of mass and transverse momentum prior to writing the event on tape.

Engineering Considerations

While a general goal has been the application of special purpose digital processors to the enrichment of recorded experimental data and their possible use as off-line track processors, the specific goal of this project was the improvement of the fraction of good events taken by E-400 at Fermilab.

To meet the goal of two orders of magnitude trigger reduction with reasonable dead time it is necessary to scan the input detectors and compute the dimuon mass and momentum in the order of ten microseconds.

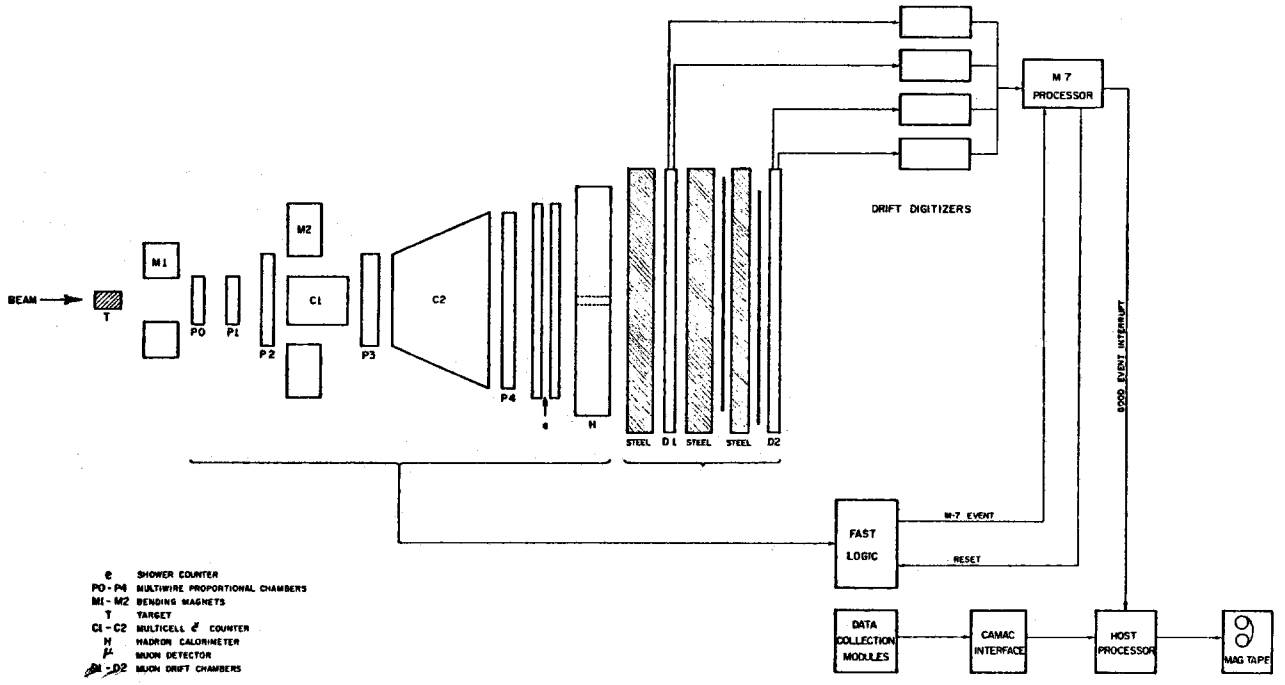


Fig. 1. Configuration of Experiment 400 at Fermilab with the M7 Processor used for Second Level Trigger Selection

This requires fast computation as well as fast digitization of the chamber drift time and fast scan of the drift digitization modules. An arbitrary time budget goal of 5 microseconds was set for the entire operation.

It is immediately apparent that the most common present technique of High Energy Physics Digital Data Collection[2] (CAMAC) is completely inadequate for trigger processors. For CAMAC to scan the 80 time digitizer modules for the expected four hits would exceed the total time budget by an order of magnitude. It was therefore necessary to design a high speed bus for this application. The physics community has recognized this limitation of CAMAC and work is underway to develop a standard bus for high speed data collection.[3] To reduce engineering time it was decided to modify standard CAMAC crates to incorporate a token passing scheme similar to the DEC UNIBUS[4] which allows data to be located in a CAMAC crate in under a microsecond.

The drift chamber digitizers were further divided into four quadrants, each of which would expect one hit in a normal event and these were arranged to be scanned in parallel. To measure delay line times to a resolution of 1ns while avoiding problems of 1 GHz clocks and scalers or long run down times, vernier type digitizers[5] were designed which run down in less than 1μs. This time is overlapped with the read out scan time.

Analysis of the mass and momentum equations (Figure 2) indicated that they could be efficiently solved by a processor which could perform ax ± by → c. In fact, 14 of 18 program steps (Figure 3) make full use of this operation. Meeting the goal of a 5 microsecond trigger requires that this operation be performed in about 200ns. Since error analysis indicated the need for 9 bit precision, parallel combinatorial multipliers were selected as the only available devices which could match this speed.

$$P_\perp^2 = \frac{K^2\left[\left(A_1X_2 - B_1X_1\right)^2 + \left(A_2Y_2 - B_2Y_1\right)^2\right]}{\left(A_3Y_2 - B_3Y_1\right)^2}$$

$$M^2 = \frac{K^2\left[\left(A_1\Delta X_2 - B_1\Delta X_1\right)^2 + \left(A_2\Delta Y_2 - B_2\Delta Y_1\right)^2\right]}{\left(A_3Y_2 - B_3Y_1\right)\left(A_3\bar{Y}_2 - B_3\bar{Y}_1\right)}$$

Fig. 2. Equations Solved by the M7 Processor for Fermilab Experiment 400

At first it was planned to build a fast very special processor which could efficiently perform successive operations of the form ax ± by → c. From the beginning it was planned to fetch the four operands simultaneously from memory and have two parallel multipliers. The initial plan was to make the processor a very simple microcoded processor which solved the specific equations with a second hardware processor which would scan the input data for track candidates. Later it was realized that the addition of a few instructions would allow the same processor to efficiently search for tracks.

1. $\bar{X}_1 - X_1 \rightarrow \Delta X_1$
2. $\bar{X}_2 - X_2 \rightarrow \Delta X_2$
3. $\bar{Y}_1 - Y_1 \rightarrow \Delta Y_1$
4. $\bar{Y}_2 - Y_2 \rightarrow \Delta Y_2$
5. $a_1X_2 - b_1X_1 \rightarrow A$
6. $A_2Y_2 - b_2Y_1 \rightarrow B$
7. $(A^2 + B^2)/256 \rightarrow C1$
8. $a_1\bar{X}_2 - b_1\bar{X}_1 \rightarrow A$
9. $a_2\bar{Y}_2 - b_2\bar{Y}_1 \rightarrow B$
10. $(A^2 + B^2)/256 \rightarrow C2$
11. $a_1\Delta X_2 - b_1\Delta X_1 \rightarrow A$
12. $a_2\Delta Y_2 - b_2\Delta Y_1 \rightarrow B$
13. $(A^2 + B^2)/256 \rightarrow C3$
14. $a_3Y_2 - b_3Y_1 \rightarrow D1$
15. $a_3\bar{Y}_2 - b_3\bar{Y}_1 \rightarrow D2$
16. $K_1C_1 - D1^2 \geq 0?$
17. $K_1C_2 - D2^2 \geq 0?$
18. $K_2C_3 - D1D2 \geq 0?$

Fig. 3. An 18 Instruction Program Implementing the Equations from Figure 2, where the Particle Coordinates are $x_1x_2\ y_1y_2$ for the First Track and $\bar{x}_1\bar{x}_2\ \bar{y}_1\bar{y}_2$ for the Second Track. K1 and K2 Determine $P_\perp{}^2$Min. and M^2Min.

Many of the design decisions were made to reduce the amount of time required to build this processor since to a great extent real costs to the Laboratory (i.e., what fraction of total laboratory effort went into this project x total budget = real cost) are proportional to calendar time and in any case were an order of magnitude higher than the identifiable parts cost.

ECL logic was chosen over Schottky TTL even though TTL would have been fast enough because it is believed that noise problems are easier to solve with ECL.

While the design was in progress the 2 x 4 bit ECL multiplier chips[6] became available increasing the processor speed by a factor of two and eliminating the need for level translation in and out of TTL multipliers.

Standard ECL wire wrap panels were selected for packaging the electronics with several panels mounted together on a single ground plane. This plane was then mounted vertically in a relay rack. Interconnections were wire wrapped single ended with only moderate care taken to push all wiring close to the ground plane. All connections were terminated in 100 ohms. While this is not the optimum interconnection technique for ECL, the result is only a small increase in rise time and overshoot, particularly if care is taken in locating major units so that most critical wiring runs are short.

An interesting feature of High Energy Physics (HEP), is the statistical nature of the data. Compared to payroll processing computers, for example, it is not necessary to get the right answer all the time. Since all the data taken will be later completely re-analyzed the worst result of an occasional error is to sometimes throw out a good event. Since good experimental practice requires taking samples of rejected events there is a continous operational check on the health of the processor. Note that an error rate of 1 in 1000 would be quite tolerable (if random) and error rates this high without complete failure are difficult to achieve in digital systems so that the deletion of error detection schemes, particularly those such as parity which slow down computation, is justified.

Again the nature of HEP is that it is event oriented. There is little chance of events occuring in an unexpected sequence and those that do can be serviced slowly by the host processor. Thus, there is no necessity for interrupts, eliminating a decision process that both complicates and slows down every instruction in conventional computers.

The announced approach to software was to design a processor for which it was not possible to write a complicated program. Thus the announced design did not include any kind of jump instruction or index registers. This kept thinking along lines of how to solve a specific problem (i.e., E-400 trigger) rather than solving general problems of computation. Later features were added where they were easy or where they would solve problems elsewhere - i.e., index registers and jumps allow doing the track finding in software rather than the original plan of having hardware present all possible track pairs to the processor for selection.

It was recognized that this is a one of a kind digital processor with no real expectation of additional copies. Since it promises at least an order of magnitude improvement in data rate in an experiment that costs on the order of $5 million to equip, (if we include the cost of building the accelerator and consider the total number of experiments which might be run), it is believed that this effort is justified. Still the one of a kind nature and thus the high ratio of burdened labor cost to material cost

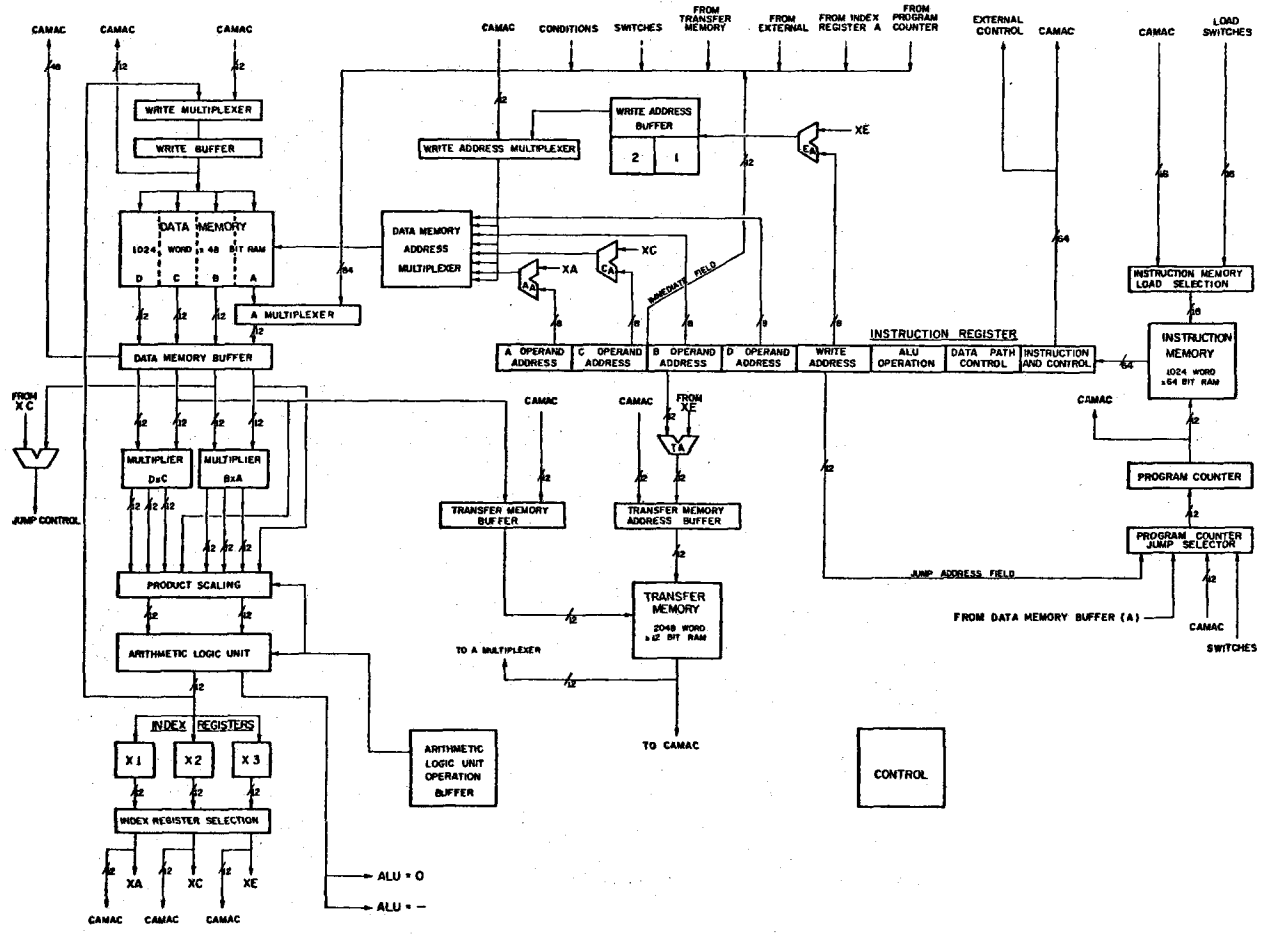
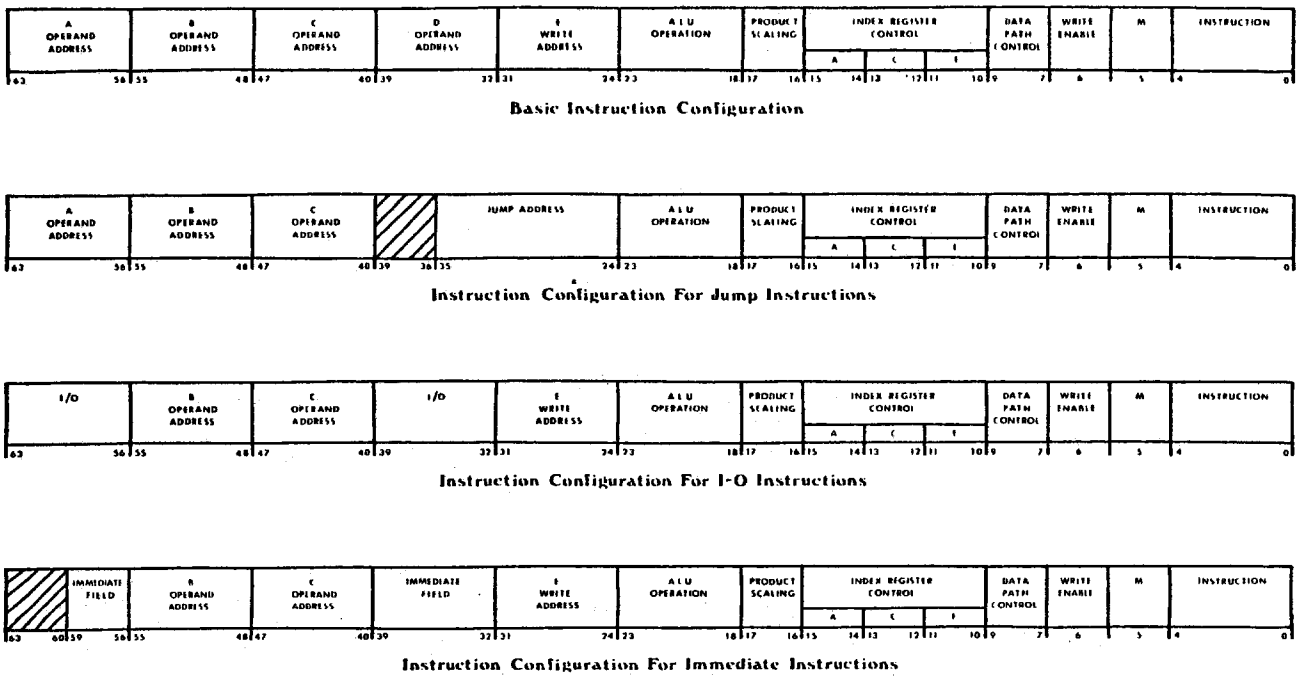


Fig. 4. M7 Processor Block Diagram



Basic Instruction Configuration

Instruction Configuration For Jump Instructions

Instruction Configuration For I-O Instructions

Instruction Configuration For Immediate Instructions

Fig. 5. Instruction Microcode Format

(at least 10/1) considerably affected the design approach. Indicators were put on all important registers, and all were interfaced to CAMAC. In general hardware was added wherever it would reduce checkout and test time and the tempation to be clever was resisted at the expense of more parts if the result was easier to understand and thus check out.

The design is very conservative for the parts used. By using extra parts and by keeping well back from the state of the art, long check out and debugging time was avoided. Again to avoid problems the package design is very conservative with respect to power distribution, cooling, and packaging density.

The M7 Processor

The M7 Processor is a 12 bit parallel machine (Figure 4) constructed from 10,000 series ECL. Major components are a 1024 word by 12 bit 4 port Data Memory, two 12 bit parallel multipliers, an arithmetic Logic Unit, three hardware Index Registers, a 1024 word by 64 bit microprogram Instruction Memory and a 2048 word by 12 bit Transfer Memory. The memory read access time for all three memories is 29ns, and the settling time for the 12 bit multipliers is typically 55ns.

The 64 bit microcoded instruction contains 5 eight bit address fields (Figure 5). Other fields include 6 bits for control of the ALU, 7 bits for control of various data paths, 5 bits for the instruction, and 6 bits to specify index registers for use with two of the source addresses and the destination.

Each instruction can access up to four words simultaneously from memory so that two multiplys with an ALU operation on the two products can be started every major clock cycle (approximately every 100ns). Instruction fetch and execute cycles are completely overlapped so that the primary limit on speed is the settling time for the 12 bit ECL multipliers.

The four port memory is implemented by the brute force technique of having four separate memories, one for each operand address. The memories are addressed separately for the read operation but are always addressed in common during write operations.

Since the processor program for this application is mostly computation with few decision operations no special effort is made on the jump instructions, or computations which require the results of the previous step -the processor simply stops and waits on these instructions. Since the primary program requires only one jump and one wait for a previous result this is not a problem.

The instruction cycle for the M7 varies from 4 to 8 micro cycles with the normal instruction (i.e., ax ± by → c) being 4 micro cycles long. Figure 6 shows the pipe line operation for three successive instructions. Data Memory write operations always take place during the first two micro cycles and read operations during the third and fourth micro cycle. If cycles 5 and 6 are used they are a memory write cycle.

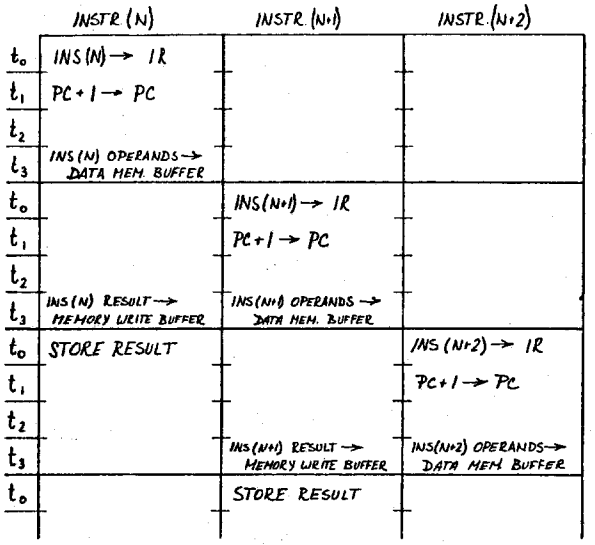


Fig. 6. Instruction Pipeline Timing

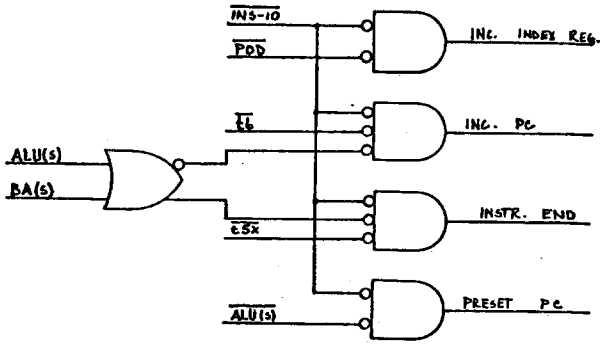


Fig. 7. Gates Required to Add List Search Instructions

At the leading edge of the first micro cycle, the Instruction Register (IR) is loaded with the instruction from the Instruction Memory (IM) which has been settling since t_1 time of the previous instruction. If the previous instruction is a jump instruction, then the previous instruction length is extended into t_4, t_5, t_6, t_7 time as necessary to allow sufficient IM access time. At the trailing edge of t_3 time, the operand addresses have settled and the Data Memory (DM) contents are loaded into the Data Memory Buffer.

The next instruction (n + 1) is now started by performing another instruction fetch. At the end of t_3 time of the n + 1 instruction, the output of the ALU is loaded into the Write Buffer (WB). Thus each instruction can use the entire instruction time for the multipliers and ALU to stabilize.

At t_{01} time of the n + 2 instruction the result is written into memory.

The relatively large microprogram word allows fairly complex instructions to be designed if fields are assigned different meanings than that required for the primary instruction, ax±by→c. One instruction, for example, was designed for efficient track searching. This instruction will search a list until an entry is found larger than one operand. If such an entry is found it exits with a jump, otherwise the next inetruction is taken when the list is exhausted. The simple register structure of the M7 allows this instruction to be implemented with only 5 gates (Figure 7).

Check Out

The M7 was checked out and placed into operation with minimum programming effort. About 4 total man months were spent in debugging the system. We consider this a modest effort and attribute the short time to several design factors:

1) Elimination of design complications which are difficult to debug and test- such as interrupt logic.

2) Very simple register structure, with most operations simple register to register transfers. The result is that a typical instruction requires only 4 gates, and even the list search instruction required only 5 gates.

3) Use of a relatively long microprogram instruction word.

4) Use of ECL 10,000 logic which is much quieter even though faster than TTL.

5) Use of an extensive CAMAC interface which allows all major registers to be examined and loaded from CAMAC.

6) Use of a Time-Sharing Computer CAMAC Controller.

The experimental configuration of the M7, Figure 1, will transfer data to the host computer (at first a Sigma 3 - later a PDP-11) through a CAMAC interface. While it would have been possible to acquire a CAMAC interfaced PDP-11 for check-out and test purposes, this was rejected for several reasons:

1) PDP-11 systems are a scarce resource at Fermilab and the system that would have been available to us was limited in operating features.

2) It would have been necessary to train a programmer and maintain programming skills. This is estimated to be at a half-time effort whether useful work is done or not.

3) A computer would require space, air-conditioning and maintenance effort- even when done by others, this is a considerable load.

4) A computer would require software and file structure maintenance to assure that test programs would be available

when needed.

The scheme adopted (Figure 8), was to build a CAMAC Crate Controller, the FBB, which is able to perform most CAMAC operations (except interrupts), by interpreting character strings received from a remote time-sharing computer. Operation of the FBB as well as its design is relatively simple. The RS232 cable between the terminal and its audio coupler is disconnected and plugged into the FBB. A cable from the FBB completes the circuit. Character strings generated in the central computer is a high level language (in this case, BASIC) perform all necessary CAMAC functions. One control character is used to suppress printing to save paper.

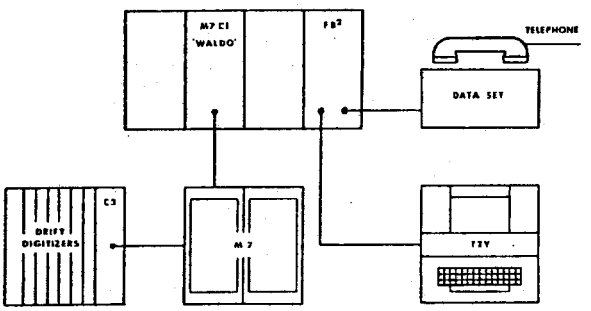


Fig. 8. M7 Check-Out Configuration

While the result is very slow, several CAMAC operations per second, it is still quite adequate for test and debugging and the editing features of BASIC allow very fast program modification. High speed tests are performed by downloading programs to the M7 which then tests itself. Note we consider this solution superior to a microprocessor crate controller in that while it is much slower, we have available to us the features of a large central facility with its extensive software and file maintenance.

With 10,000 series ECL a variable speed clock is a very useful debugging tool. One of the more common design errors is to forget to terminate a signal line. When this is the problem, operations will be performed correctly at clock speeds which allow about 1 microsecond between operations but not faster. In this case, if correct operation is achieved by slowing the system clock from 40 MHz to 1MHz it is a certain indication that a terminator is missing. Oscilloscope tracing in the indicated area typically revealed a "ski slope" signal.

Current Status

The M7 processor is now executing the program of Figure 3 in 2.1 microseconds. The present speed limitation is due to control logic, not arithmetic speed. The clock can presently be varied from zero to a 38MHz micro cycle rate. A variable speed clock is a convenient debugging feature and it is planned to limit operation to this speed until pressed. The M7 can be run at a clock speed of 53MHz by timing critical parts and operating at a fixed clock frequency. This corresponds to a program execution time of 1.5 microseconds at which time arithmetic errors begin to be made.

In order to evaluate the effectiveness of the M7 in the final experimental environment a Monte Carlo program is presently being run on the laboratory time-sharing computer. This program simulates an event and transmits the coordinates of the drift chamber hits over the telephone link to the M7. The time-sharing computer starts the M7 and then examines the result. In this way, it is possible to study how well the M7 will perform, though at seven orders of magnitude slower data rate. Figure 9 is the result of this simulation which was performed by the real M7 and which indicates that the M7 will improve the trigger efficiency by nearly two orders of magnitude.

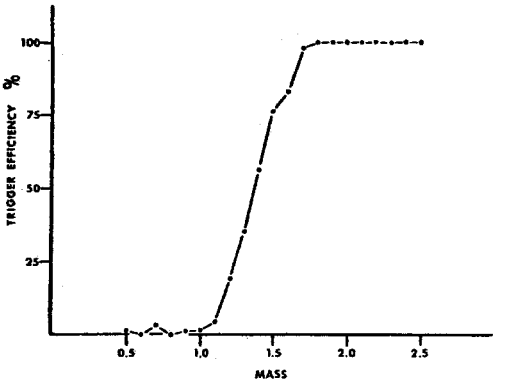


Fig. 9. Trigger Efficiency for 100Gev Dimuon Events after Processing by the M7 with a Mass Cut at 1.41 GeV/c^2

References

1. M. Atac, et al, Nucl. Instr. & Meth. 140 (1977) 461.

2. "CAMAC, A Modular Instrumentation System for Data Handling", USAEC TID5875, July, 1972.

3. "Future Data Bus Requirements for Laboratory High Speed Data Acquisition Systems", USERDA TID-27621, June 1977.

4. UNIBUS is a registered trademark of Digital Equipment Corporation.

5. T. F. Droege, et al, IEEE Trans. Nucl. Sci. NS-23, No. 1 (1976) 248

6. Motorola MC10183, Motorola, Inc., Phoenix, Arizona