FILE COPY

CERN-Data Handling Division
DD/78/25
November 1978
C. Verkerk

ON-LINE FILTERING

Lectures presented at the 1978 CERN School of Computing
Jadwisin, Poland, 28 May - 10 June 1978

ON-LINE FILTERING

C. Verkerk
CERN, Geneva, Switzerland

1. INTRODUCTION

Present day electronic detectors used in high-energy physics make it possible to obtain high event rates and it is likely that future experiments will face even higher data rates than at present. The complexity of the apparatus increases very rapidly with time and also the criteria for selecting desired events become more and more complex. So complex in fact that the fast trigger system cannot be designed to fully cope with it. The interesting events become thus contaminated with multitudes of uninteresting ones. To distinguish the "good" events from the often overwhelming background of other events one has to resort to computing techniques.

Normally this selection is made in the first part of the analysis of the events, analysis normally performed on a powerful scientific computer. This implies however that many uninteresting or background events have to be recorded during the experiment for subsequent analysis. A number of undesired consequences result:

- the dead-time losses for recording the uninteresting or "bad" events can be considerable.
- a large number of magnetic tapes has to be written and read, imposing a heavy burden of work and administration.
- a very considerable amount of costly computer time is used in relatively simple calculations needed to narrow down the event selection.

Each single one of these effects constitutes a sufficient reason for trying to perform the selection at an earlier stage, in fact ideally before the events are recorded on magnetic tape. We will call this early selection "on-line filtering" and it will be the topic of the present lectures.

We will first have a closer look at some electronic detectors and experimental techniques in order to clarify more precisely *why* we want to perform filtering of events on-line. We will then, after having defined *what* we want to do, give some insight in *how* it can be done. This latter part will be based on several examples of filtering, performed on running or planned experiments and will cover a wide range of implementations: from pure software to the fastest hardware techniques.

As these lectures were intended to be of a tutorial nature, the examples taken from present or future experiments have been chosen with the sole purpose of illustrating the material presented. The experimenters should therefore not be held responsible for possible misrepresentations, which are entirely to be attributed to the author.

2. GENERAL

2.1 A typical large experiment

2.1.1 Detectors

A better understanding of the need for on-line filtering can be obtained by examining more closely a typical experiment. We have chosen for this purpose an experiment[1] which is at present in preparation and which is typical in the sense that it makes use of many different types of detectors, that the selection criteria are based on the signals from several of those detectors and that the experimenters plan to make extensive use of on-line filtering techniques.

A schematic of the planned lay-out of this experiment is shown in Fig. 1, while Fig. 2 gives an enlarged view of the first metres of the equipment. One of the characteristics of large experiments is apparent from inspection of these figures: many types of detectors are used. Each has specific properties and is therefore useful for measuring specific quantities of interest in the analysis of the events. Wire chambers or drift chambers provide space coordinates along particle trajectories, allowing momentum measurements when a magnetic field is present. Several detectors of the same type are then used to provide the coordinates: in Fig. 2 five drift chambers and ten planes of proportional chambers are shown.

Čerenkov counters measure velocity and thus make mass-determination possible when combined with momentum measurements. Calorimeters measure the amount of energy deposited when a particle traverses matter and make further distinction possible between particles. They also provide a direct energy measurement in case of total absorption.

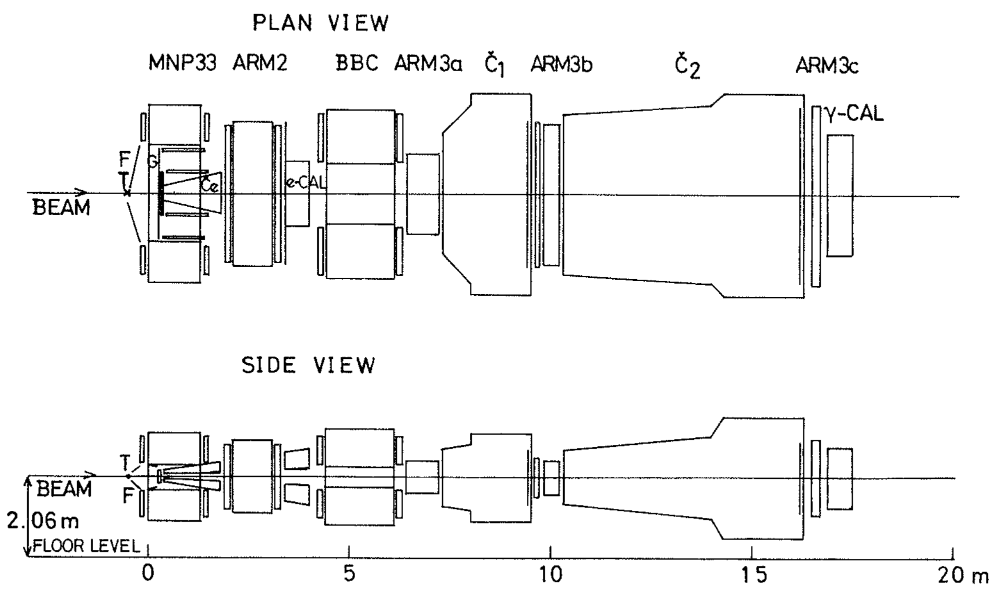


Fig. 1 Lay-out of an experiment to measure charmed particle production. Note the use of many different types of detectors.

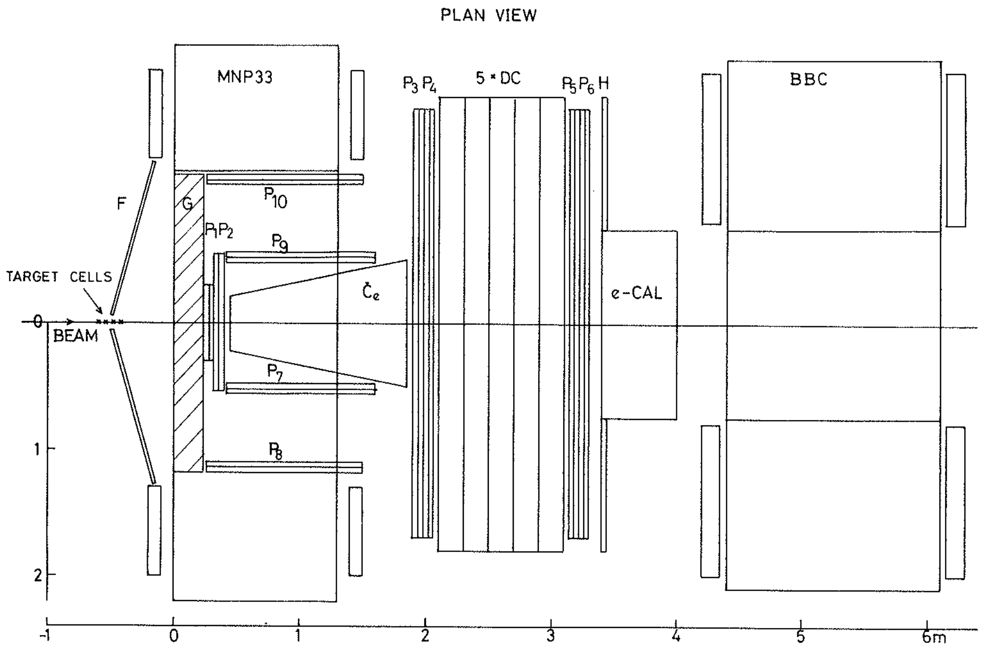


Fig. 2 Enlarged view of the first few metres of the set-up of Fig. 1.

2.1.2 Data volumes

The majority of these electronic detectors generate considerable amounts of data; some typical values are:

- Multiwire proportional chambers: 10,000 or more wires, spread over a dozen or more planes are typical for a present day experiment. Approximately 50 wire clusters will be hit in an event and must be read out and recorded.
- Drift chambers: 1000 wires are typical, with again approximately 50 hits in an event. Multiple hits on a single sense wire are possible. The coordinates are only known after a delay (≈ 1 μs), caused by the drift time of the electrons.
- Calorimeters: the scintillation counters which are sandwiched between plates of heavy material require of the order of 500 photomultipliers.

A typical event will activate ≈ 20 channels, where also the pulse height of the signal must be measured and digitized.

- Scintillation counters: they are the workhorses of every experiment, as the primary trigger is derived from their signals. Several tens of counters are typically found in an experiment.

Other detectors, like ionization detectors and Time Projection Chambers are being developed, which will provide still larger amounts of data.

From this brief reminder of some of the characteristics of detectors one can see that a typical event is described by 200-300 words (16-bit) of data, giving mainly coordinates and pulse heights, but also patterns of counters which are hit and general information (magnet currents for instance). Larger experiments can easily reach 1000-2000 words/event and future experiments will probably require even more information to describe the event.

2.1.3 Limitation of the trigger selectivity

The trigger electronics in its first and very fast decision based on signals from scintillation and Čerenkov counters cannot take all the additional information into account and its selectivity is therefore limited. Several reasons make it difficult to improve the selectivity. To distinguish events from all the other things (noise) happening in the detectors, the time correlation of the particles is essential. The resolution time of the coincidence circuits is generally in the few nanosecond region, so the electronics must be very fast. The modules used accommodate only a few simple logic functions with a small number of inputs per module. For large numbers of counters and complicated trigger requirements the number of modules needed rises above all acceptable limits. The number of NIM crates (containing the fast electronics; at 20 modules/crate) averages about 30 for the experiments at present on the floor at CERN. The larger experiments use up to 70 NIM crates.

The counters used in the trigger are usually rather large and this has as a consequence that the selectivity on kinematical quantities is reduced. Another limitation on the fast trigger is that the precise data from wire and drift chambers cannot be used, either because it represents far too many input signals or the data are not available in time.

2.1.4 Types of background

The fast trigger, although essential for detecting time-relations and for providing the start signal for further actions, will therefore leave the experimenter with events highly contaminated by background. This background can be of different nature: firstly purely random occurrences of signals can lead to a trigger without a real event being present. Secondly there are the events which genuinely satisfy the trigger requirements, but which will turn out to be of an undesirable type when further and more sophisticated selection criteria are applied. Another class is formed by those events which, although they belong to the reaction being studied, are nevertheless in an uninteresting kinematical region. This could for instance be the case when elastic scattering at high momentum transfer is being studied. A simple trigger could detect the presence of a pion in one and of a proton in the other arm of a spectrometer (using Čerenkov counters to make this distinction). As the differential cross-section decreases exponentially with $|t|$ the events with $|t| \gtrsim 5$ GeV will then be entirely drowned, if no adequate selection on the scattering angle is made at the same time.

2.2 Definition of on-line filtering

The primary purpose of the analysis is of course to eliminate the background, applying progressively more restrictive selection criteria. The above example of the elastic scattering shows however that a large pay-off will be obtained if part of the more stringent selection criteria could be applied in real-time, e.g. before the event is written onto tape and recorded for off-line analysis. We will call this process on-line filtering. It is important to realize that this definition of on-line filtering implies eliminating events (hopefully the large majority of triggers) *before* they are recorded. These events are therefore lost without any possibility of recovery.

On-line filtering should also be clearly distinguished from certain other treatments the data could undergo in real-time. Monitoring of an experiment by building up histograms and plots of rates, particle distributions, etc. is essentially different. So is sample analysis, a more sophisticated form of monitoring. Often the data are transformed on a word-by-word basis: reformatting, packing or clustering (of wire hits) are examples. In general these processes have to remember one or more of the previous words of data, but they can be performed for a single detector, independently of the signals from the other detectors. A filter on the contrary works on the ensemble of the data from more than one detector (but not necessarily on *all* the data from *all* detectors).

On-line filtering introduces the concept of a multi-level decision process performed in real-time. Two levels are normal: the fast trigger followed by the slower filter. One can however very well think of several levels of filtering, each level working on the events accepted by the previous level.

The filter is obviously always slower than the fast trigger, but in most cases it must work under severe time constraints, to avoid excessive dead-time losses or the need for running at a lower trigger rate.

2.3 Example of filtering in a planned experiment

To illustrate some of the above we return to the experiment shown in Figs. 1 and 2. The purpose of this experiment, at present in preparation, is the observation of charmed particles, characterized by the production of a prompt electron. The presence of the electron is ascertained by the Čerenkov counter and by the energy deposited in the calorimeter. This energy must fall between the limits of 2 and 10 GeV. Electrons in this energy range are however produced in abundance by other well-known phenomena: decay of π^0 and η-mesons followed by conversion of the γ-rays. A careful selection of the events must be made to discriminate against these and other sources of background. In the proposal for the experiment[1], the physicists made a careful analysis of these sources of background, which we will not reproduce in all detail. They pı pose to eliminate the undesired events in different steps. The primary trigger requires an electron depositing > 2 GeV in the calorimeter (class 0). Additional criteria for the rejection of events with a pair of electrons (instead of a single one), are also applied in the fast trigger logic. Also for this purpose, extra counters have been added catching the second electron when it goes off in uncommon directions. Table 1 shows the rate of events (per 10^4 interactions) in which the second electron undergoes the fate given in the first column. The complete hardware trigger reduces this background by a factor 4, compared to the class 0 trigger.

Table 1

Rejection of electron pairs in the example experiment (slightly compressed from table in Ref. 1). The rates are normalized to 10^4 interactions

Fate of second electron	Class 0	Hardware trigger	Software trigger (to tape)	After off-line rejection
Trigger calorimeter	10.84	0.00		
H-counters	36.20	0.00		
Misses both but picked up in P4	8.00	8.00	0.78	0.06
Misses all three but picked up in P7 or P9	4.36	4.36	0.20	0.01
Emerges downstream but not detected	0.13	0.13	0.13	0.13
Hits sides of MNP-33 and picked up in P7/8 or P9/10	4.26	4.26	0.31	0.00
G-counters	1.41	0.00		
Hits MNP pole faces and picked up in P1/2 or P1/2/8	2.09	2.09	0.01	0.00
F-counters (reflected electrons)	6.42	0.00		
Other lost electrons	0.66	0.66	0.66	0.66
Total	74.4	19.50	2.09	0.86

An on-line filter, using microprogrammable processors (ESOP, see below) will further reduce the rates of these background events to the numbers given in the fourth column (the final rates after off-line analysis are given in the fifth column). The microprocessors will work on data from different detectors and apply different criteria to improve the selection. A block diagram is shown in Fig. 3, which is in fact part of the over-all data acquisition system. Different tasks are distributed to a number of processors. One of the processors can make a finer analysis of the pulse heights produced in the calorimeters, together with the position information provided by the drift chambers and thus ensure that not

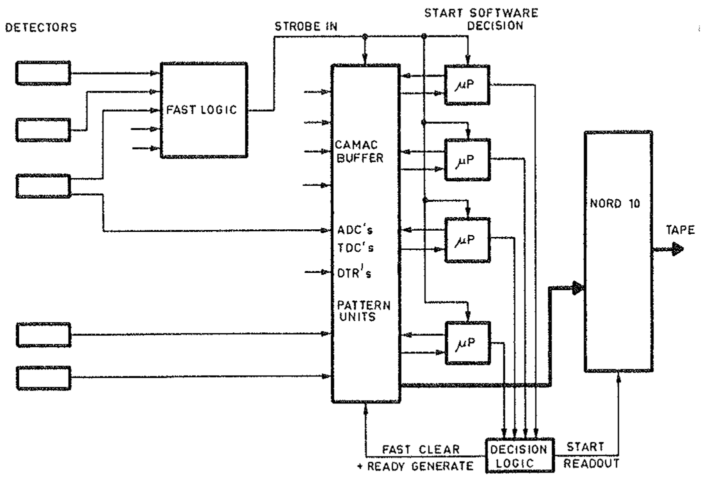


Fig. 3 Basic block diagram for 2-stage decision (hardware trigger followed by on-line filter) in the experiment shown in Figs. 1 and 2. In principle more than one microprocessor is foreseen, each working on a subtask.

more than one electron was produced. Another can trace back particles found in wire planes P4-P10 in different combinations (see Table 1) and determine their momentum. A further aid in suppressing the background comes from the pronounced difference in p_T distribution of electrons from π^0 decays and those from charmed particle decays (Fig. 4).

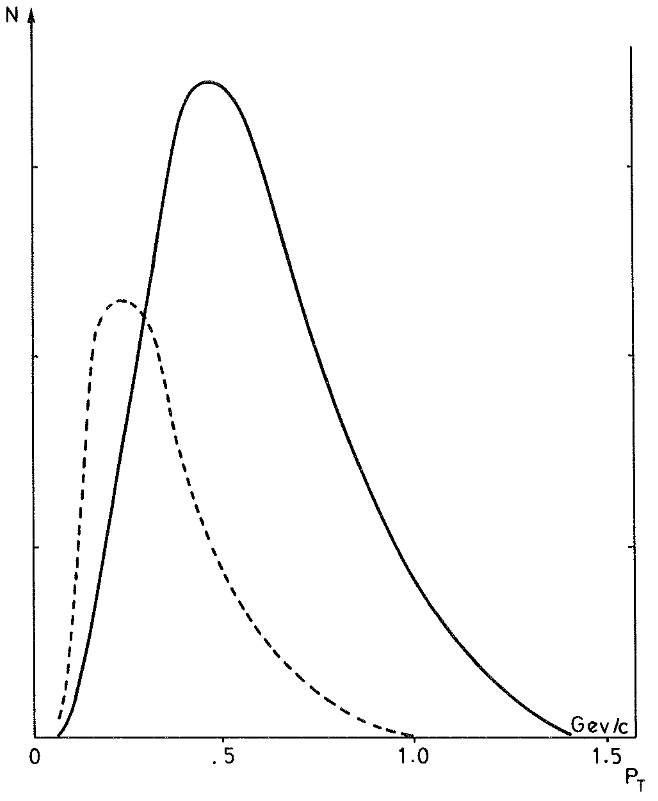


Fig. 4 Distribution of transverse momentum for electrons from charmed particle decays (full curve) and from $\pi^0 \to \gamma\gamma \to 4e$ decays (dotted curve). This is an example of a selection criterion applicable only after p_T has been calculated (in an on-line filter).

From Table 1 one sees that this on-line filter, executed before the events are written to tape is expected to reduce the background by an order of magnitude; another factor 2-3 is gained in the off-line analysis.

Other sources of background have also been considered and also here, as Table 2 shows, the on-line filter is expected to reduce the rate by considerable factors (2-10). The numbers in Table 2 are based on an estimated rate of 1.4×10^5 interactions/burst. It is important to note that a hardware trigger rate of ≈ 500 triggers per burst (duration ≈ 1 second) is expected, which means that the filter is constrained to a time of less than 1 millisecond, if dead-time losses are to be kept reasonable.

Table 2

Estimated rates for different sources of background, based on 1.4×10^5 interactions/burst (from Ref. 1)

	Triggers/burst		
Background source	Hardware	On-line filter	Off-line
1) π^0, η	280	29	4.1
2) $K^\pm \to e^\pm \pi^0 \nu$	4	2	2
3) Prompt $e^\pm$	2	1.5	1.5
4) $\pi^\pm$	41	8	0
5) $\pi^\pm \pi^0$	180	18	0
	507	58	7.6

2.4 Trigger rate and filter speed

We will examine a little more precisely the relationship between the filter speed, the trigger rate and the purity of the sample obtained by the first level trigger. We assume that we have N triggers per second, containing ρN "good" events, e.g. events which are still considered acceptable after the filter process and retained for off-line analysis. We further assume that the filter introduces a dead-time after each trigger and that the dead-time τ_b following a "bad" event is different from the dead-time τ_g following a "good" event. The time τ_r needed for a complete read-out of the event data is included in τ_g, while τ_b includes the time for partial read-out, limited to the data needed for the filter. Without the filter a dead-time of τ_r follows each trigger and the observed rate of unfiltered events is therefore:

$$U = \frac{N}{1 + N\tau_r}\,, \qquad (1)$$

containing

$$U_g = \frac{\rho N}{1 + N\tau_r} \qquad (2)$$

good events.

With the filter in place the observed rates of bad and good events will be R_b and R_g, respectively. Fractions $R_b\tau_b$ and $R_g\tau_g$ of time are then lost following the triggers on bad and good events, respectively, so that:

$$R_g = (1 - R_g\tau_g - R_b\tau_b)\rho N \qquad (3)$$

and

$$R_b = (1 - R_g\tau_g - R_b\tau_b)(1 - \rho)N\,. \qquad (4)$$

From (3) and (4) follows:

$$R_g = \frac{\rho N}{1 + (1 - \rho)N\tau_b + \rho N\tau_g} . \qquad (5)$$

The on-line filter will be particularly attractive if it would allow us to record more good events on tape in a given time interval than would be possible without filter. In other words: can we have situations where

$$R_g > U_g?$$

From (2) and (5) it is immediately seen that the condition for $R_g > U_g$ is (supposing $\tau_r = \tau_g$):

$$\alpha - \alpha\rho + \rho < 1 , \qquad (6)$$

where $\alpha = \tau_b/\tau_g$. As $\rho \leq 1$ by definition, α must be less than one, for (6) to be satisfied. The necessary condition for $R_g > U_g$ is:

$$\alpha = \frac{\tau_b}{\tau_g} < 1 . \qquad (7)$$

The gain in the rate at which good events can be recorded is then:

$$G = \frac{R_g}{U_g} = \frac{1 + N\tau_r}{1 + (1 - \rho)N\tau_g\alpha + \rho N\tau_g} . \qquad (8)$$

Maximal values of G are given in Table 3 for $\tau_g = \tau_r$.

Table 3

Gain factors in recording good events on tape

$$G = \frac{1 + N\tau_r}{1 + (1-\rho)\alpha N\tau_g + \rho N\tau_g}$$

The table entries are for $\tau_g = \tau_r$. $\alpha = \tau_b/\tau_g$.

α	$\rho = 0.1$				$\rho = 0.01$			
	$N\tau_r$ =				$N\tau_r$ =			
	0.1	1	10	∞	0.1	1	10	∞
0.9	1.01	1.05	1.09	1.10	1.01	1.05	1.10	1.11
0.5	1.04	1.29	1.69	1.82	1.05	1.33	1.82	1.98
0.1	1.08	1.68	3.79	5.26	1.09	1.80	5.26	9.17
0.05	1.08	1.75	4.49	6.90	1.09	1.89	6.90	16.8

One sees that for reasonable values of ρ (≈ 0.1) it is in fact possible to increase the rate of recording good events, but a considerable gain can be obtained only when the dead time following a bad event is small and when $N\tau_r$ is large.

We can also interpret the result in a slightly different way. If one accepts a certain level of dead-time losses, then an on-line filter will allow running at a higher trigger rate, without increasing these dead-time losses. Without a filter the trigger rate for a dead-time loss of a factor $1/(1 + k)$ is $N\tau_r = k$. With the filter in place the same losses will result for a trigger rate N′ when

$$(1 - \rho)N'\tau_b + \rho N'\tau_g = k . \qquad (9)$$

Supposing again that $\tau_g = \tau_r$ we find

$$\frac{N'}{N} = \frac{1}{(1 - \rho)\alpha + \rho} . \qquad (10)$$

The trigger rate can thus be increased by the values given in Table 3, without affecting dead-time losses.

2.5 Other limiting factors affecting event rates

Until now we have tacitly assumed that the read-out time of an event was the limiting factor in obtaining high recording rates. Two other factors may limit this rate and thus have an influence on the desirability of filtering: tape writing speed and the size of the available buffer memory. A distinction must be made between continuous machines (the ISR for example) and pulsed machines (e.g. SPS).

Supposing that we use normal 1600 bpi, 75 ips tape, the highest rate at which we can write 16-bit words is approximately 50K words/s. Assuming that a good event at the ISR requires 500 words, we cannot hope to record more than 100 events per second. The read-out time will be close to 1 ms and if we assume that a bad event could be rejected in 100 μs, we find from (5) that the trigger rate would be limited to $\approx 1200\ s^{-1}$ if $\rho = 0,1$. The dead-time losses would then be $\approx 23\%$. We are therefore limited in rate, but on the other hand, since we are recording continuously, we do not need more buffer capacity than is required for two physical records on tape.

For a pulsed accelerator like the SPS one can make use of the interval between pulses to write tape, but the events have to be stored in a buffer during the beam bursts. We will assume that events at the SPS are larger than at the ISR and consists of 1000 words of data. If we also assume that they can be read at a typical DMA speed of 1 word/μs, we could read out an absolute maximum of 1000 events per 1 second burst and write at most 300 events on tape during a full 6 second cycle. In principle, we could buffer the raw events and apply the filter before the events are written to tape. We would then need a very large buffer: 1M word. On the other hand the time one can spend on a good event would be of the order of 10 ms and one could think of times of the order of 1 ms for rejecting bad ones. Tape writing would not constitute a real limit for $\rho \lesssim 0.3$.

When we buffer only the good events, τ_b has again to be small compared to τ_r. Assuming $\tau_b = 50\ \mu s$ and $\tau_r = \tau_g = 1$ ms, we find that for $\rho = 0{,}1$ we can run at a trigger rate of $\approx 5400\ s^{-1}$, with $\approx 37\%$ dead-time losses. The buffer needed would be 300K words. These figures represent severe conditions for an on-line filter, but one can conclude that in most cases the tape-writing is not the limiting factor, but rather the performance of the filter itself and/or the buffer size. This is the more true for 6250 bpi tapes.

2.6 Requirements for filter algorithms

We have seen that for an on-line filter to be effective it is very important that it is capable of rejecting undesired events in the shortest possible time. Obviously one also wants to spend a minimum amount of time on the good events, although this is of lesser importance, particularly for small values of ρ. We have also seen that the decision making requires in general numerical calculations to be made on part of the data of an event.

These calculations can in general be performed with a rather low precision, since we are not trying to obtain accurate results. We only want to calculate the value(s) of one or more parameters, so that a decision may be taken. In making the decision we will always be prudent and avoid throwing away good events. Safety margins are therefore large and the accuracy attained in the calculation of the parameter is of secondary importance. In practically all cases the calculations can therefore be performed using integers of 16 (or less) bits.

The speed requirement on the contrary may impose severe constraints on the filter algorithms. Each case has to be considered on its merits, but generally speaking it is imperative to avoid time consuming procedures such as least squares fits and iterative approximations. One has also to beware of combinatorial problems, where the execution time $T = \alpha n^k$ increases with a large power ($k \geq 3$) of the number n of data points (the amount of data is roughly proportional to the number of particles, and thus increases with energy). Unfortunately this dependency on a power of the number of particles cannot always be avoided entirely. It is therefore worth while to examine the problem carefully, and to try using short cuts, with the aim of reducing at least the proportionality factor α. For instance, one should avoid continuing the execution of nested loops, when the application of a simple test would indicate that no further solutions can be found.

Another point to take into consideration is the importance to be given to the treatment of particular cases. Each particular case will complicate the algorithm, which might be undesirable. It is particularly awkward if a hardware implementation is envisaged. It need not necessarily lead to a loss in speed, although it often does add some extra time. The disadvantages of a proper treatment of the special cases must therefore be carefully balanced against the expected gain. A point-finding algorithm, to determine space points in wire chambers with four wire planes may serve as an example. The algorithm, which has been described *in extenso* elsewhere[2,3], is basically very simple and consists of executing nested loops over four sets of wire coordinates and checking if two conditions

$$|X + Y - 2U| < \varepsilon \tag{11a}$$

and

$$|Y - X - 2V| < \varepsilon \tag{11b}$$

are fulfilled simultaneously. In a time proportional to n^3 (n is the average number of hits per wire plane) the space points are found where the traversing particle generated a signal in all four wire planes. To find also those particles which gave a signal on three planes only (because of inefficiency of the fourth) the algorithm has to be extended, adding considerable complication[2,3]. Two more scans through all the coordinates are required, so the execution speed is affected by a factor 3. This loss in speed and the added complication must be balanced against the gain in efficiency of finding space points. For an average efficiency of a wire plane of $1 - \eta$ ($\eta << 1$), a fraction 4η of points cannot be found by the basic algorithm, which requires a signal on all four planes. Considering that the space points of several chambers are often needed in further stages of the filter process, the fraction of events lost can be rather large. For instance, if the chambers are 99% efficient and points in 4 chambers are needed for event reconstruction, one would lose around 16% of the events, if the simple basic algorithm would be used.

Sometimes the algorithm can be simplified or its speed improved by an appropriate design of the detectors. Examples exist of wire chambers designed to avoid the need for multiplications in the point-finding algorithm. This has for instance been done by choosing a wire spacing in the planes with inclined wires (by ±45°) a factor $\sqrt{2}$ different from the spacing in the planes with wires at 0° and 90° [2,4]. This is the reason why in (11a) and (11b) the factor 2 appears, instead of $\sqrt{2}$. In any implementation of the algorithm (except on the largest scientific computers) the factor $\sqrt{2}$ would have been much more awkward.

One can also make judicious choices of the wire directions. A simple example is given in the experiment described before[1]. (For another example, see Section 5.3.1.) From Fig. 5 it is clear that

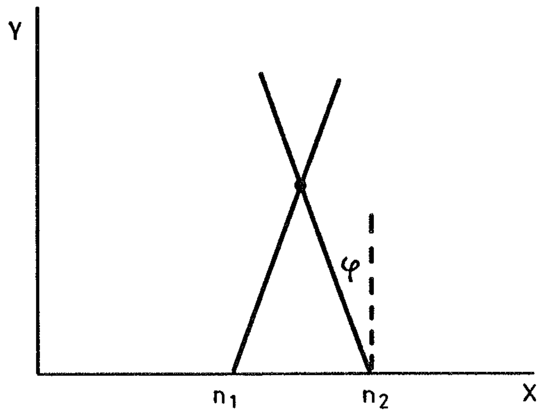


Fig. 5 The coordinates x,y of a particle can be determined from the wire hits n_1 and n_2 in two planes with inclined wires. A judicious choice of ϕ can speed up the calculations (see text).

when wire n_1 is hit in the first plane and n_2 in the second, the x and y coordinates of the particle are given by:

$$x = n_1 + y \text{ tg } \phi$$

$$y = n_2 - y \text{ tg } \phi \ ,$$

from which follows:

$$x = \frac{n_1 + n_2}{2} \qquad (12a)$$

$$y = \frac{n_2 - n_1}{2} \times \frac{1}{\text{tg } \phi} \ . \qquad (12b)$$

ϕ can be chosen so that $\text{tg}\phi = \frac{1}{2}$ or $= \frac{1}{4}$, $\frac{1}{8}$, and (12a) and (12b) become very simple.

Straight-line finding can be simplified -- by eliminating multiplications -- when the distances between detectors can be chosen judiciously.

2.7 Over-all system design

Before we will look at different implementations of on-line filters, we must emphasize one final point of general importance: the need for a good design of the whole system. When not enough attention is given to this point, the result might be deceptive! Speed is generally the dominant requirement and one should make a careful analysis of the over-all problem to see where the bottlenecks really are and try to avoid them. The pay-off will be considerable if the total data flow is examined and not just a particular part of it. Speeding up a subroutine by a factor two or so can have a large effect, but it can equally well result in a gain of a few percent only. It depends of course on how central this subroutine is to the problem. It is relatively easy to be misled here and to form the wrong idea. Ideally one should make measurements, but this is practically impossible during the design phase of an experiment.

The importance of eliminating the bad events as quickly as possible has been shown in the preceding paragraphs. A corollary of this which merits attention is that one should not give all events a treatment which only the good ones deserve. Complete read-out and tape-writing have already been given as obvious examples of this. There are others like reformatting of the data or reconstruction of all space points or all tracks.

There are many other things that can influence the over-all speed and a good system design would include some or all of the following features:

- fast read-out system (of the order of 100 ns per word);
- selective read-out, which makes it possible to read out only those detectors which are needed for the filter calculations or to read them out before the others);
- parallel read-out. This can be very effective if one can give a meaningful treatment to the partial data;
- hardware clustering of wire hits. This can be done on the fly by hardware, while it is slow in software;
- wire numbers should be counted from zero for each plane, or from a meaningful constant. This is again easy to organize in hardware;
- inside a program one should have a good data structure, which avoids complicated address calculations (either user coded or compiler generated);

- a simple data flow inside the program, with a minimum number of intermediate stops for the data. Communication of data between parts of programs is an important factor, particularly if several programmers work on the parts. Transfer of all data from an output buffer area to an input buffer located elsewhere in memory is not the fastest way of assuring this communication;
- efficient programming.

It is very unlikely that this list is exhaustive!

3. SOFTWARE FILTERS

3.1 Software versus hardware methods

Until a year or so ago an on-line filter implemented in software was necessarily residing in the experimental minicomputer. Software methods were then superior in the flexibility they offered, but were lagging far behind in speed, compared to hardware methods. Technological progress, e.g. the introduction of bit-slice microprocessors, apparently tends to bring the two methods closer together. The bit-slices are building blocks from which programmable processors can be built, leaving a large amount of freedom to the designer as to the structure and the functions of the processor. At first sight they therefore offer the best of both worlds, but at closer inspection also here speed and flexibility are closely related. The maximum speed will be attained when the bit-slice processor is closely integrated into the experiment, interfacing directly to the read-out of the different detectors. Most likely the structure will then also be adapted to the task at hand. The consequence is that the processor can only be programmed directly in microcode by a person who knows very well all hardware details. The processor is then still "programmable", but it can hardly be said to be "flexible".

The opposite solution that the bit-slices offer is to make the final product resemble as closely as possible a mini or other computer. This emulation could go as far as producing a machine capable of executing a program originally written for another machine[5,6]. This would offer full flexibility, as the programs could be written in high-level language if desired, but speed is largely lost. In fact the final product in this case is a minicomputer, comparable in speed if the same technology has been used. It would have the same sort of interface to the experiment, unless special equipment has been added.

As the choice between a hardware filter or a software filter is really one between speed and flexibility, the bit-slices have not solved the dilemma, but added another dimension to it. The reader himself will be able to appreciate this from the few examples of the use of bit-slice processors that are given below.

Making speed comparisons between soft and hardware filters in general terms is nearly impossible as it depends too strongly on the problem, the over-all system design and, for the software, the machine and the language used. Very broadly speaking, one can however say that, except for very simple problems, a dedicated minicomputer would not be able to filter more than 10-50 events per second. In most experiments the mini available is not entirely dedicated to filtering, but busy doing other things as well. One can estimate that on purely computational problems a bit-slice processor, not too different in structure from a mini, could attain about twice the mini speed, mainly due to the use of fast memory. As stated before, better speeds could be attained if the bit-slices were to be closely integrated into the experimental set-up; in this way however, the approach would be close to a hardware solution. The relatively low cost of a bit-slice micro is, however, an incentive to use more than one for filtering purposes, so that the desired speed increase might be found in multiprocessing. We will consider this aspect in more detail later. Even so, it will be difficult to reach the speed of pure hardwired processors. In a particular case (point-finding) a hardwired processor was found to process the data 80 times faster than a PDP 11/40 programmed in assembly language.

An additional remark about the so highly desirable flexibility is also in order. Flexibility, as we saw above, decreases when going from the minicomputer, via the bit-slice microprocessors to hardwired logic. Flexibility in this context is inversely proportional to the resistance of the system to changes. If the system is programmable, it is not yet necessarily flexible. The bit-slice microprocessor can be used in a way where it is not much more flexible than hardwired logic. The programmability can be very poor, when difficult microcodes must be produced. In addition some adaptations might not be possible without making changes to the hardware.

Although the hardwired processor is the loser in the flexibility race, it is an honourable loser. Real changes to the algorithm can only be made by modifying the hardware, but a good design will try to make a number of options already available to the user. Obviously all constants and parameters used by the algorithm must be loadable by the user from the data-acquisition mini. In addition it can often be arranged that certain parameters control the flow through the process and that a considerable adjustment can be made to changing requirements.

Generally speaking, the trade-off between speed and flexibility will then lead to the following choices:

- hardwired processor, when speed is paramount and the problem is known in all foreseeable detail well in advance, to allow for the time to design and construct the hardware;
- software, if speed is not important and if one expects frequent changes and adaptations to be made;
- the bit-slice microprocessor will be a good choice if one understands well enough the way it should be integrated into the environment but one cannot yet precisely predict the filter algorithm it should execute. The total effort required (design, construction, check-out, development of software tools like cross-assembler and production of user software) is comparable to the effort for producing a hardwired processor.

3.2 Example of a software filter: SFM

The Split Field Magnet facility at CERN has recently been augmented by the addition of several peripheral detectors, which play an important rôle for triggering on two sorts of events[7]:

i) events with an electron produced around $\theta = 90°$ and with transverse momentum $p_T \gtrsim 500$ MeV/c.

ii) Events with a hadron at $\approx 45°$ to accumulate high statistics for $p_T > 5$ GeV/c.

A typical event in the present set-up is shown in Fig. 6, and a block diagram of the data acquisition system in Fig. 7. On the latter we see that a PDP 11/20, marked "filter machine" has been added specifically for running the on-line filter program. A slow trigger, based on the presence of hits in wire groups lying within certain predefined "roads", makes a first selection of those events where a track of one of the types defined above is present.

The hardware in front of the filter and the data acquisition computer (ROB and multiplexor) then make a selective read-out possible in one of the two branches. Thus the filter computer only receives data for which there is a good probability of finding tracks. The purpose of the filter is precisely to find those tracks and to transmit their parameters to the data acquisition computer. A track-finding algorithm has been developed[8] which uses linear parametrization to find candidates in projection and to match them in space. From the points found in the first (n - 1) planes, which are supposed to lie on a track, the predicted intersect with plane n is calculated from a linear form and required to fall within predefined limits:

$$c_{1n} < \left[Y_n - \sum_{i=1}^{n-1} \alpha_{in} Y_i \right] < c_{2n} \tag{13}$$

Y_i and Y_n are measured points (not necessarily in the same projection); α_{in} are the coefficients determined beforehand by tracking high momentum tracks through the magnetic field. The field is however very inhomogeneous and the detector volume must be divided into 20 regions, each with its own set of α_{in}. In addition, to predict point n the set α_{in} has elements differing from those of the set $\alpha_{i,n-1}$.

The track following starts from the outside of the detector going toward the centre. At the outside, tracks are generally well separated and ambiguities are avoided to a large extent. This method of track finding requires some storage space for the coefficients, but it is fast as it does not need geometrical calculations. Also the program flow is simple. The momentum of a track can be approximated at all stages using a linear form and another set of coefficients β_{in}. The algorithm was developed for off-line analysis and then recoded in assembly language for the on-line filter. Earlier this year the data taking rate was limited to 11 events/second (due to tape-writing), but the filter saturated only at 90 events per second. The software filter spends an average of 4.7 ms per event, including the selective read-out and some reformatting of the data.

During test runs with the high p_T trigger, 10% of the events were found to have a track with $p_T >$ > 3 GeV/c. With the electron trigger, the trigger rate was 30 s^{-1} and it is expected that the filter will reduce this by a factor 2-3 so that a good match with tape-writing may be obtained.

3.3 Speeding up software

Is it possible to speed up software, without losing its inherent flexibility? The answer is of course affirmative and different methods exist, of

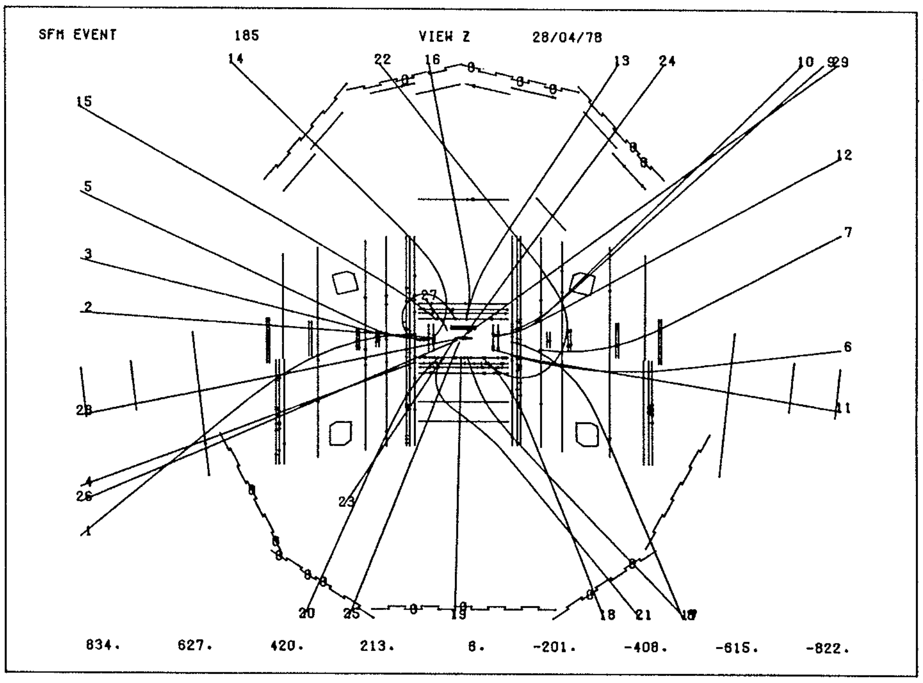


Fig. 6 A typical event in the present set-up of the SFM facility. Different detectors are schematically indicated.

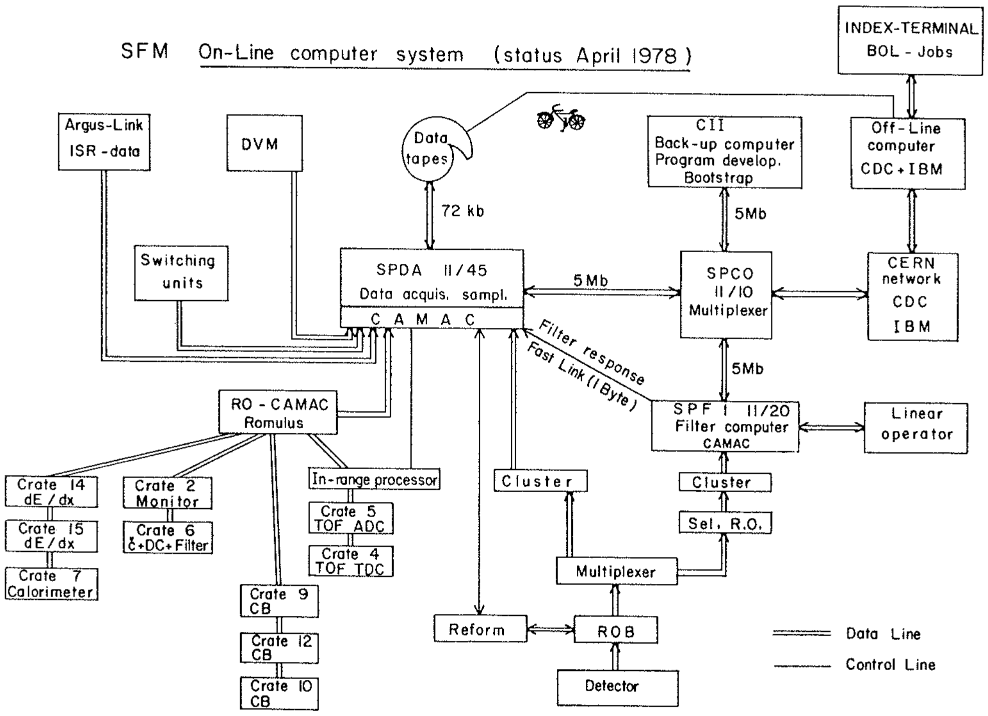


Fig. 7 On-line computer system of SFM. Two PDP-11's are used; the on-line filter computer is the PDP-11/20 at the right.

which some have been applied. The first is very obvious and should not need to be mentioned.

3.3.1 Efficient programming

Efficient means resulting in the fastest execution. This often conflicts with other requirements like readability, maintainability, etc., of the program, but the emphasis should be placed where it needs to be. Programming the on-line filter -- or at least critical parts of it -- in assembler language is worth the effort as can be seen from the filter for the SFM.

3.3.2 Microprogramming

Some commercial minicomputers are user-microprogrammable. The Hewlett-Packard HP2100 is an example, although it is not expected that users will make large use of this facility. No particular effort has therefore been made towards making it "user-friendly". It is also considerably more difficult to microprogram this particular machine than it is to write an assembler program for it. Nevertheless, a factor of 2 or more may be gained by avoiding the storing of intermediate results, or by making use of the possibility to do certain elementary operations simultaneously with others. The gain is largest for repetitive operations requiring one or two registers for intermediate results (multiplication algorithm, for example). Microprograms have been written for critical parts of the vertex finding in a proton scattering radiography experiment[9].

3.3.3 Add hardwired functional units

This is an old and well-known technique: to the basic computer which 25 years ago had only the capability of adding two numbers, fixed-point multiply/divide and floating-point units have been added. The idea can be pushed further and one could add other hardwired functional units, like matrix multiply, inner product or polynomial units. Such a unit can be very fast[10] and a mismatch between processing speed and I/O rate for such a special unit is not exceptional.

An example of a special unit which has been built to speed up a software filter is the "linear operator"[11], connected to the SFM filter computer and also shown in Fig. 7. A block diagram of this unit is shown in Fig. 8. The unit is connected to the PDP-11 Unibus and it calculates the linear form (13), including the comparison, if required. During initialization all the different sets of coefficients are stored in the coefficient store, which has 70 ns

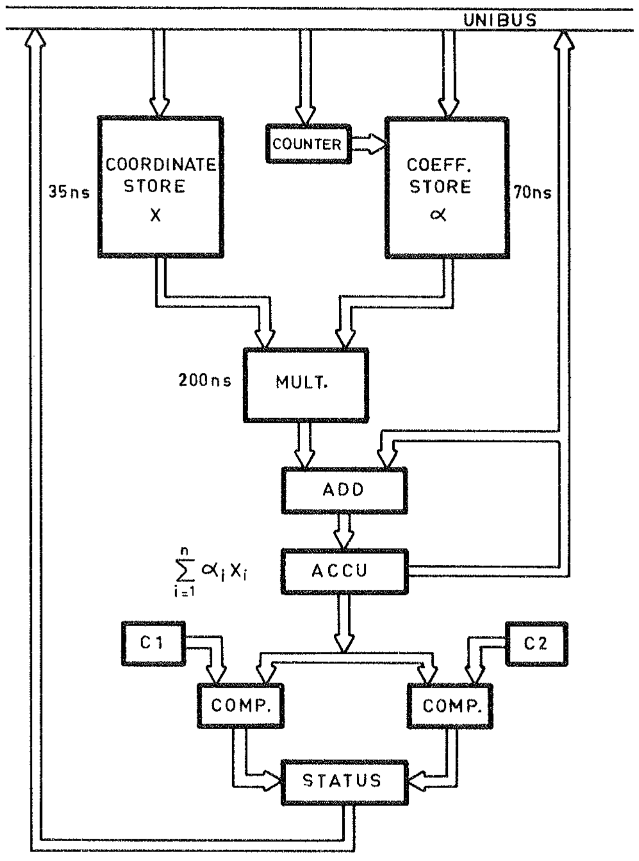


Fig. 8 Block diagram of the "linear operator", used as an additional functional unit for the filter computer in the SFM.

access time. The coordinates are stored in a separate memory, under control of the PDP-11. The process is then started by loading a pointer to the coefficient table. The result $\Sigma\alpha_{in}x_i$ can be read back by the filter computer which then decides on the next step. Alternatively c_{1n} and c_{2n} are also retrieved from the coefficient store and the comparison made. If the outcome is positive the PDP-11 has only to load the next coordinate (in plane n + 1) following the others (from plane 1 to n) in the memory and then provide a new pointer. The linear operator is very fast: one term is added to the sum in 300-350 ns. Nevertheless, the over-all gain in the SFM filter was small when the unit was used to calculate the sum, passing the result back. Less than 20% was gained on the 4.7 ms. This is due to the fact that only about half of this time is spent in the filter algorithm and that only a fraction of the latter is spent on calculating $\Sigma\alpha_{in}x_i$. In addition, in the mode described, the processing time of the unit is small compared to I/O, which takes place at PDP-11 speed. Using the other mode of operation will give an additional speed-up.

3.3.4 Multiprocessing

To speed up filter software multiprocessing should be considered. Generally speaking, two different ways may be envisaged:

- parallel processing on several events at a time. This is conceptually easy, a new event is given to a free processor for treatment. There are problems however if one wants to apply this to an on-line filter: buffer storage must be provided for the data of all events in the course of being processed. If this is not provided one cannot gain over a monoprocessor. Unfortunately the entire data must be stored and selective read-out is therefore of little or no use;
- distributed processing in several processors, each working on a part of the task for a single event. This is very attractive in principle, but not easy in practice. Dividing the task into (nearly) independent subtasks might not be easy, but the problem can be solved for certain cases. Even so, the effort will not be negligible, communication (of data, results and status) can be complex and so is the problem of synchronization. The system design will often not be easy. It is very likely however that in the coming years we will see examples of some sort of distributed processing applied to online filtering.

An example of multiprocessing is given by experiment WA10 at present running at the SPS. It is a parallel scheme where the two subtasks are distributed in time. Five to eight processors are used (see Fig. 9) in the experiment which studies the reaction

$$K^{\pm}p \rightarrow K^0_S \pi^{\pm} p \;. \qquad K^0_S \hookrightarrow \pi^+\pi^-$$

and similar topologies.

There is no magnetic field and by measuring the directions of all tracks and the energy of the recoil proton (by time of flight) a 2-constraint fit can be made. The processors perform the complete kinematical analysis before writing events to tape. The processors DPNC 811 are home made and slightly simplified versions[12] of the PDP-11, based on the MMI 6701 bit-slice and the Intel 3001 sequencer.

The system works as follows: during the 1 s burst events are read and stored in the DPNC 811's.

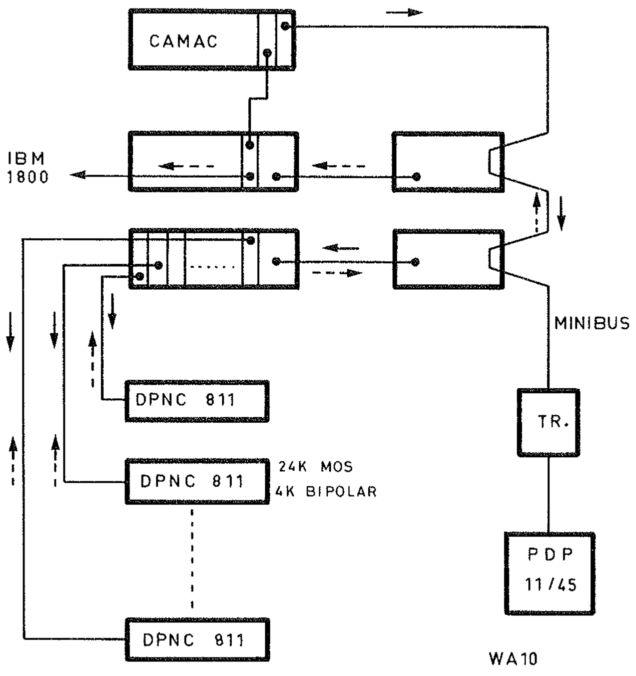


Fig. 9 Block diagram of the data acquisition system for experiment WA10. A number of small computers (DPNC 811) perform the on-line filtering between accelerator bursts. They serve also as buffer memory for the events.

This is done under control of the PDP-11/45, which directs each new event to a free processor. The data do not transit through the PDP-11/45's memory however. Upon reception of an event the DPNC 811, executing a program residing in a 4K bipolar memory, compresses the data from approximately 200 words to 100 words (mainly by packing two data words into one memory location). The compressed event is stored in a 24K MOS memory. The system therefore acts also as a large buffer memory for the events. Between bursts the DPNC 811's perform the kinematical analysis and whenever a result is available the event is transferred, under control of the PDP-11/45 again, to the IBM 1800 which does the tape-writing. Each 811 can process 120 events between bursts and with five processors an event rate of 600/burst has been obtained[13]. It is intended to expand the system to double capacity.

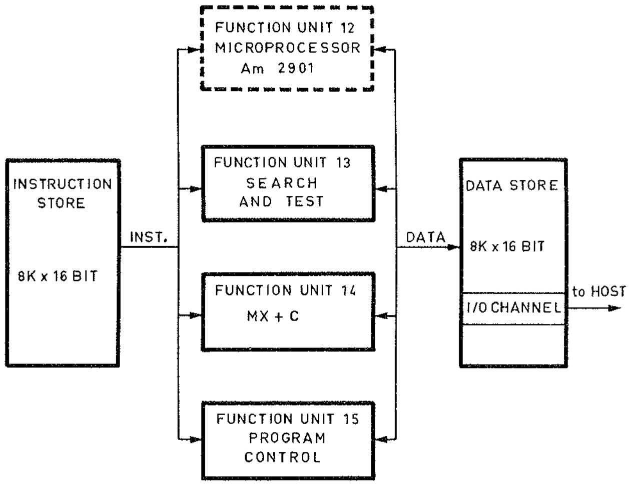
3.3.5 Build your own fast processor

The last possibility for speeding up a software filter is building a programmable but fast machine. From the hardware point of view this is an attractive solution as you can

- specialize the structure to suit the particular problem;
- add special functional units with a minimum overhead in data flow;

So there is the possibility of beating every existing machine in speed. Viewed from a software standpoint, enthusiasm risks being mitigated, since programming this machine will unavoidably be awkward. To make programming less tiresome a large effort will be needed in writing assemblers, loaders, linkers, etc., with the risk of also slowing down the machine considerably.

There are several examples of such machines implemented using MSI technology (and more if we include bit-slices; those will be treated further on). The oldest example is probably the processor originally called the "Formula Evaluator"[14] developed at the Rutherford Laboratory. This processor centres around an arithmetic unit specialized to calculate $y = mx + c$. In its present version (Fig. 10) other functional units have been added, including a general purpose unit based on the Amd 2901.



FUNCTION UNIT 12 MICROPROCESSOR Am 2901
INSTRUCTION STORE
8K x 16 BIT
INST.
FUNCTION UNIT 13 SEARCH AND TEST
FUNCTION UNIT 14 MX + C
FUNCTION UNIT 15 PROGRAM CONTROL
DATA
DATA STORE
8K x 16 BIT
I/O CHANNEL
to HOST

Fig. 10 Block diagram of the Rutherford Special Purpose Processor, consisting of different special function units.

The second example is a microprogrammable processor[15] built for a muon experiment (#400) at FNAL and called the M7. It is intended for on-line filtering and specialized to calculate

$$c \Leftarrow ax \pm by$$

very fast, but with limited precision (12 bit). It is implemented in ECL and uses two parallel multipliers, while simultaneous access can be made to the four operands. In 18 successive steps, making use of the elementary operation, the transverse momentum and the effective dimuon mass can be calculated:

$$p_T^2 = \frac{K^2[(A_1x_2 - B_1x_1)^2 + (A_2y_2 - B_2y_1)^2]}{(A_3y_2 - B_3y_1)^2} \qquad (14)$$

$$M^2 = \frac{K^2[(A_1\cdot\Delta x_2 - B_1\cdot\Delta x_1)^2 + (A_2\cdot\Delta y_2 - B_2\cdot\Delta y_1)^2]}{(A_3y_2 - B_3y_1)(A_3\bar{y}_2 - B_3\bar{y}_1)} \qquad (15)$$

K, A_i and B_i are coefficients, x_i, y_i, $\bar{x}_i$ and $\bar{y}_i$ are coordinates measured in drift chambers, and $\Delta x_i = \bar{x}_i - x_i$. The design aim was that the decision must be taken within 5 μs. The present implementation which uses a three-stage instruction pipeline executes the program to calculate (14) and (15) in 2.1 μs.

The last example is ESOP[16], originally developed at CERN to perform track filtering on the Erasme machines for measuring bubble chamber film. This microprogrammable processor is implemented in Schottky TTL and has a very short microcycle ($\leq$ 100 ns). Besides being used in Erasme, preparations are under way to use it in different electronic experiments. A new CPU board contains the necessary logic to make a "fuzzy compare", that is to check relations of the type

$$|a - b| < \varepsilon$$

and to branch accordingly. Some other special function units have been designed, e.g. a bit test and shift unit[17]. In addition the interfaces to CAMAC have been improved and a buffer memory system has been developed[18]. On the other hand, the program memory (separated from the data memory) is limited in size (256 words) and programming the device is not particularly easy.

4. BIT-SLICE MICROPROCESSORS

As we have seen before, the bit-slice microprocessors form a sort of transition from the software to the hardware filters. The bit-slices are a convenient way of packaging, in a single integrated circuit, arithmetic and logic units together with register files. They do not bring fundamentally new architectural ideas to the construction of processors, be they general or special purpose. It is however easier and faster to build a processor out of bit-slices than building it out of "discrete" elements[5], as was the case with the processors described in the foregoing section. They do also leave freedom to the designer who is sufficiently expert, so that he is not limited to making little variants to a manufacturer's standard design. Somewhat better speed performance can in general be obtained however, by using MSI elements.

When a bit-slice processor has been designed for easy programmability it is not fundamentally different from a minicomputer as was mentioned before. For use in a filter a few differences should be noted. In this application there is no need for sophisticated peripherals nor for the execution of large programs. The price can therefore be kept rather low and this may stimulate using more than one processor in an experiment as we have already seen.

When on the other hand the programmability is kept at the level of microcode, the bit-slice processor approaches very much the philosophy of the specialized processors described before. Much more liberty can then be taken in the architecture and the processor can also be very intimately integrated with the read-out or the control of the experiment. Clearly one can go too far in this direction: when the architecture becomes too specialized for the algorithm, no real difference exists then anymore with a hardwired processor. The development effort will be large, programming cumbersome and modifications of the algorithm difficult to implement. There is another pitfall: the temptation will be great to use bit-slice processors even in those cases where a hardwired processor would be the best solution from all points of view (parts cost, development, programming effort, interfaces). For example, it would be unwise to replace hardwired cluster-logic by a microprocessor.

The next few years will undoubtedly see several applications of bit-slice microprocessors to particle physics experiments. Large efforts will probably be dedicated to developing bit-slice based systems and a large amount of practical experience will be gained. At present already a number of examples can be given of their application.

4.1 168/E

The 168/E 19), developed at SLAC makes use of the Amd 2901 bit-slice RALU, but decoding and sequencing of micro-instruction relies largely on random logic. The present version5) executes a large subset of the IBM 370 series instructions, including single precision (32-bit) floating point instructions. It therefore emulates an IBM 370/168 and programs written for this large machine (in FORTRAN) can be run on the pocket version. One additional operation is required on a program which runs satisfactorily on the IBM 370/168 to prepare it for running on the 168/E: a translator transforms the sequence of 370 instructions into a sequence of 168/E micro-instructions. The microcode is executed directly by the 168/E, which does not recognize 370 code at all. The 168/E therefore does not need peripherals for program development. The powerful facilities of a large computer are used instead and only at the last stage is the program transformed and run on the 168/E. Programmability of the 168/E is therefore excellent and the processor is well adapted for number crunching. Present plans at SLAC seem to be more directed towards using the 168/E for the full analysis of experiments than in on-line applications. The 168/E has a program memory separate from data storage. It is capable of executing typical code at a speed which is only 1.5-2 times less than the speed of the large IBM 370/168. An important factor contributing to this good performance is the fact that the direct execution of microcode suppresses the (macro)-instruction fetch and decode times.

4.2 GESPRO

GESPRO is a processor developed at Strasbourg20) and originally intended to serve as an intelligent CAMAC controller. The data acquisition computer can specify which read-out sequence to execute and GESPRO would go through all the individual steps autonomously. The intention is also to use it for data compression in experiment WA2, which studies hyperons. It is based on the Intel 3000 series and is one of the first bit-slice processors developed in Europe for particle physics. Programs, which must be written in microcode, can be developed using a cross-assembler implemented on a NORD-10.

The processor developed by Guzik for the channelling experiment at FNAL is mentioned elsewhere in these proceedings5).

4.3 μ77

A very fast processor21), based on the Amd 2900 series has been developed at the Ecole Polytechnique for use at CERN in an experiment to measure with high precision the total cross-section for $\pi^- p$ 22). Small angle scattering is observed, in the region of interference with Coulomb scattering. A sketch of how the differential cross-section varies with the angle θ is shown in Fig. 11. Extrapolation to $\theta = 0°$ allows us to find the total cross-section (optical theorem). The events beyond the Coulomb scattering limit are retained and the others rejected. The processor is intended for this filtering problem, which is solved by using particle coordinates from 5 small wire chambers.

We are approaching more and more a specialized solution, as the read-out of these chambers is very

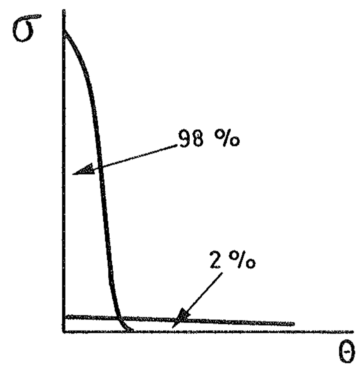


Fig. 11 The experiment to measure the total cross-section for π^-p (S157) collects high statistics on elastic events, which are a small fraction of the total number.

intimately related to the processor. A block diagram of the processor is shown in Fig. 12. The read-out and encoding of the wire planes is performed in a sort of "peripheral processor" (PP1 to PP5). The wire chambers are quite small (256 wires) and encoding is done by priority encoders in 65 ns (or 130 ns in case of double hits). The wire coordinates are stored in scratch pad memories of 16 words by

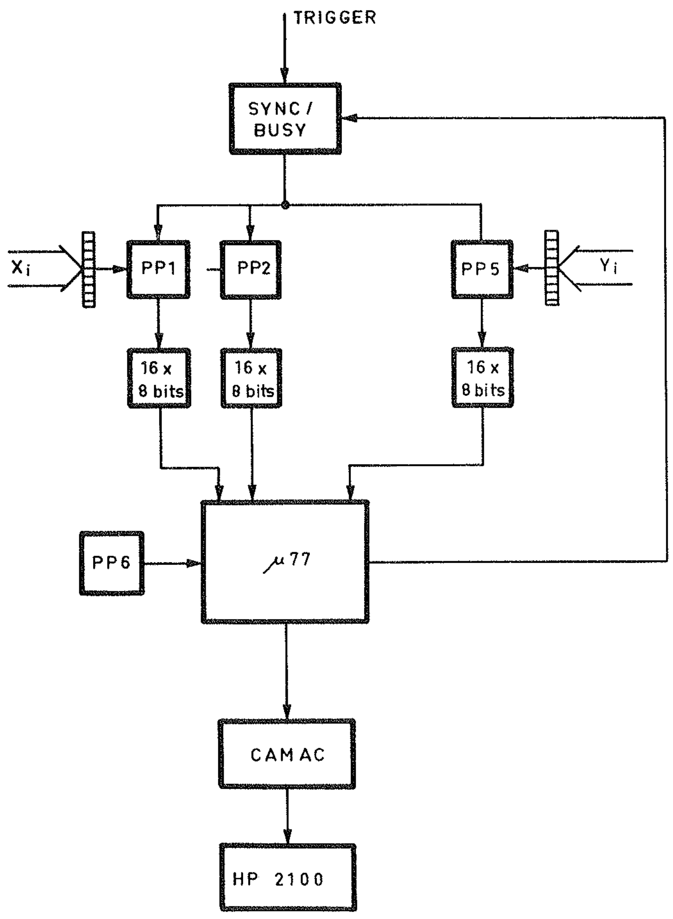


Fig. 12 Block diagram of μ77, used in experiment S157. The processor is directly interfaced to the experiment (PP1 to PP5).

8 bits, from where they are retrieved by the processor. The process is pipelined and the calculations on event n take place simultaneously with the encoding of event n + 1. At the same time event n + 2 can be latched. The processor executes a short cycle to select events on the basis of the scattering angle:

$$\Theta_x = \sum_1^3 \alpha_i n_i + b \quad \text{and similarly for } \Theta_y \qquad (16)$$

$$\Theta^2 = \Theta_x^2 + \Theta_y^2 \,. \qquad (17)$$

In addition the synchronization with the trigger and the loop control is done in this short cycle, which consists of 22 micro-instructions. Since a micro-instruction requires 168 ns for execution, the total time spent in the short cycle is 3.5 μs. This time has been obtained by performing the multiplication in an 8 × 8 bit array multiplier.

The accepted events (< 10% at this stage) can be submitted to further treatment. Here we see clearly the advantage of a programmable processor. For instance Δp/p can be calculated for the incoming beam particle

$$\frac{\Delta p}{p} = \sum \alpha_i' n_i + k \qquad (18)$$

so that a two-dimensional histogram (frequency of θ and of Δp) can be updated and maintained. Approximately 2% of the triggers contribute an event to the histogram. The processor is capable of treating $\approx 3 \times 10^5$ events/second and a two-dimensional histogram of 10^6 bins is filled in about 100 bursts at the PS (≈ 5 minutes). The update of the histogram is transmitted to the minicomputer, which keeps the master histogram and prints out results. Remarkably enough, no magnetic tape is used in the experiment.

Another interesting point is the fast rejection of bad events, while some extra time is being spent on the accepted ones to refine the selection. Finally one is left with a small fraction (< 1‰?) of pathological events, which cause a dilemma. Should they be treated further, at the cost of slowing down everything, or should they be rejected or accepted? A valid answer can of course only be obtained after careful study of these pathological events. All in all, this experiment is a very good example, not only concerning the use of bit-slice microprocessors, but also with respect to the application of general principles of on-line filtering.

5. HARDWARE FILTERS

In the domain of hardware filters it is useful to distinguish two speed classes:

- the processor type, with speeds in the few to 100 μs range;
- the "slow trigger" or second level decision type with a decision time of the order of one to a few microseconds.

5.1 Hardwired processors

The most salient advantage of a hardwired processor is the fact that for a given algorithm we can realize the fastest design possible, within the constraints of technology and budget. In fact one can fully exploit all tricks of the trade, which are mainly

- parallelism
- pipelining
- duplication of elements
- non-sequential control.

As is well known, there is a price to pay for achieving top performance. This price is expressed in long development time, for rather bulky and expensive equipment, which is difficult to maintain. It should therefore always be carefully considered if a hardwired processor is the right solution to the problem at hand. If it is, then interesting results can be obtained, as we will see from some of the examples.

5.2 Some history: global methods

A few years ago, global methods of pattern recognition seemed very attractive and particularly suited for implementation on hardwired processors. They presented in fact severe combinatorial problems and were therefore very time consuming on a general purpose computer. The idea was to recognize *all* tracks in an event, relying on methods like principal components[23] or nearest neighbours[24]. As these methods have been described elsewhere we will only recall the principle. All possible combinations of points detected in several detectors (taking one point per detector plane) are checked either against certain constraints (principal component method) or for their distance in multi-dimensional space to some standard set of track points (nearest neighbours). For n particles detected in m detector planes, n^m combinations must be checked in the absence of pre-selection criteria.

It turned out that these methods were easily implemented in hardware if one limited oneself to the basic algorithm[2,3]. As the time necessary was still considerable, the introduction of pre-selection criteria was desirable. Such a pre-selection could be made on the assumption that tracks in a magnetic field are approximately straight lines in one projection. Checking for straight lines can be done in a time proportional to n^3, which represented an enormous gain over n^{10} or n^{12}. This introduced however considerable difficulties. Very awkward complications are introduced by the fact that one has to take into account many types of special cases: for example, a missing point in a detector upsets the principal component method entirely. The result is that a hardwired processor for performing pattern recognition by such a global method becomes very complex.

A still more fundamental criticism can be made. The methods are aimed at finding all tracks. This is very good if one wants to save computer time in the analysis phase, but it is not necessarily a requirement for on-line filtering and it certainly conflicts with the principle of rejecting bad events quickly. In addition a Monte Carlo study must be made first and the results of the Monte Carlo provide the parameters used in the calculations. This does not contribute to making the methods acceptable for the physicists.

5.3 Hardwired filter processors

A number of processors have been built with the aim of providing fast event rejection. Examples are the processor for the Mark II spectrometer at SPEAR[25] (described by P. Kunz in a seminar at this school), and the filter processors for experiments WA7 and WA18 at CERN.

5.3.1 Coplanarity processor

Experiment WA7 [26] studies two-body reactions at large p_T, where the cross-section is very small. The trigger is based on a coplanarity test, performed by coincidence matrices receiving signals from scintillator hodoscopes. As the hodoscope cells are rather large ($\approx 20 \times 20$ cm^2) the trigger is not very selective. A filter processor[27] is used for a second level decision, made with the full precision of the wire chambers. The processor can be used in master or in slave mode. As a slave it only transmits its decision to the data acquisition computer, as a master it also resets the read-out system in case of a rejected event. The processor receives the wire chamber data selectively (the essential wire planes only) at a rate of $\approx$ 120 ns per coordinate. It consists of three separate processors. The first one (pre-processor) is in charge of preliminary tests on the number of hits per plane, the number of planes hit, etc. It is also in charge of communication with the read-out, the data-acquisition mini and the other two processors. Finally it performs the operations which can be done on a single coordinate, like adding or subtracting it to or from a constant in order to express all coordinates in a common reference system.

The second processor performs point-finding in three chambers sequentially. The wire chambers have U and V planes with wires at angles given by $\sin \alpha = \pm 8/17$ and $\cos \alpha = 15/17$ (see Fig. 13). The point-finder therefore checks the following relations:

$$|15X + 8Y - 17U| \le \varepsilon \qquad (19a)$$

$$|15X - 8Y - 17V| \le \varepsilon \,. \qquad (19b)$$

The required pre-multiplication is also done in the pre-processor.

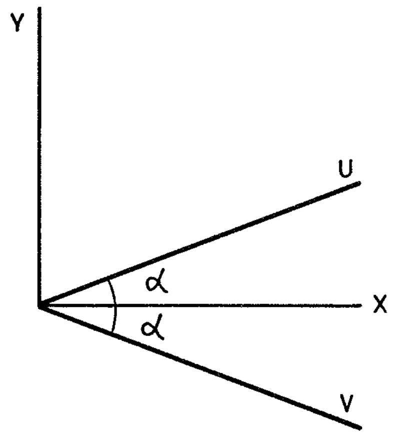


Fig. 13 Coordinate axes for a four-plane wire chamber (the wires are perpendicular to these axes). A correct choice of α leads to formulae (19a) and (19b) which can be evaluated without needing multiplications.

The last processor checks the coplanarity of two outgoing (straight) tracks with the incoming beam and checks the opening angle. The following formula are used:

$$COP = \Delta X_L \cdot \Delta Y_R - \Delta Y_L \cdot \Delta X_R + k_{COP}(\Delta Y_L - \Delta Y_R) + K_{COP} \qquad (20)$$

$$OPA = \Delta X_L \cdot \Delta X_R + \Delta Y_L \cdot \Delta Y_R + K_{OPA} \qquad (21)$$

where ΔX_L is the lateral displacement in the X-direction for the particle in the left arm of the spectrometer and so on. k_{COP}, K_{COP} and K_{OPA} are constants, which partly correct for the influence of the magnetic field between the target and the first detector. The coplanarity processor is the slowest. For an ideal, clean event the whole process would take no more than ≈ 20 μs, but when spurious hits or particles are present, the timing for the coplanarity test increases as n^4. The read-out by the mini-computer via CAMAC takes, however, 4 ms so the processor is fast enough except in extremely complicated cases. The processor is essential for the enrichment of the sample, as in test runs, without it, very few two-body events are found in the recorded events (≈ 1 in 10^4). The reader interested in knowing more about the techniques used to attain the fast execution is referred to more extensive descriptions of similar processors[2,3].

5.3.2 Histogramming processor

The experiment WA18 [28] proposes to study neutral current processes in ν and $\bar{\nu}$ interactions. A large part of the experimental set-up is occupied by a series of marble plates, sandwiched with counter hodoscopes. In each counter plane, perpendicular to the beam axis, there are 20 strips of scintillators. Planes with horizontal strips alternate with planes having vertical strips. There are 80 counter planes in total, i.e. 1600 scintillation counters.

About 300 primary triggers are expected during a 2 ms spill. In order to reduce the number of events a selection should be made between those events where an electron shower is generated and those where a hadron shower is present. A processor has been built[29] which should make this selection in a very short time. The processor makes a histogram of the number of hits per vertical strip, totalled over the 40 planes (see Fig. 14). Similarly, for the 40 planes with horizontal strips. For each

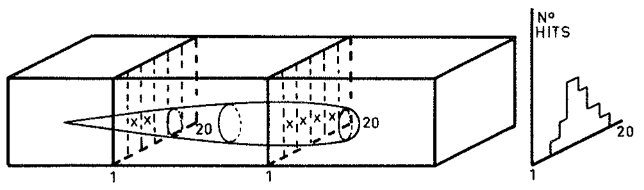


Fig. 14 A schematic view of a shower in the set-up of the neutrino experiment WA18. Only two of the 40 planes with vertical counter strips are shown and the number of strips shown is 6 instead of 20. All the counter hits are projected on the end plane and histogrammed.

histogram (containing 20 bins) the maximum content of a bin is found and a parameter calculated:

$$P_H = \frac{\text{total number of entries}}{\text{maximum}}$$

Similarly for P_V. The final parameter is

$$P = P_H + P_V \,.$$

Figure 15 gives an idea of the selection the parameter P allows to obtain. Above P = 5, 96% of the hadron showers are expected, contaminated with 1% of electron showers.

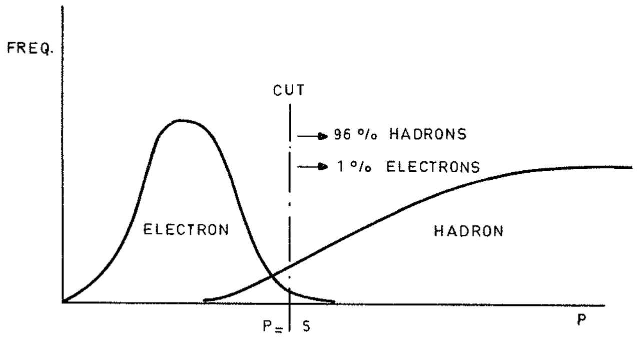


Fig. 15 Distribution of the parameter P for hadron and electron showers. A cut-off at P = 5 will select 96% of the hadron showers, contaminated by 1% of the electron showers.

A block diagram of the read-out and processing system is shown in Fig. 16. The 80 counter planes are read out by four fast systems identical to a

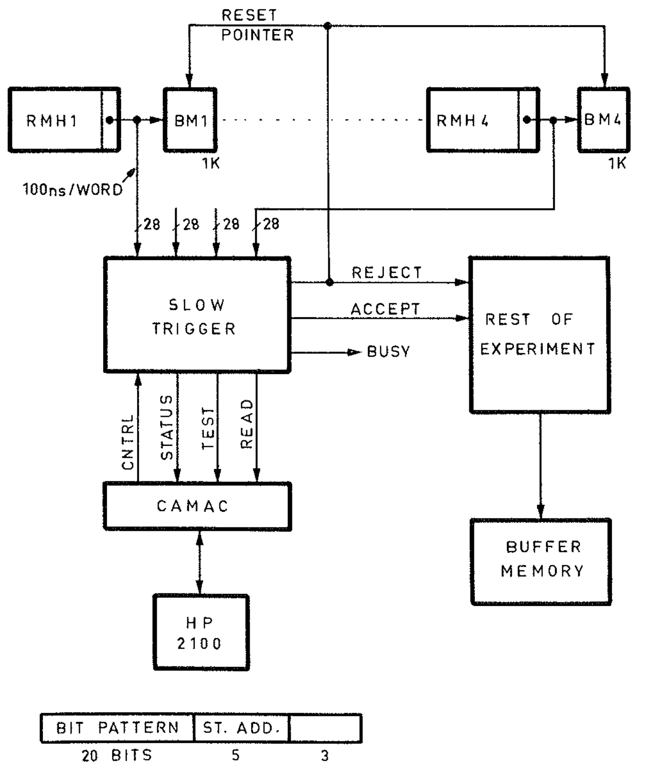


Fig. 16 Block diagram of the data acquisition system of experiment WA18. The hardwired processor is marked "slow trigger". Four fast read-out systems work in parallel, each into its private buffer memory. The "rest of the experiment" (ADC's, and data from additional detectors) does not take part in the on-line filter.

wire chamber read-out[30], operating in parallel. Each read-out writes into a buffer memory and simultaneously into the processor. Each word transmitted represents a 20-bit hit pattern of a plane and a 5-bit identification. When the event is rejected by the filter processor, the pointers to addresses in the buffer memories are reset to their previous values. The next event then overwrites the data of the rejected event. A block diagram of the processor itself is shown in Fig. 17. The four read-out

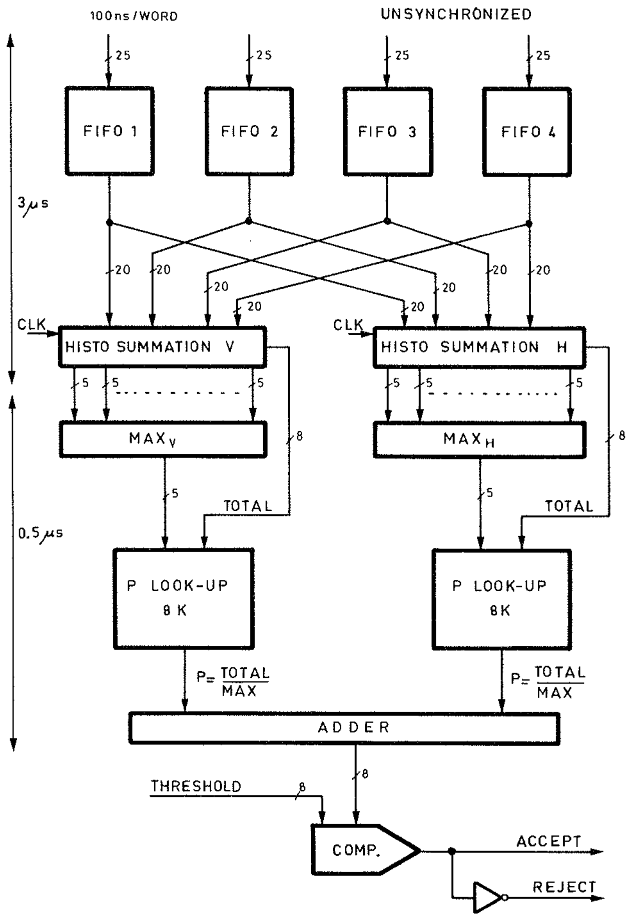


Fig. 17 Block diagram of the hardwired histogramming processor for experiment WA18. Note that the division necessary to determine P_H and P_V is done with look-up tables.

systems operate independently of each other and therefore FIFO memories are needed to obtain the necessary synchronization at the input of the processor. Depending on the identification number the data is routed to the horizontal or vertical histogramming unit. This summation unit adds 0, 1, 2, 3 or 4 to the contents of the different bins in the histograms. This is done simultaneously for all bins. The adders used need only 5-bit precision; the contents of a single bin is theoretically limited to 40 and in practice to 31. The maximum bin content is found and a table look-up scheme used to determine the values of P_H and P_V. This is possible since the total content is limited to 8 bits and the maximum content to 5 bits. There are therefore not more than $2^{8+5} = 2^{13} = 8192$ possible values of P_H. The histograms are filled on the fly during read-out of the data and this takes a maximum time of 3 μs. An additional 0.5 μs are needed to find P and to take the final decision.

5.4 Slow trigger processors

In the previous example a table look-up scheme was used to perform a rather complicated arithmetic operation (division) in a single memory access time, e.g. in approximatley 50 ns. This table look-up technique is not only valid to replace arithmetic operations whenever the precision of the operands is limited, but can also be fruitfully applied in decision logic. In fact the technique can be used to recognize as valid or invalid any bit pattern of limited length, whatever its origin. The bit pattern under examination is used as address to access a memory location in which the correct answer has been stored previously. In case a decision valid/invalid has to be taken, the memory need only be 1 bit wide. When an arithmetic result is desired the memory can be as many bits wide as desired. A memory can even be used as a many input universal trigger logic unit[31)], provided the trigger signals -- which together form a bit pattern -- are applied simultaneously and last for a time of the order of the access time of the memory. The memory has been loaded with a "1" for those input patterns on which one wants to trigger and with a "0" in all other locations. The bit patterns applied could, for instance, represent the rough shape of a track, the position of which has been determined in say 5 planes, with a 3-bit precision in each plane.

There is however a practical limit to the applicability of these methods. When the bit pattern exceeds 15 bits, the memory size becomes excessive (> 32K). There exist however, several methods to extend the possibilities, which are all based on the fact that in practical situations the vast majority of memory cells will be loaded with zeroes. For instance, in a particular case it was found that out of 32768 possible track shapes defined by 3 bits in each of 5 successive planes, less than 256 represented possible interesting tracks.

5.4.1 Extension of the input range

How can the input range be extended beyond 15 bits? Several possibilities exist:

i) store the valid codes (bit patterns) and compare with the one at hand making a binary search. This can be done by hardware, in $\log_2 M$ memory cycles, where M is the total number of codes stored. The time will in general not exceed 1-2 μs.

ii) The valid codes can be stored in a content addressable memory. A single cycle is sufficient to decide on the validity of the code at hand. Unfortunately, commercially available content addressable memory (CAM) chips have very small capacity (8×2 bits) and are expensive.

iii) The valid codes can be stored in a Programmable Logic Array (PLA). Again a single access cycle (50 ns) is sufficient to obtain the answer. At best, PLAs exist however in field programmable versions, programmed by blowing fuses at the appropriate places in the array. Re-programming therefore means using a new -- expensive -- chip.

iv) Cascaded access. The bit pattern is broken up in a number of fields, not necessarily of the same width, each containing a fraction of the total number of bits. The first field is used to address a memory containing pointers. If none of the valid codes contain the bit pattern in the field at hand, the pointer gives an immediate "no" answer. If the field can lead to a valid result, the pointer indicates the address of a further memory block. Individual words in this block can be accessed by a new field from the original code. The word accessed will again contain a pointer and so on, until all fields have been tested. The scheme is illustrated in Fig. 18. The advantage of this cascaded access can be seen from the following: changing from a 15- to an 18-bit pattern, the memory size must be increased from 32 K to 256 K, i.e. by a factor $8 = 2^3$ when direct access is used. If the result is obtained in three cascaded accesses, the total size of the memory need only increase by a factor two, since the extra 3 bits can be distributed over the 3 fields. Table 4 gives some examples for bit patterns of different lengths and for access in one, two and three stages. It also indicates how the bit pattern should be broken up in fields.

v) Hash-coding. This method is well known from software for data retrieval problems and can be directly applied to the recognition of bit patterns by hardware[32)]. The only problem resides in conflict resolution when the hashing algorithm gives the same result for two different input patterns.

The table look-up technique is very promising and in fact already applied in several instances: the "Programmable Track Selector"[33)] is based on it and one experiment at CERN uses fast 1 K ECL memories in its trigger logic[31)]. Also the hardwired processor for the Mark II detector at SPEAR[25)] uses table look-up in at least two places. Look-up tables can

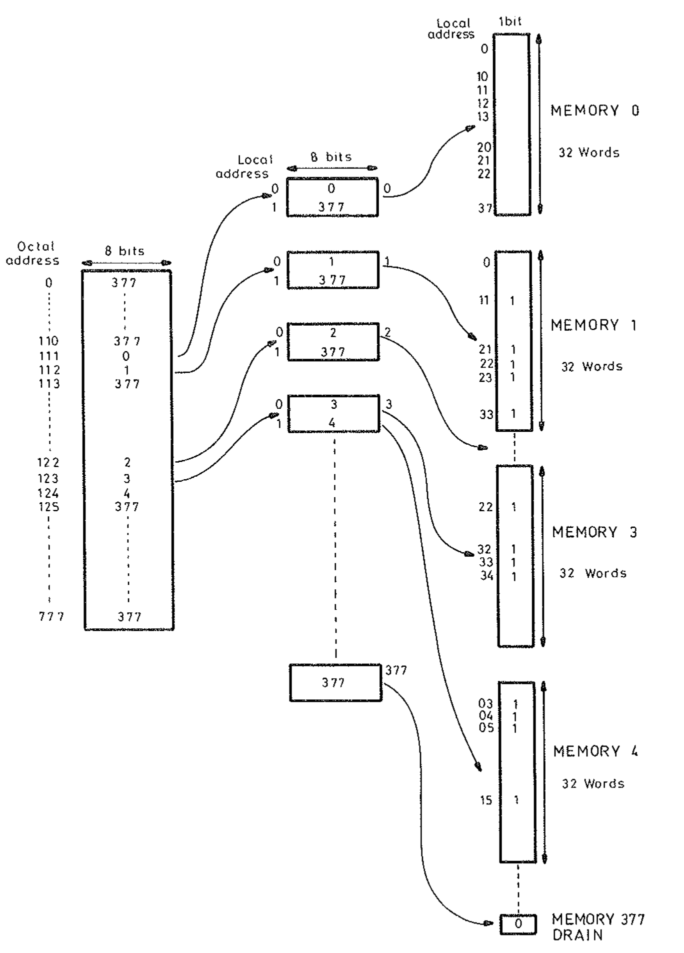


Fig. 18 Cascaded access scheme. Only when the first field(s) of the pattern is recognized as valid will the search continue. The pointer then indicates where the check for validity of the next field is to be made. Otherwise the pointer will indicate the "waste bin".

Table 4

Minimal memory sizes for cascaded access. The table entries give the total number of bits necessary to obtain a yes/no answer on a pattern of the indicated width. The field lengths are those which minimize this number of bits. It is assumed that there are not more than 256 valid input patterns (1 K = 1024)

Width of input pattern	Direct access	2-stage access		3-stage access	
	Number of bits	Number of bits	Field lengths	Number of bits	Field lengths
15	32 K	16 K	10;5	16 K	9;1;5 10;1;4 9;2;4
18	256 K	48 K	12;6 11;7	32 K	10;2;6 11;2;5 10;3;5
21	2048 K	128 K	13;8	64 K	11;3;7 12;3;6 11;4;6
24	16384 K	384 K	15;9 14;10	128 K	12;4;8 13;4;7 12;5;7

be useful to replace complete formulae and not just a single arithmetic operation.

This application has been suggested for straight line finding[34]. The memory size can be drastically reduced if the expression to be evaluated is linear. This is illustrated in Fig. 19 for the case of two

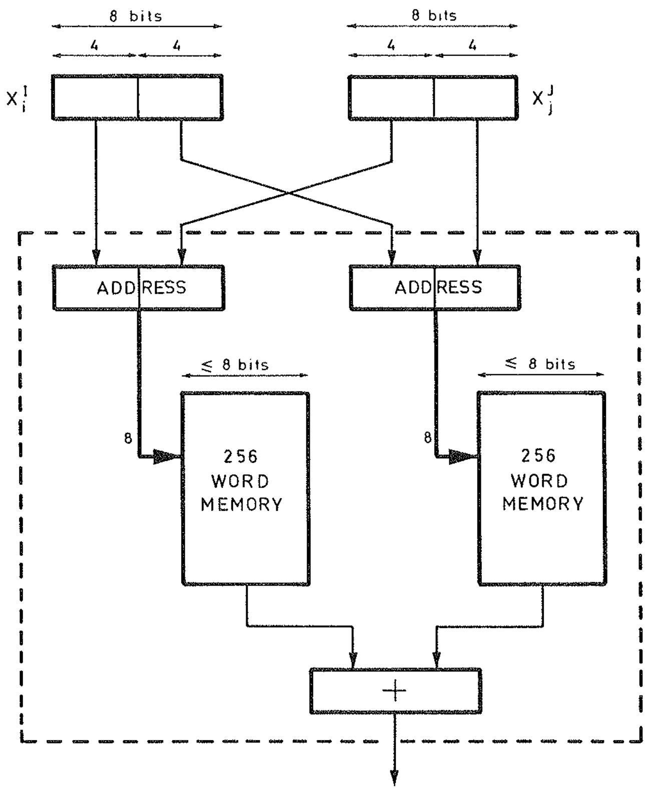


Fig. 19 For the evaluation of linear expressions the look-up table can be made very much shorter than the lengths of the operands suggest. Two 8-bit operands would require *a priori* a look-up table of length $2^{16} = 65536$.

independent variables. Straightforward application of the table look-up method to two 8-bit variables would require a 64 K memory (1 K = 1024). The variables can however be broken up in a most and a least significant part of 4-bit each. The expression is evaluated using the most significant parts of the variables to form an 8-bit address and similarly for the least significant parts. Since the expression is linear, the final result is the sum of the two partial results. The memory size has been reduced from 64 K to 2 × 256 words! Memories can also be used to build coincidence matrices with two, three or possibly more sets of inputs[35,36]. They can also be used to encode simultaneously more than one bit in an input pattern, in contrast to a priority encoder which will give only the position of the most significant bit.

However, several of these applications would lead us astray from our subject, on-line filtering.

6. CONCLUSION

These lectures attempted to give an overview of the area of on-line filtering in high-energy physics experiments. A large range of applications has been shown as examples. They covered a wide ground, from pure software to very fast table look-up methods, showing that considerable progress has been made over the last few years. On-line filtering is becoming widely accepted now and its application to large scale experiments is more and more unavoidable.

Hopefully these lecture notes may help the reader to find the optimal solution if he is confronted with an on-line filter problem.

7. ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to the numerous colleagues who contributed to these lectures by providing me with valuable and new information. The credit for the ideas and work in most of the examples is theirs. Amongst them I particularly wish to thank Drs. E. Barrelet, F. Gagliardi, M. Letheren, G. Lütjens and A. Vriens.

References

1) G. Blanar et al. (Amsterdam-Bristol-CERN-Cracow-Munich-Oxford-Rutherford Collaboration), Proposal to measure charmed particle production in hadronic reactions, Internal document CERN/SPSC/78-14.

2) C. Verkerk, Special Purpose Processors, Proc. 1974 CERN School of Computing, CERN 74-23, p. 223-262.

3) C. Verkerk, Hardware processors for pattern recognition tasks in experiments with wire chambers, Proc. Internat. Meeting on Prop. and Drift Chambers, Dubna 1975, p. 232-251; also CERN DD/75/18.

4) L. Baksay et al., Multiwire proportional chamber spectrometer for the CERN Intersecting Storage Rings, Nuclear Instrum. Methods 133 (1976) 219.

5) P.F. Kunz, Software for microcircuit systems, *in* Proc. 1978 CERN School of Computing. See also:
P.F. Kunz, Microcircuits for high-energy physics, *ibid.*

6) T.M. McWilliams, S.H. Fuller and W.H. Sherwood, Using LSI processor bit-slices to build a PDP-11-A case study in microcomputer design, Proc. AFIPS, National Computer Conf., 1977, p. 243-253.

7) P. Burlaud et al. (Annecy-CERN-Collège de France-Dortmund-Heidelberg-Karlsruhe-Warsaw Collaboration), Status report on experiment R416, Internal document CERN/ISRC 78-11.

8) A. Norton and F. Gagliardi, private communication.

9) Y. Perrin, private communication.

10) Examples of commercially available equipment are the AP-120B or AP-190L array processors of Floating Point Systems, Inc., and MAP of CSP Inc.

11) F. Gagliardi, J. Joosten and J. Hendel, private communication.

12) V. Hungerbühler, B. Mauron, J.-P. Vittet, An economical and fast minicomputer for on-line data processing, Nuclear Instrum. Methods 137 (1976) 189.

13) A. Vriens, private communication. See also:
A. Vriens, Description et test du mini-ordinateur DPNC-811, Travail de diplôme, Juillet 1976, Dép. Physique Nucléaire, Université de Genève.

14) C. Maclean, G. McPherson and P. Wilde, Special purpose computing hardware for processing data from experiments, Proc. 2nd ISPRA Nuclear Electronics Symp., Stresa, 1975, p. 307.

15) T.F. Droege, I. Gaines, K.J. Turner, The M7-A high-speed digital processor for second level trigger selections, IEEE Trans. Nuclear Science, Vol. NS-25 (1978) 698.

16) T. Lingjaerde, A fast microprogrammable processor, CERN DD/75/17.

17) D. Marland, ESOP Update, CERN DD/78/13.
T. Lingjaerde, CAMAC interface for the ESOP microprocessor, CERN DD/76/9.

18) T. Lingjaerde and D. Marland, A versatile multiport buffer memory system for fast data acquisition in high-energy physics, CERN DD/77/8.

19) P.F. Kunz, The LASS hardware processor, Nuclear Instrum. Methods 135 (1976) 435.

20) J.M. Meyer, private communication.

21) E. Barrelet, private communication.

22) P. Baillon et al. (CERN-Collège de France-Ecole Polytechnique Collaboration), Status report high precision measurement of π^-p total cross-section, Internal document CERN PSC/78/6.

23) H. Wind, Function parametrization, Proc. 1972 CERN School of Computing, CERN 72-21, p. 53-106. See also:
M. Hansroul, D. Townsend and P. Zanella, The application of multidimensional analysis techniques to the processing of event data from large spectrometers, *in* Proc. Meeting on Programming and Mathematical Methods for Solving Physical Problems, Dubna, Oct.-Nov. 1973 (also CERN DD/73/31). See also Ref. 24.

24) See any textbook on pattern recognition, for instance: J.R. Ullmann, Pattern recognition techniques, Butterworths, London, 1973, or
H.C. Andrews, Introduction to mathematical techniques in pattern recognition, Wiley Interscience, New York, 1972.

25) H. Brafman, M. Breidenbach, R. Hettel, T. Himel and D. Horelick, Fast track-finding trigger processor for the SLAC/LBL Mark II detector, IEEE Trans. Nuclear Science, Vol. NS-25 (1978) 692.

26) C. Baglin et al. (CERN-Annecy(LAPP)-Genoa-Copenhagen-Oslo-University College London Collaboration), Two body reactions at large p_T, Internal document CERN/SPSC/74-28.

27) J. Joosten, M.F. Letheren and B. Martin, private communication.
C. Verkerk, Special purpose processors for high-energy physics applications, Proc. 9th Internat. Symposium on Nuclear Electronics, Varna, 1977, p. 128-141. See also: CERN DD/77/6.

28) J.V. Allaby et al. (CERN-Hamburg-Amsterdam-Rome-Moscow(ITEP) Collaboration), Study of semileptonic neutral-current processes and of μ-polarization produced in ν and $\bar{\nu}$ interactions, using counter techniques, Internal document CERN/SPSC/75-59.

29) M.F. Letheren, private communication.

30) J.B. Lindsay, C. Millerin, J.C. Tarlé, H. Verwey and H. Wendler, A fast and flexible data acquisition system for multiwire proportional chambers and other detectors; to be published in Nuclear Instrum. Methods.

31) A. Fucci et al., A new and fast programmable trigger logic, Nuclear Instrum. Methods 147 (1977) 587.

32) A. de Bellefin et al., Application de l'adressage dispersé (hash-coding) à la reconnaissance de mots binaires, CERN/EF/77-1.

33) G. Fidecaro, J. Lindsay, I. Pizer and G. Delavallade, Programmable track selector system for multiwire chambers, Internal Report CERN EP/76/07, to be published in Nuclear Instrum. Methods.

34) M. Conversi and C. Verkerk, Flash chambers as 'active' detectors and a special processor for fast decision making, submitted to Nuclear Instrum. Methods (also CERN DD/78/3).

35) M.F. Letheren, private communication.

36) A. Beer et al., MBNIM: A modular approach to fast trigger logic for high-energy physics experiments, submitted to Nuclear Instrum. Methods, also CERN/EF 78-6.