USE OF INTELLIGENT DEVICES IN HIGH-ENERGY PHYSICS EXPERIMENTS

Lectures presented at the 1980 CERN School of Computing
Vraona, Attiki, Greece
14-27 September 1980
INTRODUCTION

Experimentation in high-energy physics relies more and more on the use of "intelligent" devices. Introduced in the mid-sixties, the on-line computer has been universally adopted in experiments to perform a number of tasks, such as data acquisition, data recording, and monitoring of experimental conditions. From the mid-seventies, physicists were aware that "intelligence" can be very fruitfully employed for additional tasks, amongst which event selection occupies a predominant position. At the same time it was realized that the data acquisition minicomputer was not always the most suitable device for performing these additional jobs. A number of instruments were thus invented and built to assist in -- or sometimes take over -- the most demanding tasks. It was natural that flexibility and adaptability were considered as assets of such a device. The formidable progress in semiconductor technology, has made it relatively easy, even for the non-expert, to build flexible devices. This has spurred the development of new techniques and instruments for particle-physics experimentation.

It has become a habit to call certain of these very flexible devices "intelligent". This explains the title of these lectures. For our purpose we should take "intelligent" to mean "programmable", this word in turn to be understood in the usual sense. A washing machine, where the user may select one out of a number of possible sequences, without the possibility to modify the selected sequence, is not programmable, in spite of the vendor's claim. On the other hand, it should not be assumed that programmable devices necessarily execute programs, instruction by instruction, as a computer does. Programmable means that the process leading to the result can be modified in a convenient way, with the help of software. Taken in this sense, a "look-up table" can be considered to be programmable, since its contents can be easily modified according to the results of, for instance, a simulation program. The software should define the behaviour of the instrument as a result of the stimuli -- input data -- it receives. The more this behaviour can be varied, the more "intelligent" we will find the device.

Intelligent devices are used for many tasks in experiments and the nature of the devices varies widely. In these lectures we will not consider the uses of main-frame computers, or the classical applications of minicomputers. We will concentrate on two areas for which special devices have been developed:

- On line data processing, generally to perform event selection and/or to achieve compaction of data before recording.
- Preparation of experimental apparatus: testing of detectors, optimization of operating conditions and calibration.

Much attention will be given to the event-selection process and the devices used for this purpose.
2. EVENT SELECTION

In many particle-physics experiments, the aim is to study rare events. The overwhelming majority of the interactions taking place in the target are of no particular interest and the physicist must devise sophisticated methods to recognize the small number of events of potential interest, so that these may be recorded for further analysis. The event-selection process therefore aims at distinguishing the "good" events from the "bad" or uninteresting ones. A number of benefits are derived if an effective event selection can be made:

i) The sample of recorded events becomes richer, i.e. the ratio of interesting to uninteresting events is improved.

ii) If an event of no interest can be rejected in a time which is shorter than the time it would take to acquire all data of the event for future recording, then the average deadtime of the apparatus may be shortened. The consequence is that the apparatus is sensitive for a larger fraction of time and thus the rate at which good events will be recorded increases. The sensitivity of the experiment is thus improved.

iii) The first of the foregoing benefits will result in a shortening of the computer time required for off-line analysis, for equal statistical significance.

iv) The smaller number of magnetic tapes needed alleviates operational and housekeeping problems.

v) The partial analysis of the events, performed in real time, can improve the monitoring of the experiment and the detection of abnormal operation of the apparatus.

The following example illustrates the first three of these benefits. Assume that the trigger rate, as determined by the usual scintillation counters and their associated fast logic is $10^3$/s and that only one out of 100 events thus selected is of potential interest. Assume further that $\tau = 4$ ms are needed to read out an event. As the fraction $F$ of triggers which can be recorded is

$$F = 1 - \frac{1}{1 + \frac{\tau}{R}},$$

where $R$ is the trigger rate, we see from Fig. 1 that we will be able to read not more than 200 events/s. The rate at which we are recording good events is then $2 \text{ s}^{-1}$. Assume now

![Fig. 1](image-url)

The fraction $F$ of potential triggers $R$ that can be recorded, plotted for different values of the dead time $\tau$ [from Halatsis et al. 93].
that each trigger is submitted to a further event-selection process, requiring 100 μs to
decide if an event is of potential interest or not. Assume also that 10% of the triggers
are accepted by the selection process and that all the really interesting events are con-
tained in this accepted sample (e.g. all good events are recognized as such and none is
rejected). Each trigger making its way to this selection will then be followed by a dead-
time of 100 μs and for 10% of these triggers a further 4 ms will be needed to acquire all the
data of the event. The average dead-time thus becomes \( \tau = 0.1 + 0.1 \times 4 = 0.5 \) ms. From
Fig. 1 we then see that 670 out of the 1000 triggers can now be examined, and that 67 events/s
will be recorded, containing 6.7 good events.

The volume of recorded data has been reduced by a factor of 200/67, resulting in a
compensatory reduction in CPU time for off-line analysis. In addition, the rate of recording
good events, and thus the sensitivity of the experiment, has improved by a factor of 6.7/2.
A figure of merit could be defined which is the product of these two improvement ratios.

A quantitative analysis can be found elsewhere\(^1,2\). Figure 2, from Ref. 2, shows the
results as the ratio of recorded events with the event selection process on \( (R_{on}) \) and off
\( (R_{off}) \). Figure 3 shows the ratio of recorded good events with the selection on \( (R_{G\ on}) \) and
off \( (R_{G\ off}) \). In both figures \( t_R \) is the read-out time of an event, \( t_D \) the decision time,
and \( 1/\rho \) the fraction of events accepted by the filter. Figure 3 shows that for very high
trigger rates \( R_{G\ on}/R_{G\ off} \) approaches \( \rho \). This is correct, but it should be remembered that
the absolute rates in this case are very small.

We saw that an effective selection process reduces the number of events to be recorded.
If now, instead of immediately recording this reduced number of events, we submit them to a
further selection process, we can obviously repeat the argument developed in the example.
But as we are now dealing with a reduced rate, we can afford to spend more time per event

---

**Fig. 2** Ratio of number of events recorded with event-selection process switched on
\( (R_{on}) \) to the number of events recorded with selection process switched off \( (R_{off}) \).
\( t_R \) is the recording time for an event, \( t_D \) the decision time. \( 1/\rho \) is the fraction
of events accepted by the filter [from Turals\(^1\)].

**Fig. 3** Ratio of number of interesting events recorded with selection process
switched on \( (R_{G\ on}) \) to the number of interesting events recorded with the
selection process switched off \( (R_{G\ off}) \). \( t_R \) = recording time, \( t_D \) = decision time,
\( 1/\rho \) = fraction of events accepted by the filter [from Turals\(^1\)].
without adverse effect on the average dead-time. We can then devise more sophisticated, but slower, selection processes and thus make a further reduction of recorded data. The number of interesting events will not increase however. (Note: The analysis in Refs. 1 and 2 suggests that the number of good events recorded does increase. This is, however, a matter of definition: "good" was taken to mean "acceptable to the selection process").

The decision time of an event filter is an important parameter, as we saw. Different accelerators impose different constraints, however. For a d.c. machine, such as the ISR, the considerations are valid all the time, while for a pulsed accelerator (PS, SPS) they are valid during the burst only. There is more time for processing available between bursts for complex selection processes, but the event data must then be transferred to a large buffer during the spill and at a high rate. At colliders, where particles circulate in bunches (PETRA, LEP), dead-time-free operation can be obtained if the decision time is shorter than the interval between the crossing of successive bunches. At LEP = 12 μs decision time will thus be available without penalty. Whatever the time constraints are for a first decision, it is generally true that more time is available for a decision at the next higher level, since less events are presented to it.

The higher level selection, in order to be more discriminating, must be more precise. This is illustrated in Fig. 4. Figure 4a shows the resolution in dimuon mass as it could

---

Fig. 4 Effect of limited resolution on the distribution of selected events. a) Hypothetical resolution for a selection based on effective mass. b) Acceptance when resolution is as in a) and the cut-off is set at 4 GeV/c². c) Sketch of the cross-section for producing dimuons. d) Result when acceptance of b) is applied to c). The J/ψ peak at 3 GeV/c² still dominates.
be obtained by a selection process, working on data from a hypothetical piece of apparatus. This means, that if we want to discriminate against J/ψ by setting the threshold of acceptance at a measured effective mass of 4 GeV/c², then the probability of accepting different masses is as in Fig. 4b. The cross-section being as sketched in Fig. 4c, the mass distribution for the accepted events will be as in Fig. 4d. This distribution is still largely dominated by the J/ψ peak. The better the resolution that can be obtained, the better the discrimination against the unwanted J/ψ becomes.

To obtain a good resolution for the selection, two conditions must obviously be fulfilled:
- the input data, representing the measurements, must be sufficiently accurate;
- the algorithm must take full advantage of this accuracy.

Limiting ourselves to input data representing measurements of particle positions, then the sources of data could be as indicated in Table 1. Going from scintillation-counter hodoscopes to drift chambers the accuracy increases from several centimetres to a fraction of a millimetre. Similarly for calorimeters, where valuable data can be extracted already on a "hit/no hit" basis, but where full exploitation requires precise pulse-height information.

Table 1

<table>
<thead>
<tr>
<th>Source</th>
<th>Resolution</th>
<th>Time delay</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scintillation counters</td>
<td>1-20 cm</td>
<td>1-100 ns</td>
<td>Simple apparatus, simple algorithm, short decision time</td>
</tr>
<tr>
<td>(hodoscopes)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groups of prop. wires, drift wire number</td>
<td>≈ 5 cm</td>
<td>10 ns-1 μs</td>
<td></td>
</tr>
<tr>
<td>Individual prop. wires, drift-time information, pulse heights</td>
<td>0.1-1 mm</td>
<td>1 μs-1 ms</td>
<td>Complex apparatus, complex algorithm, long decision time</td>
</tr>
</tbody>
</table>

For the event-selection process the complexity of the apparatus and of the algorithm increases in general with improved accuracy. An important consequence is also that complex read-out systems are needed to acquire the data and that a fair amount of time is required to accomplish this task. As the algorithm is not limited any longer to simple logical operations, but requires performing arithmetic operations, on many pairs of input data, its time requirements also increase. The decision time unavoidably increases when accuracy and selectivity are aimed at.

One more remark should be made concerning the principles of event selection: the condition that the decision time be shorter than the read-out time must be satisfied to achieve a higher rate of recording good events. This can only be obtained if the data needed for the selection algorithm can be made available fast enough. The normal read-out, via CAMAC modules, into a standard minicomputer takes 1-2 μs per data word. As we have also to allow time for the processing stage proper, it is clear that the reading of the input data for
the selection process must be done at a rate which is considerably higher: 100-200 ns/word. In addition, only those data really needed for the selection should be read and all data of no immediate interest should be left in suspense. Effective event selection has therefore an important influence on the general configuration of the read-out system and on the selection of its components. We will see examples in these lectures.

3. EXAMPLES OF THE SELECTION OF DIMUON EVENTS

To illustrate the points made above we will examine rather rapidly the selection process in four different experiments, each aiming at the study of high-mass dimuons. This will also serve to introduce the different techniques and some of the devices used. The triggers and event filters will be presented in an order of increasing sophistication, particularly for the devices. The effective mass of a dimuon is given by

\[ M_{\mu\mu}^e = 2p_1 p_2 (1 - \cos \theta), \]  

(2)

where \( p_1 \) and \( p_2 \) are the momenta of the two emerging muons and \( \theta \) the angle between them. In very first approximation one may write

\[ M_{\mu\mu} \approx p_{T1} + p_{T2}, \]  

(3)

where \( p_{T1} \) and \( p_{T2} \) are the transverse momenta. The last formula is useful for a rough selection as we will see.

1) The first example concerns an experiment at FNAL in a 225 GeV/c hadron beam\(^3\). The input signals were derived from scintillator hodoscopes, as schematically indicated in Fig. 5.

A very fast decision (\( \approx 500 \) ns) was required, as the trigger signal had to be used to strobe multiwire proportional chambers (MWPCs). The trigger is, in principle, based on the use of the exact formula (2), or rather of its logarithm:

\[ \log M^2/2 = \log p_1 + \log p_2 + \log (1 - \cos \theta). \]

Because of the requirement of short decision time, which does not allow different combinations to be tried, it was assumed that one of the muons was detected in the upper half of the apparatus and the other in the lower half. The two halves were independent, as indicated in Fig. 5. In each of the two halves a coincidence between a \( J_X \) and an \( F \) counter corresponds to an interval in \( p \), determined by the magnetic field.

![Fig. 5 Schematic view of hodoscopes used in a dimuon trigger at FNAL [from Hogan\(^4\)].](image)
Fig. 6 Block diagram of the logic used for the fast mass dependent trigger of Hogan^{1}.

Several combinations, two by two, of a $J_x$ and an $F$ counter lead to the same $p$. The outputs of the hodoscopes were thus wired up in two-input AND gates and the outputs of the AND gates corresponding to eight predefined intervals in log $p$ were ORed together. The eight intervals in log $p$ were then encoded into three bits. In a similar way log $(1 - \cos \theta)$ was encoded. The three encoded signals log $P_{up}$, log $P_{down}$ and log $(1 - \cos \theta)$ were added and then compared with a threshold value, set by front-panel switches. A block diagram is shown in Fig. 6. It should be realized that the blocks marked "matrix encoder" represent a fair amount of logic and a fixed pattern of connections to the scintillators, which has been determined once and for all by Monte Carlo calculations. The user has thus very little control over and influence on the selection process, apart from setting the threshold. Owing to the coarse input data and the use of few intervals in log $p$, etc. the resolution was $\approx \pm 4$ GeV/c$^2$ at the $J/\psi$ peak, just good enough to discriminate against $p$'s.

ii) The second example is experiment NA3 at CERN^{6}). Cathode read-out of two proportional chambers is used. The cathodes of these chambers are divided into a particular pattern (Fig. 7): sectors are defined such that the increase in $\tan \phi$ from sector to sector is constant: $\Delta \tan \phi = \text{const}$. The cathodes are also divided into horizontal strips. The second chamber is an enlargement of the first, in the ratio of their distances $Z_1$ and $Z_2$ to the target. The first-level selection is based on the vertical component of the transverse momentum $p_{TV}$, which can be expressed in terms of $\phi$ and $Z_1$ and $Z_2$:

$$\Delta = \tan \phi_2 - \tan \phi_1 = \frac{qBZ}{p_{TV}} \left( \frac{1}{Z_1} - \frac{1}{Z_2} \right).$$

(4)

As this first-level trigger has again to be very fast (less than 200 ns), a large amount of circuitry is required: all 64 horizontal strips are treated simultaneously, in 64 replications of the same circuit. The logic is more sophisticated than in the previous example:
Fig. 7 Schematic diagram of the detectors used in experiment NA3 [from Dubé]. The detectors of interest here are the wire chambers M1 and M2, with their particular pattern on the cathodes.

It allows for two hits in a horizontal strip. The outputs from the cathode fields are fed to two priority encoders, one encoding the position of the leftmost hit, and the other the position of the rightmost hit, for each strip. The encoded positions (6 bits each) of the first and the second chamber are subtracted, to give a 4-bit result corresponding to \( \tan \phi_1 - \tan \phi_2 \). As we can have two hits in each chamber, four differences are calculated, as can be seen from the block diagram of Fig. 8. The 4-bit result is used as an address.

Fig. 8 Block diagram of logic for transverse momentum trigger of NA3 [from Boucrot et al.].
for a look-up table, which does the conversion of $\Delta$ into $p_{TV}$. More precisely, the look-up table delivers an output bit if $p_{TV}$ is above a certain threshold; otherwise a "0" is output. One now only has to OR the signals of the 64 strips to derive the trigger. If the trigger conditions need to be changed, the look-up tables can be reloaded with another pattern. In fact, the output is 4 bits wide, which allows the definition of four different trigger conditions without the need for reloading. The decision is made in 110 ns, as the whole logic is made in ECL circuitry.

The user still does not have much freedom to influence the decision process, apart from a wider choice of trigger conditions and greater convenience in loading these conditions in the apparatus.

This first-level trigger, which has been in use in many runs of the experiment, has however had two interesting extensions. The first is based on the same technique, but selects by dimuon mass, according to the formula $M = \sqrt{p_{TV1} p_{TV2}} \theta_{12}$, which can be approximated by

$$M \approx \frac{\theta_{1V} + \theta_{2V}}{\sqrt{\theta_{1V} \theta_{2V}}} \cdot \frac{1}{\cos \theta}.$$  \hspace{1cm} (5)

Here $p_{TV1}$ is the vertical component of the momentum of the first particle, and is available in the form of $\Delta$ from the first-level trigger; $\theta_{1V}$ is the angle in a vertical plane for the first particle and it is made available by encoding the number $N_1$ of the horizontal strip hit; $\theta$ is the average azimuthal angle and it can be obtained from encoding the sectors $A_1$ and $A_2$ of the cathode, as was already done before. The formula for the mass thus separates into three independent parts:

$$M \approx f(A_1, A_2) \cdot g(N_1, N_2) \cdot h(A_1, A_2).$$  \hspace{1cm} (6)

Each of the three functions can be calculated by making use of look-up tables. The factor $f$ is obtained as a 4-bit result; $g$ and $h$, which vary less in magnitude, are obtained as 3-bit quantities. The three quantities are concatenated into 10 bits, which are used as the address for a final look-up table to find $M$. The total decision time for this extension of the trigger is 170 ns \(^1\). The resolution at the $J/\psi$ is close to ±0.3 GeV/c\(^2\).

The second extension of the event selection process uses a hardwired processor. The events selected during the beam burst are stored. Between bursts the processor finds all tracks in an event and the data-acquisition computer (a PDP-11/45) then calculates the dimuon mass, using the exact formula (2). MORFION, the hardwired processor\(^5\) finds straight tracks in 8 planes of MWPCs. A minimum of 6 points must be found on a track. The processor uses a similar technique to the one used in previous track-finders\(^6,7\); a block diagram is shown in Fig. 9. The processor needs 5 ns to find all tracks in a high-multiplicity event (6 tracks in x- and in y-projection). The events with a calculated mass $M > 3.8$ GeV/c\(^2\) are written on a separate tape. They represent = 4% of the previous total and 97% rejection at the mass of the $\psi$ is achieved (see Fig. 10).

This experiment provides an example of a multilevel selection. The selection criteria can be reasonably well and conveniently controlled by the experimenter, although none of the devices used can be programmed in its own right.
iii) The third example of high-mass dimuon selection is experiment E400 at FNAL. Drift-chamber data are used, which are read via a fast digitizing system into a specialized micro-programmed processor). Two $x$-planes and two $y$-planes are used and the mass is expressed directly in the measured coordinates:

$$M^2 = \frac{K^2[(A_1\Delta x_2 - B_1\Delta x_1)^2 + (A_2\Delta y_2 - B_2\Delta y_1)^2]}{(A_1y_2 - B_1y_1)(A_2y_2 - B_2y_1)}$$

(7)
The transverse momentum is calculated by a similar formula. Both formulae can be calculated in an 18-step algorithm. Fourteen of the 18 steps turn out to be of the form \( ax = by + c \).

The processor has therefore been optimized for very fast execution of this operation. Four data memories allow simultaneous access to the four operands, two multipliers operate simultaneously to produce \( ax \) and \( by \) and an ALU performs the addition or subtraction. The result \( c \) can be returned to the data memories. The processor is microprogrammable and not restricted to the execution of this 18-step program. For instance, three index registers have been added to make the device more generally applicable.

The processor has been implemented in ECL technology and only 200 ns are needed to obtain \( ax \pm by \). In addition, the instruction fetch and the instruction execution are pipelined, such that a result becomes available every 100 ns. The 18-step calculation is thus performed in 2.1 \( \mu \text{s} \) for a single set of input data.

The diagram in Fig. 11 shows a general way of integrating such a processor into the data-acquisition system. The primary trigger sets off the processor and the fast read-out of a subset of the data. The processor provides either a reset signal, or, for a selected event, an interrupt to the data-acquisition computer, which then proceeds to the full recording of the event.

iv) The last example is again one of a multilevel selection; it concerns experiment NA10 at the SPS \(^{9,10}\). The experiment uses a toroidal magnet and the scintillator hodoscopes are adapted to this: as a particle trajectory remains entirely in a plane containing the beam axis, ring counters centred on the beam would be indicated. For practical reasons, four

---

Fig. 11 Diagram of the equipment for second-level event selection in experiment E400 at FNAL \([\text{from Drege et al.}^{8}]\). Note how the processor is integrated with the data-acquisition system.
hexagonal hodoscopes are used; each consisting of 6 sectors with 32 counters each (Fig. 12). The experiment receives a very high intensity beam and the counting rate for an individual counter may be as high as $10^6$-$10^7$ s$^{-1}$.

The first-level trigger therefore must have a very short dead-time and nevertheless be sufficiently selective. It is based on selecting the mass, approximated by

$$M = P_{T1} + P_{T2}.$$  

A twofold coincidence between one counter in $R_1$ and one in $R_2$ defines the angle of a track. For each angle one out of 32 coincidence matrices is enabled, receiving inputs from $R_3$ and $R_4$. The output of the coincidence matrix is an encoded value of $p_T$, to be used later in the second-level selection. For the first-level decision $p_T$ is subdivided into four intervals, A, B, C, and D. The logic is replicated for each of the six sectors. The four-interval $p_T$'s from two sectors are combined and a trigger signal generated according to Fig. 13, when $M \geq 4$ GeV/c$^2$. The decision time $t_D = 140$ ns.

Fig. 13 Approximation of $M_{uu} \geq 4$ GeV/c$^2$, in the first-level trigger of NA10. A, B, C and D correspond to intervals in transverse momentum [from Degré's].
Fig. 14. Diagram of event-selection apparatus for NA10 [from Degrè9]. Four GESPROs and four event buffers are used.

The second-level selection uses 4 microprogrammable processors, GESPROs 11-13). Each GESPRO works on data contained in its own multiport memory. The four buffer memories (see Fig. 14) are filled with chamber data in a round-robin fashion by a fast read-out system RMH 14). The RMH system performs a read-out in = 100 µs. One mass calculation in GESPRO takes = 20 µs. If every sector has a hit, a total of 15 mass-calculations must be made and the total time may be as long as 500 µs. Therefore a buffer memory may be occupied during 400 µs for an event and four buffers and GESPROs are needed if we do not want to introduce more dead-time than determined by the speed of the RMH. The CAAC read-out uses a third port on each of the buffer memories and thus has access to the data of selected events. Via a separate path the GESPROs receive the values of PT determined by the hodoscopes and the first-level trigger. This allows an early abort of the uninteresting events.

What conclusions can we draw from these four examples? Concerning the techniques, we may notice that sometimes a special layout of the experimental apparatus is used, aiming at a simplification of the selections e.g. special patterns on the cathode of chambers, or a toroidal magnetic field combined with hexagonal geometry of the detectors.

In addition to the classical NIM logic, the first-level triggers use coincidence matrices, look-up tables and, sometimes, simple arithmetic. Scintillator data or other low-resolution data are used and often simplifying assumptions are made, e.g. on the number of particles or on their spatial distribution. It should also be clear from the examples that a fast first-level trigger needs a large amount of modules and a large amount of wires. The use of look-up tables has clearly been a very important step forward allowing a reduction of the amount of logic. In practice, this technique is not so much used to reduce the amount of logic for a well-established trigger system, but rather to build much more sophisticated
selection mechanisms, using the same volume of electronics as required before. The interested reader will find several more examples in the digest of a meeting on trigger logic which was held at CERN \(^{18}\).

In the second-level decision process we see that arithmetic processors are used: sometimes hardwired, but more often microprogrammable. They use generally a subset of the data, at the full precision available, and their algorithms make no or few simplifying assumptions. For effective operation they require in most cases a fast, separate read-out. An aspect which has not been explicitly shown in the examples is that powerful test facilities are essential to overcome any doubt that may arise concerning the correct operation of these devices. The strict minimum needed is the possibility of being loaded with programs and data for which the results are known. Checking of these results improves confidence, but is not an absolute proof of correct operation under all circumstances.

4. DEVICES FOR EVENT SELECTION

Now that most of the types of devices used in event selection have been introduced in the examples, we should have a closer look at some of them. We will follow a rising slope of user programmability.

4.1 Look-up tables

The majority of the look-up tables in use in high-energy experiments are implemented as random access memories (RAMs), although some examples of programmable read-only memories (PROMs) or field-programmable logic array (FPLAs) can be found, notably in some PETRA experiments. A RAM has the obvious advantage of being conveniently loadable, which allows rapid change of selection criteria, without the need to remove circuits.

It is important to note that look-up tables are used for two different purposes:

- Logic decisions: In this case a bit pattern which must be recognized as valid or invalid is used as the address of the memory. The addressed memory cell needs to contain not more than a single bit to indicate the answer.

- Arithmetic: In this case the address is formed by a single operand to evaluate the value of a function or by a concatenation of two or more operands to evaluate the result of a -- possibly complex -- arithmetic operation.

Obviously the availability of cheap memory chips of considerable capacity has stimulated the application of this technique. Since each bit added to the input pattern increases the memory size by a factor of two, the use of look-up tables is limited in practice to inputs of not more than 15 or so bits. In some cases techniques exist to reduce the memory size required, but always at the expense of speed. For instance, an 8-bit by 8-bit multiplier would require a 64K memory of 16-bit words: a total of \(2^{23}\) bits. Splitting the operands into two groups of 4 bits each and using cross-multiplication, we only need 4 memories of 8-bit addressing capability (256 words) and 8 bits wide: a total of \(2^{13}\) bits. We need, however, additional adders and more time to add the partial results together. If we can allow the system to be still four times slower we can limit the equipment to a single memory to look up the four partial products in successive time slots. It is important to note that this technique can be applied to all linear problems\(^1\).
Fig. 15 Example of the use of MBNIM modules. From the 16-bit pattern in chamber H1, the pattern in chamber H2 is predicted and compared with the true pattern of bits (from Ref. 18).

Fig. 16 Another example of the use of MBNIM. The momentum of a particle is estimated from bits in three planes (from Ref. 18).
For logic decisions two techniques to reduce memory size are known:

- hash coding\textsuperscript{4)}, and
- successive indirect addressing\textsuperscript{1}).

A rapid survey of the recent uses of look-up tables\textsuperscript{14}) reveals that in most cases ad hoc solutions have been adopted. Only two modular approaches have emerged so far.

The first is the MBNM system, which uses direct ECL inputs to the modules, from a 16-bit data bus, which runs along the front panels\textsuperscript{17}). A number of modules exist\textsuperscript{18}), the most important ones being:

- a 16-bit ALU allowing arithmetic and logic operations (e.g. a bit-by-bit AND),
- a 1 k × 16-bit random access high-speed memory (so called RAME) for use as look-up table,
- the so-called bit-assigner, which converts a 16-bit pattern into another one. The bit-assigner is useful to predict, for instance, from a pattern of hits in one chamber, the expected pattern of hits in another chamber. This is illustrated in Fig. 15. Figure 16 shows a more complicated example, where the momentum of a particle is calculated using standard modules. Complete triggers for experiments have been constructed using the MBNM system\textsuperscript{19,20}).

The second modular system has been developed at FNAL\textsuperscript{21}). The system is very complex and ambitious. A total of 18 modules have been designed; the most important ones are:

- memory look-up (MLU); this module can be changed from a 12-bit address, 16-bit output configuration to one with a 16-bit address, 1-bit output via a set of intermediate possibilities;
- a stack module;
- a do-loop indexer;
- a track finder;
- a vertex parameter module.

The system is intended for building complex triggers, containing complete hardwired processors. Owing to the complexity of the modules, it would take us too long to describe in more detail how the modules can be put together to implement complete hardwired processors. The reader is referred to the original publication\textsuperscript{21}).

4.2 Hardwired processors

Barsotti's system provides a transition from the look-up table to the hardwired processor. Several hardwired processors have been used in a number of experiments. MORPION was already mentioned; other examples are the track-finding processor of the Mark II detector at SLAC\textsuperscript{22}) and the processors used in WA7\textsuperscript{7}) and WA18\textsuperscript{11,23}) at CERN. A review has been presented by Turala\textsuperscript{2}). Hardwired processors are superior to look-up tables when more complex situations must be dealt with. Particularly in the case where several possible combinations of input data must be examined, with the need of looping over arrays of data, these processors are much more powerful than the simple look-up table. Their speed is, however, lower than for the look-up technique and the user cannot exert a great influence on their behaviour. They execute one particular algorithm, and since the time to implement a processor is of the
order of several months, this algorithm must either be known in all details well before the experiment starts or else the experiment must take data for a year or so before the processor can be integrated. Both situations do not help to gain the confidence of the physicist in such a "black box", whose operation he cannot control in detail.

Although a hardwired solution to an algorithmic process is in general faster than a programmable implementation, it is therefore not astonishing that the latter are in most cases preferred. Nevertheless, a rather modular approach has been proposed rather recently\(^2\).  

4.5 Microprogrammable processors

Similar to the way in which the use of look-up tables has been spurred by the abundance of cheap memory chips, the development of microprogrammable processors received its impetus from the availability of bit-slices. The bit-slice integrated circuits were introduced by semiconductor manufacturers in an attempt to overcome the speed limitations of MOS microprocessors. The technologies (TTL, ECL) required to attain higher speeds do not allow integration at densities as high as MOS circuits. The semiconductor manufacturers were thus obliged to use multiple chips to implement fast microprocessors. Two ways were used to reduce the number of transistors per chip. Firstly, the data flow path was kept separate from the control of the processor, leading to two different types of integrated circuits. Secondly, a further reduction of the number of transistors per chip was obtained by implementing these two separate parts in the form of vertical "bit-slices" of, for instance, 4 bits width. Connecting several bit-slices in parallel would then lead to a processor of 16, 24, or even 32 bits. The bit-slices were sold, inappropriately, as "bipolar microprocessors". The price to be paid for the increased speed was the necessity of microprogramming. More details on bit-slices, how they can be combined into a complete processor, how they can be microprogrammed, and the tools available for program development, can be found in the lectures by van Dam\(^3\) and Halatsis\(^4\).

Naturally, physicists were attracted by the bit-slices as their speed promised to be more satisfactory than for the single-chip, fixed-instruction-set microprocessors. Several microprogrammable processors were thus developed using different types of bit-slices. Note, however, that the two oldest ones (ESOP and M7), did not yet use bit-slices, but normal medium-scale integrated circuits instead\(^5,\)\(^6\).

The microprogrammable processors developed for high- and medium-energy physics applications known to the author are summarized in Table 2. In this table are grouped together those processors having their proper structure without regard to an existing machine or instruction set. (Another group of microprogrammed processors, which imitate or "emulate" existing machines will be treated later). As a result, these processors must be programmed at the microcode level, with -- to the author's knowledge -- only one exception at present: GESPRO.

Inspection of Table 2 reveals some interesting points:
- The microcycle time varies by a factor of two between the extremes: the processors built in ECL have a cycle time at best 0.6-0.7 of the cycle of the TTL processors.
- Nearly all of the processors are related to CAMAC, either because they are implemented as a CAMAC module, or because they have been designed to control a CAMAC crate or branch.
Table 2
Microprogrammed processors

<table>
<thead>
<tr>
<th>Processor</th>
<th>CAB</th>
<th>ESP</th>
<th>GESIPO</th>
<th>&quot;Olivetti&quot;</th>
<th>&quot;Mach 4&quot;</th>
<th>VERTIX</th>
<th>M7</th>
<th>SAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Institute</td>
<td>Ecole Poly.</td>
<td>Genova</td>
<td>Grenoble</td>
<td>Strasbourg</td>
<td>Pisa</td>
<td>Lisbon</td>
<td>Hamburg</td>
<td>Bologna</td>
</tr>
<tr>
<td>Reference</td>
<td>26</td>
<td>27</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>21</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>Technology/slice</td>
<td>2901</td>
<td>3000</td>
<td>3000</td>
<td>2901</td>
<td>2901</td>
<td>10000</td>
<td>2901</td>
<td>2901</td>
</tr>
<tr>
<td>Microcycle (ns)</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>Micro-instruction (bits)</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Progr. mem. (words)</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
</tr>
<tr>
<td>Data word (bits)</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Max. data mem. (words)</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
</tr>
<tr>
<td>Special features</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
</tr>
<tr>
<td>Read-out systems for which interfaces exist</td>
<td>CAMAC, GPIB</td>
<td>CAMAC, GPIB</td>
<td>CAMAC, GPIB</td>
<td>CAMAC, GPIB</td>
<td>CAMAC, GPIB</td>
<td>CAMAC, GPIB</td>
<td>CAMAC, GPIB</td>
<td>CAMAC, GPIB</td>
</tr>
<tr>
<td>Remarks</td>
<td>FORTRAN</td>
<td>FORTRAN</td>
<td>FORTRAN</td>
<td>FORTRAN</td>
<td>FORTRAN</td>
<td>FORTRAN</td>
<td>FORTRAN</td>
<td>FORTRAN</td>
</tr>
</tbody>
</table>

- A number of the processors have been equipped with special interfaces to enable fast data acquisition. One may even wonder why this is not always the case. The answer is probably that in those cases where no direct interface to a fast read-out is foreseen (e.g. GESIPO), rapid access to the data is expected to take place via a fast buffer memory.

- Another interesting point is that several of these processors are equipped with special features: hardwired subunits designed to speed up the operations occurring often in the filter algorithms.

- Finally, we note that the width of the micro-instruction word varies considerably.

The last fact leads to a small aside on "vertical" and "horizontal" microprogramming\(^{33-35}\). In a vertically microprogrammed machine the micro-instruction word is short and the subfields of the word are encoded. Since each field is encoded, only one elementary action can be specified by it. This has the advantage that there is no danger of specifying conflicting actions in the same instruction. The disadvantage of a vertical micro-instruction format is that the potential for parallelism the hardware may have cannot be fully exploited. In addition, and in order to reduce the width even further, the same field may be used to specify actions on different parts of the machine. Which part should be acted upon is then specified by one or more mode bits in the micro-instruction. Since this limits even further the possible parallelism, a single micro-instruction does not have great power, but programming resembles writing an assembly-language program. Programming is then relatively easy. This advantage -- and some saving in hardware -- has been obtained at the expense of speed: more micro-instructions -- and thus cycles -- must be performed to arrive at a tangible result. An example of a vertical microprogram is shown in Table 3, which is a piece of a program for the CAMAC booster (CAB)\(^{28}\). It leaves, in fact, the impression that such programming could be quickly understood and mastered.
Table 3
Example of vertical microcode [CAB18]

<table>
<thead>
<tr>
<th>$CYCLE COURT: ADD. RETOUR BE SAMPLING $</th>
</tr>
</thead>
<tbody>
<tr>
<td>0400 053304 START TMA LOT</td>
</tr>
<tr>
<td>0401 006357    FIF = D</td>
</tr>
<tr>
<td>0402 007203    FIF = B3 + 1</td>
</tr>
<tr>
<td>0403 166017 NEXT G 3 3 REN REP AC</td>
</tr>
<tr>
<td>0404 036334    FIF = A3 - 1</td>
</tr>
<tr>
<td>0405 132035    JSN SAMPLE</td>
</tr>
<tr>
<td>0406 104071    JNC IDLE</td>
</tr>
<tr>
<td>0407 134002 COURT JSY NEXT - 1</td>
</tr>
<tr>
<td>0410 040036   TPX 0</td>
</tr>
<tr>
<td>0411 040136   TPX 1</td>
</tr>
<tr>
<td>0412 126245    JSR WFX</td>
</tr>
<tr>
<td>0413 040236   TPX 02</td>
</tr>
<tr>
<td>0414 040003    TKA</td>
</tr>
<tr>
<td>0415 001025    NFF = D - A0</td>
</tr>
<tr>
<td>0416 132253    JSN CTX</td>
</tr>
<tr>
<td>0417 011120    NF + A1 = Q</td>
</tr>
<tr>
<td>0420 132253    JSN CTX</td>
</tr>
<tr>
<td>0421 126257    JSR WFX</td>
</tr>
<tr>
<td>0422 040236   TPX 02</td>
</tr>
<tr>
<td>0423 040036   TPX 00</td>
</tr>
<tr>
<td>0424 040136   TPX 01</td>
</tr>
<tr>
<td>0425 040003    TKA</td>
</tr>
<tr>
<td>0426 021025    NFF = D - A2</td>
</tr>
<tr>
<td>0427 132262    JSN CTY</td>
</tr>
<tr>
<td>0430 011120    NF = A1 - Q</td>
</tr>
<tr>
<td>0431 112005    JSN NEXT</td>
</tr>
<tr>
<td>0432 120262    J CTY</td>
</tr>
</tbody>
</table>

In a horizontally microprogrammed machine, on the contrary, many bits are used in the microword. Many fields are defined and none is used for more than one purpose: e. g. the control of the ALU has its own field, which is separate from the field which controls the origin of the operands, which in turn is separate from the field controlling the destination of the result. Sometimes an individual field may be encoded, but often this is not the case. Every bit has then its own, proper function. It is now easy to specify conflicting actions -- which ought however to be detected by the micro-assembler! -- but one has gained full control over the whole potential of the machine. Micro-instructions can therefore be very powerful if the microprogrammer is capable of exploiting all this potential. Very effective and short programs can be written, the classical example being the single micro-instruction loop: the operation, the test on the result, the branch condition, and the branch address are specified in the same word, together with a bit to inhibit incrementing the microprogram address counter. The programmer must, however, have an intimate knowledge of the hardware. Good programming tools (micro-assembler and a simulator) are also very important. The lectures on software tools for microprocessors18) treat in more depth the microcoding of a true horizontal machine (MICE). Table 4 shows an example of microcode for MICE, taken from Joosten18).
Example of horizontal microcode ([MICE] Ref. 36)

ORR 10020 "MICRO-CODED COMPARE LOOP"

"Compare the elements of an array with a constant, taking one microcycle per element:
R0 = base address of array
R1 = compare argument
R2 = two's complement of array length
after execution R2 contains zero if a compare hit was found
otherwise it is non-zero
R0 points to the element that gave a hit
the PDP11 condition codes are returned as at entry
note that the incremented micro-instruction must be assembled at an
address which is a multiple of four"

; CPR
RF(0,RJ)B PNT308,0,MAR ADR
RF(2,RW)IB LATCH IB
DECRL1,1111B
"load instruction register at the end of the cycle"
READ LTR
INC
"increment micro-program address register CRO"

1002:00000000 00000000 0110110001 10111000 0010 0000 0 0100 0 0000 0 0010 0 0000
00 11100000 1101111111111111 01 0100 0010111111111110 0000000000 0000000000 0100000000
ZEROZ IB SP(3,IB)
READ BDR
MOD(3,MAR,2) MAR
RF(2,RJ)OB GDBNA RSR
"read element and load in DR"
"load RF2 in CR1 and increment CR0"

1003:00000000 00000000 00001001 10111000 0010 0101110010011010 1000 0000 0 0100 0 0000
00 11100000 1110111111111111 01 0100 00101111111110 0000000000 0000000000 0100000000
"this is the repeated micro-instruction, to scan the array for a hit"
READ PSR
"read new element load into DR while DR goes to 0-bugs"
RF(1,RJ)IB LATCH IB
SUBA(L1,1111B SP(3,SMAN)
"test an element"
MOD(3,MAR,2)MAR
CONDITION(EQ,11)
EMRMD MOD RPI L2
"increment array pointer"
"modify, repeat or jump"

1004:00000000 00000000 0110110001 10111000 0010 1110111111111111 110 0100 0100111111111110 0000000000 0100000000
"there was a hit; clear R2, load hit-pointer in R0 and jump to L2"

we entered this micro-instruction due to the RSR in the next micro-instruction that incremented CRO which was still pointing to the
previous micro-instruction
ZEROZ OB RF(2,0) OB
"clear R2"
MOD(3,MAR,-6) MAR AIB RF(0,1) IB
"load pointer of hit element in R0"
L2 "CR1-7777, therefore jump"

1005:00000000 00000000 0000100110 10111000 0001 0001111111111110 00000000 0 0100 0 0000
00 11100000 1110111111111111 110 0100 0100111111111110 0000000000 0100000000
"there was a hit and EMRMD brought us here; CRO has still the
address of the repeated micro-instruction (1004 octal).
we will load all ones in CR1 and increment CRO so that we will go
address 1005 where the RSR FF will be reset"
CONSTANT-CNS313 GDBNA RSR
"load ones in CR1 and increment CR0"

1006:00000000 00000000 0000000011 10000000 0000 0001111111111110 00000000 0 0100 0 0000
00 11100000 1110111111111111 011 0100 0001111111111111 0000000000 0100000000
JMP 948 "not used"

1007:00000000 00000000 0000000011 11001100 0010 0100 0100000000000000 0600 0000 0 0100 0 0000
00 11100000 100100000000001111 010 0100 0001111111111111 0000000000 0100000000
"we are going to leave the COMPARE micro-routine"
L2 "load the A register of the pipeline from IR"
LA PNT3(FC,2)PC,MAR TRDNO
SPS(SCR)
CONDITION(EQ,11)
BRC FORK
"fork to execute next PDP11 instruction"

1019:00000000 00000000 00000000000000001 11100000 0101 00000001100111010 0000 0000 0 0100 0000
00 11100000 1010111111111111 010 0100 0001111111111111 0000000000 0100000000
JMP 948 "not used"

1011:00000000 00000000 00000000000000001 11100000 0000 00000000000000000 00000000 0 0100 0 0000
00 11100000 101010000000001001 010 0100 0001111111111111 0000000000 0000000000
"here we came if there was a hit at the last element"
ZEROZ OB RF(2,0) OB
"clear R2"
MOD(3,MAR,-6) MAR AIB RF(0,1) IB
"load pointer of hit element in R0"
L2 "not used"

1012:00000000 00000000 00000000000000010 11100000 0001 00000000000000000 00000000 0 0100 0 0000
00 11100000 101010000000001000 010 0100 010011111111111111 0000000000 0100000000
END;
An important conclusion must be drawn from this excursion into horizontal and vertical microprogramming: the execution time of a given algorithm on different machines is by no means proportional to the microcycle times! A more useful parameter seems to be the ratio of width of micro-instruction to cycle time. At least for the different members of the PDP-11 family this ratio is fairly proportional to the speed.\textsuperscript{16, 17}

From the programming point of view, GESPRO seems to be different from the other processors in Table 2. For this machine an assembly language is defined. The cross assembler, which runs on a NORD-10 translates this language into microcode.\textsuperscript{18} GESPRO was in fact the first processor for high-energy physics built from bit-slices. It has been successfully used since 1977 in experiments and its projected use in NA10 has already been mentioned.

4.3.3 ESOP in an experiment

ESOP is at present used in three experiments at CERN: NA11, B807 and the European Hybrid Spectrometer (EHS). The processor itself will not be described here; the interested reader is referred to the reports by Lingjaerde\textsuperscript{27} and Jacobs.\textsuperscript{28} Suffice it to say that the processor has three ALUs, one for operations on data, one for calculation of the next instruction address, and one for data-address calculations. The 16-bit data memory is obviously separate from the 48-bit micro-instruction memory. A number of special processing units are available: a 16-bit by 16-bit multiplier, a fast shifter, a bit search unit, and a unit for making loose comparisons. The latter unit tests if $|A - B| < \varepsilon$. A number of I/O modules have been developed, both for CAMAC and for fast or specialized read-out systems (ROM, UTRC, ROMULUS/REBUS). A multiport memory also exists.

As an example of the use of ESOP in an experiment we will take NA11. This choice is made because it is probably the first experiment where a microprogrammable processor was used by physicists who had not been involved in its development and who had no previous knowledge of the machine. The results have been published\textsuperscript{29}.

The experiment NA11 studies charmed particles, produced by $\bar{p}$ at 175 GeV/c:

$$\bar{p} + Be \rightarrow D + \bar{D} + x \rightarrow e + \nu + x \rightarrow K + n\pi.$$

The event selection tries to detect a single electron or positron, in a set-up schematically indicated in Fig. 17. The trigger is based on a signal from the threshold Čerenkov detector.

- Fig. 17 Schematic of part of the layout of experiment NA11, indicating the detectors (MWPC and electron calorimeter) providing data for ESOP (from Damerell et al.\textsuperscript{29}).
and an energy deposition of $\geq 2.8$ GeV in the calorimeter. Several types of background events can cause the trigger conditions to be satisfied:

- a $\pi$ above 12 GeV/c (the Čerenkov threshold for a $\pi$) which deposits more than 2.8 GeV in the calorimeter;
- a slow hadron, accompanied by a photon and a noise signal from the Čerenkov;
- an electron pair, where one of the electrons escapes detection.

Without further event selection 610 triggers are generated during a burst (corresponding to 0.62% of the interactions in the target); only 23% of these triggers can be recorded on tape.

To make a second-level event selection, a comparison is made between the momentum and the energy of the triggering particle. At a later stage also a search for electron pairs is foreseen. Two ESOPs are used: one for the momentum calculation and the other to determine the energy deposit and to make the final comparison. The set-up is shown in Fig. 18. Note the asymmetry: ESOP2 can control the CAMAC branch via the branch mixer, ESOP1 cannot. ESOP2 can thus access the memory of ESOP1, but not vice versa. ESOP1 is interfaced to the RH read-out of the MWPC.

---

**Fig. 18** Integration of ESOPs (E1 and E2) in the data-acquisition system of NA11 ([from Damerell et al.](#))
ESOP1 calculates the momentum from data from 4 planes of MWPCs and from hodoscopes. These data are provided at a rate of 200 ns/word. The time required for this calculation is typically 400 μs. An event is vetoed when \( p \geq 12 \text{ GeV/c} \) or when \( p_T < 0.3 \text{ GeV/c} \). ESOP2 calculates the energy deposited from data input through the CAMAC branch: the cells hit in the calorimeter and the pulse amplitude for each cell. It also reads the momentum from the memory of ESOP1, when it is available, and makes the comparison |E - p| < c-E. Figure 19, from Lütjens²⁸, depicts the sequence of events. Note that the NORD-10 starts immediately to read the event, but that this read-out can be interrupted at any moment when one of the ESOPs emits a veto signal.

ESOP1 can veto an event for various reasons, other than the momentum criteria: no hit in front of the calorimeter, too many tracks, or ambiguous events. A fraction 0.42 of the processed triggers are accepted by ESOP1. Similarly, ESOP2 can veto on a number of conditions, besides the E versus p cut, such as hadron-like shower development, energy above 19 GeV, etc. ESOP2 accepts 0.59 of the events. The two in collaboration thus accept 0.42 \times 0.59 = 0.25 of the processed triggers. In summary the results are:

Without ESOPs:
- number triggers recorded/all triggers = 0.25
- triggers per burst written on tape = 140

With ESOPs:
- number of processed triggers/all triggers = 0.53
- triggers per burst written on tape = 84
The sensitivity of the experiment has been increased by $0.53/0.23 = 2.3$ with $84/140 = 0.6$ less data to be analysed.

It is interesting to see what experience has been gained with the use of these ESOPs in an experiment. It should be remarked in passing that this experiment is an illustration of the principles enumerated in Section 2: fast read-out of a subset of the data, early vetoing of an event, both reducing the dead-time. The dead-time reduction is further accentuated by adopting two ESOPs in a "multiprocessing" configuration. In spite of all these factors, the physicists remarked that an increase in the speed of ESOP would be useful. Another fact which became immediately apparent was that 256 micro-instructions (ESOP's original maximum capacity) was not enough. As a result ESOP was modified to accept 1K of microprogram memory. The effort to write the microprograms has not been evaluated exactly, but it certainly is of the order of several man months. A few suggestions emerged from this experience, aimed at making programming easier and also execution faster. They concern improvements of the hardware, such as an increase in the number of internal registers and in the number of loop counters, and the implementation of a subroutine return stack to enable nesting of subroutines. Other suggestions concerned more the software tools (e.g. avoid or detect instruction clashes) or the integration into the environment (improve the inter-processor communication). Most of these suggestions seem reasonable and designers of new equipment would be well advised to take notice of them.

Of the two other experiments which make use of ESOP, R807 selects events with high transverse momentum from drift-chamber data with a decision time of $\approx 150$ us, whereas, in EHS, ESOP has the task, among others, to reduce the masses of data provided by ISIS, a detector designed to measure ionization by multiple sampling along the particle track$^{3)}$.

4.3.2 CAB in an experiment

CAB is an improved version of $\omega$77. The device itself is briefly described in Barrelet et al.$^{24}$) and further in a number of theses and reports.$^{1-3}$). The improvements made to $\omega$77 were in line with the remarks made above: the program and data memory capacities have been increased, the micro-instruction widened from 16 to 24 bits and the cycle time slightly shortened. A remarkable characteristic of CAB is its interface to the general-purpose interface bus (GPIB, also known as the HP bus and as IEEE standard 488). This bus, widely used in laboratory instrumentation may reveal its importance also for high-energy physics as more and more desk-top calculators and computers are equipped with it.

CAB has been used in an experiment at the FS (S157) $^{4}$) to measure the total cross-section $\pi^p$ at 5-15 GeV/c with a high relative precision: $0.3\%$ in $\sigma_{tot}$ and $0.05\%$ in $p$. To attain this precision, the momentum range must be divided into 2000 intervals and $10^8$ scattered pions per interval must be collected. Since only 2% of the incident pions are scattered $10^{10}$ incident particles are to be studied. With classical means this would have meant 5-10 years of data taking, writing some 10,000 magnetic tapes, which would have required several months of 7600 CPU time to analyse.

Instead of following this tedious road, the experimenters took a very daring step: CAB was used to analyse the data in real time and to output only accumulated histograms. During data taking the role of the data-acquisition computer was limited to collecting these histograms, and to writing them to tape. In addition, a certain number of "pathological" events were recorded on tape for further checking.
In the experiment (see Fig. 20), the target is preceded by a focalizing spectrometer and three wire chambers $W_1$, $W_2$, and $W_3$. Two more wire chambers $WF_1$ and $WF_2$ are placed behind the target. Discrimination against muons is possible with the absorber and counters downstream. CAB is for this experiment equipped with nine "peripheral processors" for the read-out of the wire chambers. These chambers are small with a limited number of wires ($< 256$ in all cases). A hit in a wire plane is directly encoded using priority encoders and written by the "peripheral processor" to CAB's memory. One missing plane in the beam chambers is allowed, but is indicated with a status bit. This read-out of an event is performed simultaneously with the processing of the previous event, in a pipeline fashion. The synchronization of the read-out and processing with the primary trigger requires some additional fast logic. Once the data acquisition is performed CAB starts its analysis tasks: sampling of the incident pion beam (to determine the incident momentum), treatment of the scatter candidates, accumulation of histograms, and finally transmission of histograms and pathological events to the minicomputer. In order to perform all these tasks, the processing is divided into three cycles: a short, a medium and a long cycle. The number of events migrating from one cycle to the next decreases progressively.

In the short cycle, the incident beam is sampled: 1 trigger out of 12 (target full) or out of 20 (target empty) is transmitted to the medium cycle, after an additional check on the number of planes hit in the three beam chambers. For the other triggers the two projected scattering angles are calculated:

$$\theta_x = a_2x_2 + a_3x_3 + \alpha_{f_1}f_1$$

and similarly for $\theta_y$. If $|\theta_x|$ or $|\alpha_y|$ is less than a cut-off, processing is stopped and one proceeds to the next event. If not, one goes to the medium cycle. The short cycle consists of 22 instructions only (see Table 3) and is executed in 3.6 us. Twenty-three per cent of the events are rejected already after 1 us.

In the medium cycle high multiplicities are rejected and then $p$ is calculated for the events retained:

$$dp = a_4x_4 + a_5x_5 + a_6x_6 + a_7$$

In addition, a phase-space constraint is determined

$$dx = x_3 - a(dp)x_3 - b(dp)$$
and a check is made to see if the scattered particle is not a muon. The medium cycle takes 9 - 14 \( \mu \)s.

In the long cycle it is ascertained that the event represents a scattered pion, or if multiple hits exist a new combination is tried. Uncertain or particularly complex events are sent to the minicomputer. The accepted events are entered into histograms which will serve to determine \( d\phi/dt \) and \( \sigma_{\text{tot}}(dp/p) \). The long cycle needs 45 \( \mu \)s.

A tight dialogue is maintained with the minicomputer, for the transfer of strange events and, once every 30-50 bursts, for the transfer of the histograms. The minicomputer has also other, very important, tasks. The most significant are the calibration runs, serving to determine the constants used in the analysis by CAB and the display of the results. Also the deviations of a number of parameters from their average are constantly logged. This constitutes an efficient and rapid check on the experiment.

The mean time spent per event is finally 6 \( \mu \)s, so that in theory \( 1.7 \times 10^5 \) events/s could be handled. The measured rate was \( 1.53 \times 10^5 \) events/s. The off-line analysis is obviously greatly reduced and consists of a number of consistency checks on the data, on a further accumulation and saving of the histograms and on checks of the pathological events to ensure that they have the expected characteristics. The off-line analysis time is therefore reduced to 10 s of 7600 CPU for the processing of histograms and strange events resulting from \( 10^8 \) events handled by CAB. Without the real-time analysis, 50 magnetic tapes and 20 h of 7600 CPU time would have been needed.

In conclusion, the step taken to analyse in real-time was very daring, but successful. In order to be successful careful preparation, perfect calibration, and continuous monitoring are required. The immense benefit is immediate feedback, covering all experimental conditions.

4.3.3 MONIKI in a PETRA experiment

MONIKI\(^{1}\) was developed for the TASSO detector at PETRA, to recognize curved tracks in a cylindrical geometry. These tracks emerge from the beam line, which is the axis of the cylinder, and are curved by a solenoidal magnetic field. Figure 21a shows that the equation of a circle, tangential to the vertical axis, can be written in polar coordinates as:

\[
\rho = 2R \cos \theta = 2R \sin \phi ,
\]

\[
(11)
\]

![Fig. 21](image)

Fig. 21 Curved tracks in a solenoidal magnetic field, looking down the beam axis. a) A circle tangential to the vertical axis, expressed in polar coordinates. b) A track leaving at an arbitrary angle.
where φ and ρ are the polar coordinates and R is the radius of the circle. Normally a track starts with an initial angle φ and then crosses a number of cylindrical drift chambers, which determine φ and ρ for several points along the track. From Fig. 21b, it follows, considering the crossings at two successive chambers, that

\[ ρ_1 = 2R \sin φ_1 \]

and

\[ ρ_2 = 2R \sin(φ_1 + α), \quad \text{with} \quad α = φ_2 - φ_1. \]

This can be solved for φ₁ and R:

\[ φ_1 = \arccot \left[ \frac{\frac{2}{ρ_1} \sin α - \cot α}{\frac{2}{ρ_1} \sin α} \right] \tag{12} \]

\[ R = \frac{1}{2} \frac{ρ_1}{\sin φ_1}. \tag{13} \]

We are faced with the necessity to evaluate complex formulae, containing trigonometric functions. Obviously this evaluation has to be done within a tight time constraint\(^5\). As the read-out is 4-10 ms per event, MONIKA, in order to be effective, must reject an event well before. The decision time should be of the order of 1 ms, and since the events have a high multiplicity the allowed time per track is \(\leq 100\) µs. For a single track the calculation of (12) and (13) must be repeated \(8\) times in order to follow the track through \(9\) successive chamber planes.

MONIKA is naturally implemented in ECL, but this alone is not enough to achieve the required speed. The processor therefore makes extensive use of (i) look-up tables for evaluating trigonometric and other functions and (ii) content addressable memories (CAMs) to save time in picking out the next coordinate with which the track following should be pursued.

In the experiment MONIKA reads 150 words from time-to-digital converters (TDCs) and converts the data into ρ and φ (from the number of the drift wire) and Δφ (from the drift time). Some corrections are applied, before φ = Δφ (left-right ambiguity) are loaded into the CAMs. The track-following program, consisting of \(\approx 200\) micro-instructions can then start. Preliminary results\(^5\) indicate that

\[ φ_{\text{MONIKA}} = φ_{\text{TRUE}} \pm 5\text{ mrad} \]

and

\[ R_{\text{MONIKA}} = R_{\text{TRUE}} \pm 15\%. \]

Eighty per cent of the tracks found are correct.

### 4.4 Emulators

In contrast to the microprogrammable processors discussed so far, which have their own very low level instruction set, an emulator executes higher level code written for another machine. The hardware structure may be totally different from the emulated machine and the performance also, but from the software point of view there should be no difference if the
program is run on machine x or on its emulator. Emulation is made possible by microprogramming. It was thus possible for IBM to emulate the 1800 on the 360 series, which is microprogrammed, and thus ease conversion problems. In fact, all machines of the 370 series are emulators of a standard architecture, but implemented with widely varying hardware structures and performance. The same is true for the members of the PDP-11 family.

It is difficult to gather precise information on the difficulties and the effort to write microcode, but the consensus of opinion is that they are considerable. For an emulator a limited amount of microcode has to be written only once: the microprogram which interprets the machine code of the emulated processor and performs the appropriate operations on the hardware, so that an identical result is obtained as on the emulated machine. Once an expert has completed this task, any program can be transferred between the two machines, including -- in principle -- operating system routines or compilers and assemblers. The latter requires that the emulation is perfect and that the capacity of the emulator -- in memory and peripheral devices -- is sufficient. It is, however, not always necessary to build a large and expensive emulator so that it may match its large brother in all respects. The interest in small emulators stems from the consideration that one does not need all those disks and compilers. They exist already on the target machine and can be used there! Programs can be developed on a large configuration of the emulated processor and machine code produced which is then transferred to the emulator. The advantages of using a small emulator are thus several, besides being comparatively cheap:

- The learning effort for the user is minimal, as he is dealing with a machine already known to him, or at least with a good complement of documentation.
- Standard high-level languages can be used: FORTRAN, PASCAL, PL/I or any other language for which a compiler exists on the target machine.
- Good software development tools exist already on the emulated machine: editors, assemblers, compilers, linking loaders, spy and trace packages, task builders, etc.; these, together with file-handling systems, fast printers, and terminals, make the development of application software a real pleasure.
- Results on the emulator can be verified by comparing with the target; user confidence is thus greatly improved.

There is, of course, a price to be paid: emulators are usually slower than their parent machine. To attain the full speed of the parent the hardware must be extended and then also its price approaches that of the target processor.

When designing an emulator, one may be tempted to take some liberties and to deviate from the target architecture on other points than speed. Simplicity of design or the introduction of new, special features may motivate such deviations. The risk is great that they would become a permanent source of irritation to the user unless some conditions are satisfied. The results of the execution of an instruction on the parent machine and on the emulator must be strictly identical. Rounding of arithmetic operations, conditional branches, and the setting of condition codes are particularly sensitive areas. One may omit from the emulation a (small) subset of instructions, if they are not used in normal application programs; for instance, privileged instructions which can only be used by operating systems. At most one may go one step further and omit a few instructions which may be used in an application program, provided that their use is rare, easily detected, and easily avoided.
Table 5
Emulators

<table>
<thead>
<tr>
<th>Processor</th>
<th>DPNC 811</th>
<th>GA 103</th>
<th>MICE</th>
<th>POMA</th>
<th>168/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Institute</td>
<td>DPN Geneva</td>
<td>LAL Orsay</td>
<td>CERN Geneva</td>
<td>Courant, NYU New York</td>
<td>SLAC Stanford</td>
</tr>
<tr>
<td>References</td>
<td>47</td>
<td>48</td>
<td>49, 50</td>
<td>51</td>
<td>52, 55</td>
</tr>
<tr>
<td>Emulated machine</td>
<td>POP 11</td>
<td>POP 11</td>
<td>POP 11</td>
<td>CDC 6600</td>
<td>IBM 370/168</td>
</tr>
<tr>
<td>Technology/slice</td>
<td>6701 + 5001</td>
<td>2003</td>
<td>10800</td>
<td>ECL</td>
<td>2091</td>
</tr>
<tr>
<td>Microcycle (ns)</td>
<td>?</td>
<td>130</td>
<td>105</td>
<td>250 (RAM) **</td>
<td>250 (RAM) **</td>
</tr>
<tr>
<td>Micro-instruction (bits)</td>
<td>?</td>
<td>80</td>
<td>128</td>
<td>85</td>
<td>24</td>
</tr>
<tr>
<td>Microprogram mem. (words)</td>
<td>?</td>
<td>1K PROM</td>
<td>1K RAM</td>
<td>1K</td>
<td>m x 4K *)</td>
</tr>
<tr>
<td>Max. target mem. (words)</td>
<td>28K</td>
<td>64K</td>
<td>28K</td>
<td>?</td>
<td>m x 4K *)</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>-</td>
<td>-</td>
<td>Fix. pt. mult. 16 bit</td>
<td>18/60 bit int. 60 bit fl. pt.</td>
<td>16/32 bit int. 32/64 bit fl. pt.</td>
</tr>
<tr>
<td>Interfaces</td>
<td>-</td>
<td>-</td>
<td>ROM, ROMULUS, CAMAC, Unibus</td>
<td>to POP 11</td>
<td>to PUP 11</td>
</tr>
<tr>
<td>Speed relative to target</td>
<td>?</td>
<td>?</td>
<td>3 x POP 11/70</td>
<td>0.5-0.4</td>
<td>0.5-0.5</td>
</tr>
<tr>
<td>Relative speed (XUNZ) **</td>
<td>?</td>
<td>?</td>
<td>1.5-1.6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Remarks</td>
<td>Imperfect emulator</td>
<td>PLA decoder, user microprogram, interprets PASCAL P-code?</td>
<td>Emulates fix. pt. instr. only, user microprogrammable, special units</td>
<td>Target code is pre-translated into microcode</td>
<td></td>
</tr>
</tbody>
</table>

*) Max. 92 Kbytes equiv.
**) 50 ns(?) when fast RMs are used.

Given the advantages of emulation, it is not surprising that a number of emulators have been designed for use in high-energy physics. Table 5 summarizes the characteristics of those known to the author. One sees that the emulators can be divided into two categories:

- The "number crunchers", cheap substitutes for a large main-frame computer (168/E, POMA) and intended for data analysis.
- "On-line" processors, intended to enhance the capability of real-time event selection in an experiment (DPNC 811, MICE).

4.4.1 Example of a powerful emulator: 168/E

The 168/E **52-54**) was designed for off-line data analysis and it was therefore realized that floating-point arithmetic was essential. It is not a true emulator of the IBM 370/168 in the sense computer scientists attach to the concept. The processor cannot be loaded with 370 machine code; it would not know what to do with it. Instead, the 370 machine code is translated by a special program running on the IBM 370/168 into microcode for the 168/E. It is this microcode which, after being linked with library routines (Fig. 22) is loaded and executed in the 168/E. The data and the microcode are stored in separate memories. The translation program has therefore a little more to do than simply replace each instruction by a short sequence of micro-instructions: the memory layout must be re-arranged as well. As each machine instruction produces two or three micro-instructions, a program occupies more space in the 168/E than in the 370/168. It must be executed from fast memory,
lost the speed of the 168/E be seriously degraded. As fast memory is expensive, it is unavoidable to divide a large analysis program into several overlays, both for the program itself and for the data.

In the translator program for the 168/E all the 370 instructions used by the FORTRAN compiler are implemented. The processor can thus perform 16- and 32-bit integer arithmetic, and all the branching and memory addressing normally used. With its 150 ns cycle, it is 1.3-1.8 times slower than the 168. In addition, 32- and 48-bit floating-point arithmetic has been implemented. The 48-bit format is a cheap replacement of the 64-bit format which is implemented on the 370/168. Great care has been devoted to obtaining identical results within the precision imposed by the 48-bit format. The cycle for floating-point operations is 100-200 ns and typical execution times are:

ADD, SUB: 550 ns, CMP: 150 ns, MUL: 2-4 μs, DIV: 4-6 μs.

These times do not include the time for address calculations. For large FORTRAN programs, the overall speed of the 168/E is approximately 1/2 the speed of the 370/168.

A single memory board contains 16 Kbytes of data and 4K of micro-instruction words, which corresponds roughly to 500 lines of FORTRAN.

A number of 168/E's are at present being used at SLAC, where a large backlog of tapes from the LASS detector has been accumulated. These tapes contain $50 \times 10^6$ events; each event takes 0.5 s to analyse on a 370/168. The production program for this analysis consists of $\approx 20000$ lines of FORTRAN, or 100 Kbytes on the 370/168. This program has had to be split up into overlays. This has turned out to be rather easy for the program itself, with some duplication of routines. The time to overlay is $\leq 10\%$ of the execution time. To devise an overlay scheme for the data has turned out to be more complicated and software tools have
been needed to help in this task. The result is shown in Fig. 23. Nine overlays are needed to contain everything on 5 memory boards. Without overlays ≤ 250 Kbytes would have been needed. By the early summer of 1980, the results for one processor, in production since January were the following\(^{16}\): In 10 weeks, \(10^6\) events had been analysed, which is the equivalent of 220 h of CPU time on the 370/168. For various reasons the 168/E was running at only 1/3.5 times the speed of the 370/168 during this period. A comparison of the results on both machines, target and emulator, was done for a sample of 18,000 events, with 4.5 tracks/event on an average. It was found that 3 tracks were different in one coordinate. Further, the confidence levels for the track fitting never differed by more than \(10^{-6}\) and the track parameters never by more than \(10^{-5}\) (relative).

It is expected that by the end of 1980 six 168/E will be operating at SLAC in the configuration shown in Fig. 24. The IBM 370/168 is used to read the tapes, whereas the processing and the overlaying is under control of the POP-11/04. The 168/E are connected to two buses, one for data, the other for program. The overlays are contained in the two Mostek memories, to which the buses are connected via the "Bermuda triangles", which are three-sided interfaces, the third side being the POP-11 Unibus.

It is not surprising that the 168/E has raised great interest among high-energy physicists. Projects are being undertaken at CERN, DESY, Saclay, and other places, to implement 168/E and to use them for off-line analysis in a similar way as at SLAC, or even on-line.
The latter is not straightforward, as the 168/E has no I/O capability. Its memory must be filled from the outside and then the processor started.

4.4.2 An example of an emulator for on-line use: MICE

MICE\textsuperscript{8,10} emulates the PDP-11. It is not the first to do so: DPNC 811\textsuperscript{47} was used already a number of years ago in an experiment at CERN, as briefly described in Ref. 1. MICE was specifically designed for on-line, real-time use, and emphasis was therefore put on: speed, interfacing facilities and, naturally, ease of programming. On the other hand, it was considered that fast algorithms for event selection satisfy certain conditions and that therefore it was unnecessary to implement features such as floating-point arithmetic or memory management.

MICE has been implemented using the MECL 10800 family and to improve speed it uses a very horizontal microcode and an efficient pipeline scheme at the micro- and at the target-instruction level. The target pipeline scheme attempts to use every available cycle of the target memory (the memory containing PDP-11 code and data, mixed). Normally four things are happening simultaneously (see Fig. 25): execution of target instruction j; decoding of instruction j + 1 and fetching of the corresponding micro-instruction; fetching of target instruction j + 2; and preparation of the fetch address for instruction j + 3. The result is that PDP-11 code is executed on MICE three times faster than on a PDP-11/70 when the latter finds all instructions already in its cache memory.
MICE supports a number of interfaces to read-out systems, which all use DNA. Special hardwired functional units can also be attached to the processor buses (see Fig. 26). These units can be of the arithmetic type (fast multiplier, floating-point) or of the algorithmic type (track-finding, for instance). A CAMAC interface is used to control MICE and for loading of both target memory and writeable control store (WCS). All internal registers can be accessed via this interface. This is very important for debugging. A Unibus adapter is provided so that a -- somewhat simplified -- Unibus is available to attach simple peripherals. This Unibus can, for instance, be used to drive a CCl1 CAMAC interface and a CAMAC crate.

Fig. 26 Block diagram showing the different units composing MICE. WCS = writeable control store, TM = target memory, DAC = data acquisition computer [from Halatsis et al.].
MICE has been programmed in assembly, PSLl, and FORTRAN. It has two features, not existing on the PDP-11, which make it particularly adapted to fast processing:

- The "repeat" instruction, RPT n, causes the instruction following it to be executed n times, using a hardware counter. This instruction can be used to speed up -- by a factor of ≈ 2 -- the execution of short loops, or to vectorize code.

- The "jump to microcode" instruction allows to intermix direct execution of microcode with a normal program. This feature is important as it makes it possible to speed up critical parts of a program. When it turns out that a program spends 90% or more of its execution time in 10% or less of the code, these time-critical parts can be micro-coded. An over-all speed-up between two and four times may result, with a relatively small effort in writing the microcode.

It is at present too early to draw definite conclusions from these design choices of MICE; this would be better done when the processor has been used in a few experiments.

5. A TOTALLY DIFFERENT PHILOSOPHY

So far we have considered event selection and devices particularly adapted to this task. Event selection is, however, considered dangerous by a number of physicists. Rejected events are in fact irrecoverable. For exploratory experiments this is obviously unacceptable.

Another philosophy, particularly valid when one does not know beforehand what sort of phenomena one is going to encounter, consists of recording as many events as possible, using a minimal trigger. One then records a wide variety of events, with a minimum bias. During a preliminary off-line selection process, events of different types can then be distributed over a number of tapes which can be taken by participating physicists to their home institutes for analysis. Such a mode of operation is of interest when there are few experiments installed -- as at the CERN p+p collider for instance -- and the collaborations very large.

There is, however, a problem with the data storage. Normal magnetic tapes would be filled in a very short time and tens of thousands of them would be needed. Recording a video tape is at present the only possible solution, but in future video discs might become an attractive alternative. Table 6 compares characteristics of 6250 bpi magnetic tape with IVC video tape. One sees that the recording rate is not significantly higher, but that the capacity is enormously improved.

<table>
<thead>
<tr>
<th>Characteristics of 6250 bpi magnetic tape compared with IVC video tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>6250 bpi STC 1950</td>
</tr>
<tr>
<td>Density (bits/inch²)</td>
</tr>
<tr>
<td>Recording rate (bytes/second)</td>
</tr>
<tr>
<td>Capacity (bytes/reel)</td>
</tr>
<tr>
<td>Filling time/reel</td>
</tr>
</tbody>
</table>

*) For records of 16000 bytes.
6. USE OF FIXED INSTRUCTION-SET MICROPROCESSORS

The standard, single-chip microprocessors are generally considered too slow for effective event selection (the most recent ones possibly excepted). This does not mean that their use for this purpose is totally excluded and examples of the contrary do exist\textsuperscript{58,59}. Devices built around fixed instruction-set microprocessors are, however, widely used for other tasks, particularly for control and monitoring where speed is not the overriding parameter. One can roughly distinguish three types of devices.

In the first place there are the processors which play a role in the data acquisition. They are in general specialized and they work on the data stream, doing something to each data word as it passes through the processor at CAMAC speed. Examples are the processors developed for the ROMULUS system\textsuperscript{60}, which complete the data by adding crate/module numbers and word counts, thus forming a meaningful data stream, where only the coordinates of wires hit are retained. Examples of other tasks which can be performed by this sort of processor are pedestal subtraction for analogue-to-digital converters (ADCs) (the most sophisticated one subtracts a different pedestal for each ADC; the value of the individual pedestals being contained in a memory) and conversion of units (e.g. drift-wire numbers and drift time into coordinates). A popular microprocessor for this type of application is the Signetics 8X300. This is a bipolar 8-bit microprocessor, which is designed for data input, manipulation, and output within one 250 ns cycle. It has a limited instruction set, which is however particularly well adapted to the kind of operations needed. Data can, for instance, be read, rotated left or right, or ANDed with a mask and then merged with the data on the output bus and written into the destination in a single instruction. This microprocessor is at present used in ROMULUS processors (CERN)\textsuperscript{58}, in a drift chamber read-out system (FNAL)\textsuperscript{59} and in a 16-port CAMAC I/O module (Oak Ridge)\textsuperscript{59}.

The second type of microprocessor usage is in auxiliary crate controllers (ACCs). They provide processing power to a CAMAC crate and are thus particularly suited for applications where distributed intelligence is useful. An ACC can be an elegant solution for several control and monitoring functions, in an accelerator for instance\textsuperscript{61,62}. The auxiliary controller can work autonomously on data collection, process and digest these data, and only transmit summaries or abnormal conditions to the central control computer. Similarly, for control operations, the central control computer can be greatly relieved. Several of these ACCs are now commercially available.

Examples are, amongst others: ACC 2099 (from Société d'Electronique Nucléaire, based on TI 9900), the 3880 (Kinetic Systems, based on Intel 8080), the 3875 (Kinetic Systems, based on LSI-11), MIX 11 (Standard Engineering Corporation, based on LSI) and HPTEC (based on LSI-11). The last two are designed for use in the GEC system crate. For more details the reader is referred to the manufacturers' literature.

Then there is the third type not always clearly distinguishable from the ACCs, which is intended as an autonomous controller of a CAMAC crate or even branch. These processors, several of which are commercially available (for instance Interface Systems 888 based on the 8080 or SEN's STAC 2107 based on TI 9900) are particularly useful in test set-ups or even for control and data acquisition in small experiments\textsuperscript{63}. Because of their relatively low price, experimentalists may consider using several at the same time, e.g. to test a wire chamber with one, while another is used to check out a series of read-out modules, etc.
The most sophisticated instrument of this type is without doubt CAVIAR\textsuperscript{58, 65}. CAVIAR is housed in a single 19 in-crate, containing:
- a Motorola M6800 microprocessor,
- a 64 Kbyte memory (EPROM + RAM);
- a floating-point processor (AMD 9511);
- a small display (\(\geq 10 \times 10 \text{ cm}\)), with video signal available for duplication on a larger screen;
- 2 serial RS232C I/O ports, to connect to a terminal and to a serial computer port for instance;
- a GPIB (IEEE 488) interface;
- a complete CAMAC branch driver;
- an audio cassette interface.

Figs. 27-30 Examples of graphics output produced on CAVIAR \textsuperscript{58} [from Cittolin\textsuperscript{65}].
All these facilities would have had limited interest if CAVIAR did not have an excellent software system. It runs BAMBI, which is BASIC augmented with CAMAC facilities. The BAMBI code is pre-compiled into an intermediate language, which is in turn interpreted. To produce listings a de-compiler is needed, as the original source language is not conserved. CAVIAR software also comprises an editor and sophisticated graphics facilities. The reader may judge for himself from Figs. 27 to 30, which are all single-screen images produced on CAVIAR itself.

It is therefore not surprising that in a very short time CAVIAR has become very popular for test and development set-ups, for small scale data acquisition, for monitoring and control, and for on-line data sampling and processing.

7. PROGRESS SINCE 1978

Before concluding these lectures, let us summarize briefly the progress which has been made since the time when a similar overview of intelligent devices in high-energy physics was made

The number of papers and reports on the subject has clearly increased considerably: the previous review contained a bibliography of 36 references, covering fairly exhaustively the period 1974-1978. The present note has a bibliography with 87 entries.

Looking more closely, we see that considerably more experience has been gained with the use of look-up tables, where modular systems have now emerged. The other area where much more experience is available is in the use of microprogrammed processors, used now in a number of experiments. The hardwired processors, in spite of their potential for high speed, are still unpopular; possibly even more so now than two years ago. Important progress made is in the development of emulators. They are now becoming valid and valuable alternatives to the microprogrammable processors. During the last few years also some fine systems have been implemented using fixed instruction-set microprocessors. CAVIAR is an excellent example of this.

Lastly, the limitations of CAMAC have become more apparent. Its main defects are its low speed and the lack of reasonable inter-module and inter-crate communication

Systems such as ROMILUS have helped to overcome some of its awkward aspects, but clearly CAMAC cannot easily be developed further. There is thus a clear need for a new system which should combine high speed (for data acquisition) with good facilities for implementing distributed processing power. This system -- FASTBUS

8. POSSIBLE FUTURE DEVELOPMENTS

We may try to look ahead and speculate about possible future developments. Some of these are already under way and would have merited more than a very brief mention in the last minutes of these lectures. Lack of time -- and personal taste and bias of the speaker! -- have prevented this.

It is probably safe to predict that emulators will find wide application in the years to come. The first signs are already visible, as the success of the 168/E shows. Then,
for the microprogrammed processors, better software tools to produce and to test microprograms will probably be produced. They are certainly badly needed!

In the area of mass storage devices we may expect new applications in high-energy physics experiments. If they will be charged-coupled devices, bubble memories or devices based on video-recording techniques (video discs and video tapes) is difficult to say. Maybe all three. The new 16-bit and the coming 32-bit microprocessors will certainly change the picture considerably. Processors such as the Motorola M68000 or the Z8000 approach closely the classical minicomputer, both in speed and in concept. The M68000 is at present receiving much attention and a number of projects using this 16-bit microprocessor have been started.

We will probably see soon experiments starting to use FASTBUS for part of their data-acquisition equipment, particularly in those areas where event selection and filtering is performed and communication between processes becomes important.

Last but not least, the next years will see the development and the use of multimicroprocessor systems. In these lectures, some examples of multiprocessor systems have been given; in a sense all event-selection examples used at least two processors: the data-acquisition computer and some other, specialized device. All these systems were ad hoc, using their own conventions, protocols, development tools, etc. The multiprocessor systems now under development use a number -- at least two! -- of identical processors, with some distribution of tasks between them. A multiprocessor system is obviously a possible solution to overcome speed limitations, using mainly commercial products. It is therefore an attractive scheme as it can be user-friendly for the development of application software, just as much as the emulators. At least two multimicroprocessor systems are under development at present.\(^{70,71}\)

The first system uses the Motorola M68000 processor and should therefore be powerful, the second intends to use NOVAs, at least in a prototype implementation. Multimicroprocessor systems would have merited an in-depth treatment, as they present many interesting aspects and problems. They are a subject of intense study, also in fields other than high-energy physics.

Many things can still be done to improve event selection and other uses of intelligent devices in high-energy physics experiments. It will be interesting to see in a few years how much the scene has changed.

Acknowledgements

The material for these lectures has come from the work of many people. I am grateful to all colleagues who have so contributed to these lectures. I hope to have given due credit to all; any errors or omissions are my responsibility, not theirs. I would particularly like to thank those colleagues -- too many to name -- with whom I have had the pleasure to discuss the subject of these lectures.
REFERENCES


4) R. Dubé, Thèse de doctorat de 3ème cycle, Univ. de Paris-Sud, Orsay, LAL-79/6 (mars 1979).


7) I. Gjerpe, A fast filter processor as a part of the trigger logic in an elastic scattering experiment, presented at Europhysics Conf. on Computing in High Energy and Nuclear Physics, Bologna (1980).


12) C. Boulin, Contribution à l'étude et à la réalisation d'un processeur microprogrammable ultra-rapide, destiné à la gestion des taches CAMAC, Thèse de docteur de spécialité, Strasbourg, CRN/HE 76-20 (1976).


25) A. van Dam, Microprogramming and bit-slice technology, these proceedings.

26) C. Halatsis, Software tools for microprocessor based systems, these proceedings.


36) J. Joosten, private communication (1980).


38) D.A. Jacobs, Applications of ESOP, a fast microprogrammable processor, in high-energy physics experiments at CERN, CERN DD/80/22 (1980), presented at Europhysics Conf. on Computing in High-Energy and Nuclear Physics, Bologna, 1980.


40) G. Lütjens, private communication (1980).

41) B. Habiballah, Contrôleur de châssis CAMAC programmable pour acquisition de données à grand débit, thèse de 3ème cycle, LPNHE/X (1979).


44) D. Lelouch, Mesure de la section efficace totale $\pi^-$ proton de 5 à 15 GeV/c, thèse de doctorat 3ème cycle, LPNHE/X/79.


46) P.F. Kunz, private communication (1980).


61) R. Rausch, Auxiliary CAMAC crate controllers using a 16-bit microprocessor and applications in accelerator control, presented at ESONE and European CAMAC Association meeting at DESY, Hamburg, 1978.


64) S. Cittolin and B.G. Taylor, CAVIAR: Interactive microcomputer control and monitoring of multi-crate CAMAC systems, presented at Joint Conf. on Microprocessors in Automation and Communications, Univ. of Kent, 1978.

65) S. Cittolin, Tasting CAVIAR: The general-purpose microcomputer at work, presented at Europhysics Conf. on Computing in High-Energy and Nuclear Physics, Bologna, September 1980.

66) R.W. Dobinson, Practical data acquisition problems in large high-energy physics experiments, these Proceedings.


* * *

BIBLIOGRAPHY

(Additional reports not referenced in the text)


