Nuclear Instruments and Methods 178 (1980) 571–579
© North-Holland Publishing Company

AN ON-LINE TRACK FOLLOWING MICROPROCESSOR FOR THE PETRA EXPERIMENT TASSO

P. SCHILDT, H.-J. STUCKENBERG
Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

and

N. WERMES
University of Bonn, Germany

Received 21 July 1980

Storage ring experiments at PETRA use large cylindrical detectors with huge numbers of electronic channels. Running at event-input-rates of about 0.5 MHz effective preprocessing of the data becomes relevant.

We report on a programmable microprocessor used in the trigger system of the TASSO detector. An event recognition is performed within a reconstruction time of 1 ms using fast ECL-technology together with associative memories and table-look up. For input and output of the system CAMAC is used, the speed is ensured by the microcomputer. An outline of the device and the microprogrammed algorithm is given.

The microprogrammable on-line track analyzer MONICA is the first running free programmable on-line track following microprocessor used in storage ring experiments.

1. Introduction

New sources of high energy particles and much better detectors offer a possibility of high data acquisition rates. Detectors triggered by fast electronics enable the physicists to take much more interesting events than before. But due to the short decision time too many background events pass the trigger. An efficient selection can only be made if the topology of the event is known, i.e. if data reduction is done by recognizing the track pattern on-line. One has a very efficient filter if one can calculate the coordinates of particle tracks within the readout time of an event, i.e. within one or a few milliseconds. This

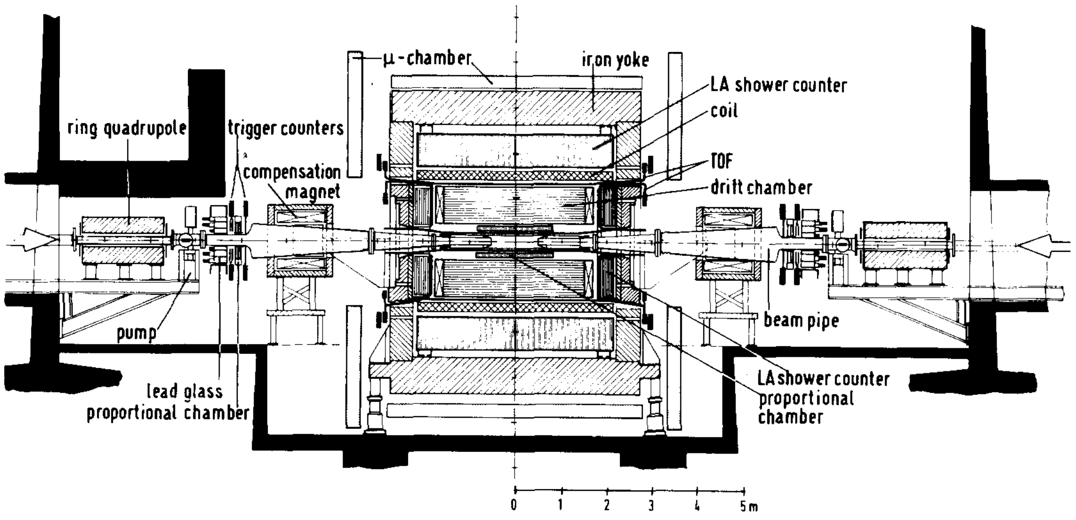


Fig. 1. Cut along beam (TASSO).

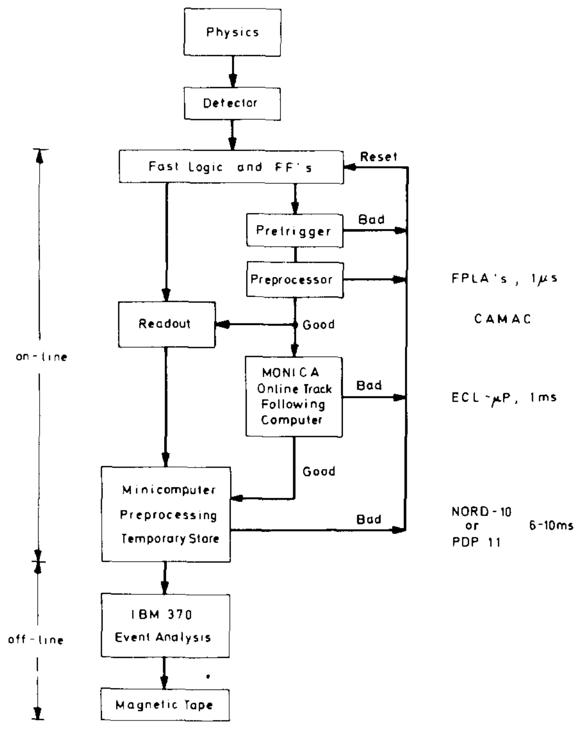


Fig. 2. Typical readout of a DESY-PETRA experiment.

can be achieved by a fast special purpose processor performing the pattern recognition task in storage ring experiments. The use of special purpose processors in high energy physics experiments has been summarized by Verkerk [1].

The PETRA storage ring detector TASSO is described in detail in ref. 2. A main feature of the detector is a large multiplane drift chamber [3], which has nine cylindrical signal surfaces measuring R and ϕ and six hyperbolical surfaces with $\pm 3^{\circ}$ stereo angles to measure the Z-coordinate for complete spatial reconstruction. Fig. 1 shows a cut along beam view of TASSO. The detector measures particles coming from e^+e^- collisions, but there exist also much background from cosmic rays and other high or low energy noise sources. The trigger rate in a PETRA detector can be up to 5×10^5 per second. Interesting events arrive with up to 10 per second, therefore one needs a reduction of some 10^4. This is done by preprocessing in two steps (fig. 2).

A preprocessor [2] is organized as a mask library stored in FPLAs, a mask means an angular range of some degrees seen from the interaction point into the R,ϕ-plane (fig. 3).

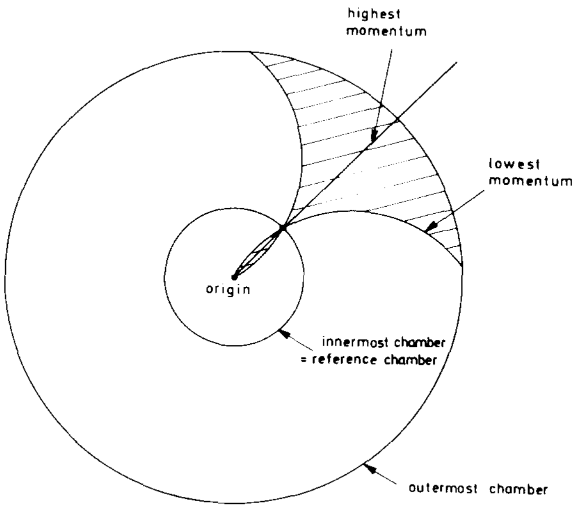


Fig. 3. Area of masks related to a reference chamber mask.

The track within a mask must begin at the interaction point and must have coincidence signals in a predetermined number of chambers. The mask description follows some rules about the minimum transversal momentum as seen in the figure. The number of tracks must be within two predefined limits. All possible events are checked, only good events lead to a preprocessor output, which starts the readout of all detector data. Together with this readout a special track following computer goes into operation. It calculates the tracks very fast and transmits track parameters to the readout if the event is found to be background of cosmic rays or other sources. If not, the event is marked "good" and passed to the experiment's computer. Thus one has a powerful tool to separate events and background on-line. The track following computer is named MONICA, which means a microprogrammed on-line track analyzer.

MONICA calculates track coordinates and momentum of particles using the signals from nine

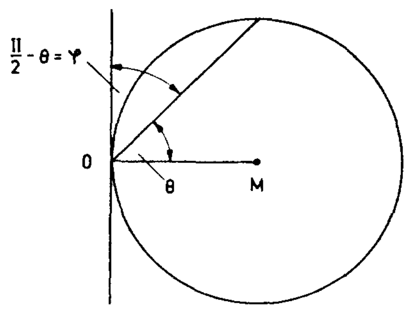


Fig. 4. Track definition in the R,ϕ-plane.

chambers, i.e. the wire number within the chamber, which gives an R and ϕ value together with the drift-time which is converted into a $\Delta\phi$ value. The problem of finding tracks is solved in two dimensions only, using the R values as the distance from the interaction point and the ϕ values which are angles between 0 and 2π. Within the detector a homogeneous magnetic field is vertical to the R,ϕ-plane, so that all tracks of charged particles are circular sections.

To describe very briefly the track finding algorithm, we start with the equation of the circle which is (see fig. 4)

$$\rho^2 - 2\rho R \cos\theta = 0 \ ,$$

or

$$\rho = 2R\cos\theta = 2R\sin\phi \ , \quad \phi = \tfrac{1}{2}\pi - \theta \ ,$$

if the origin O is on the circle and its tangent is vertical to line OM. The difficulty is that before calculation starts neither R nor the tangent are known. First we have to calculate ϕ, then the radius R. But in any case we need one point in chamber 1, one point in chamber 2 and together with the origin we have 3 points to define a circle segment (see fig. 5).

We calculate ϕ_1 the direction of the tangent. If we assume that points 1 and 2 together with the origin are on the same circle with the radius R_{12} then we calculate the tangent with

$$\phi_1 = l_1/\rho_1 \ \text{to} \ \phi_1 = \text{arc}\cot\left(\frac{\rho_2}{\rho_1 \sin\alpha} - \cot\alpha\right) ,$$

and the radius of the track to

$$R_{12} = \frac{1}{2}\frac{\rho_1}{\sin\phi_1} \ .$$

We proceed in the same way in the next chambers.

With each ϕ-value in chamber 1, which we call a point, we try the following (fig. 6).

With the assumption that this point belongs to a track coming from the interaction point we look for a

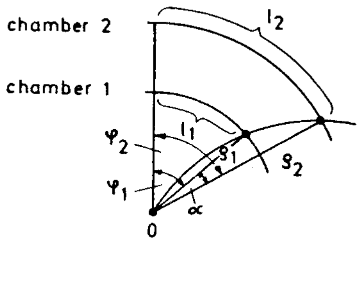


Fig. 5. Tangent calculation.

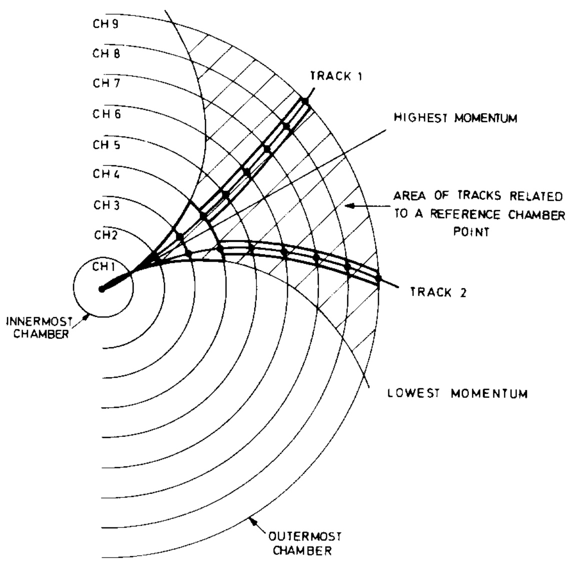
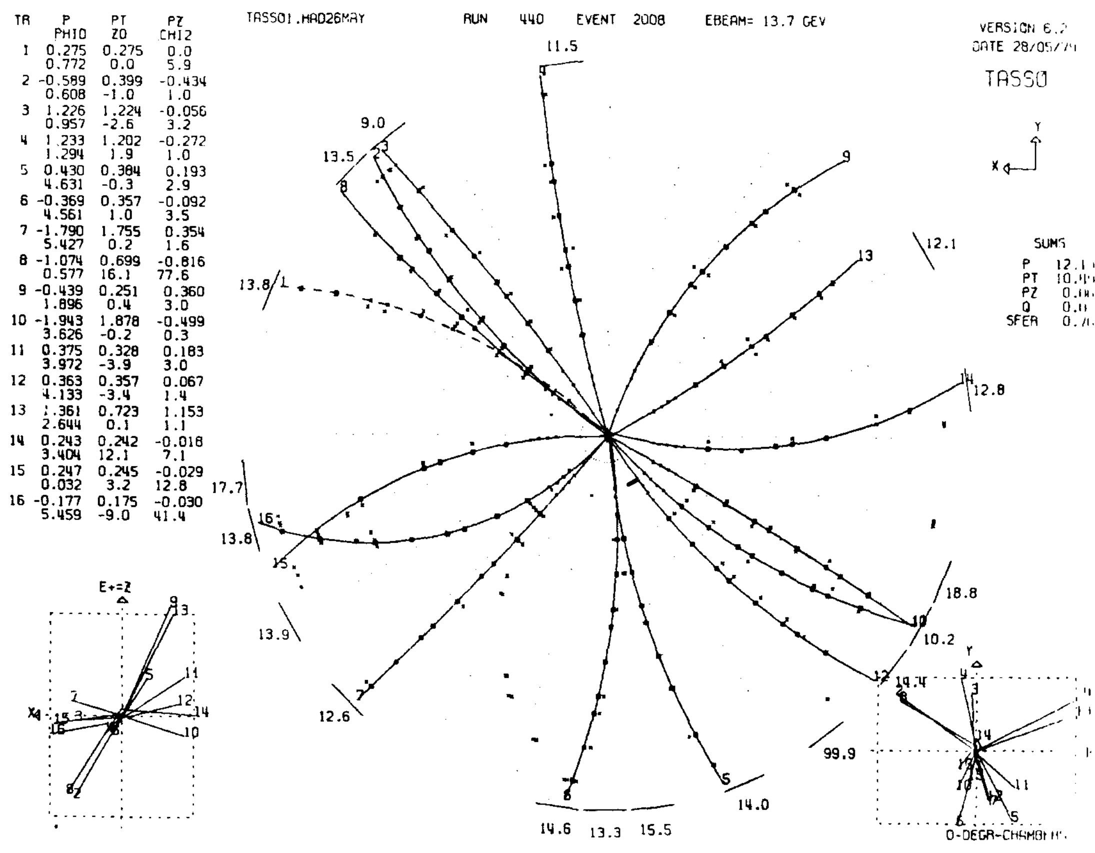


Fig. 6. Track finding method.

dedicated point in chamber 2 by defining an area, which is limited by the minimum transversal momentum as mentioned before. So only a few points of chamber 2 are in this area. With the first one we assume that the interaction point, point 1, and point 2 are on a circle and compute R and ϕ with the algorithm described before. Then we define the range of points in chamber 3, in which the next point should be. If we have such a point, we calculate the new ϕ-value and track radius. In chamber 4 we compute tangent and radius for the last time. Then we define a curved tube of some mm width beginning in chamber 4, ending in chamber 9 and look for points in it belonging to the same track radius. If the search was successful, a track is found and the used points will be cleared. We need at least 6 of 9 points to find a track. If there are less than 6 points, the program goes back to chamber 3 and the same calculation is started with the next point. If there is no other point, the program jumps to chamber 2 and looks for the next point there. If this also fails, the program starts with the next point in the reference chamber.

If MONICA finds out that the decision of the preprocessor was wrong, MONICA stops the event readout immediately, which goes from the detector electronics to the on-line computer in about 10 ms. So it was decided that MONICA's runtime should be limited to 1 ms.

The reconstruction time per track for a typical



```
TASSO1.HAD26MAY          RUN   440     EVENT  2008     EBEAM= 13.7 GEV

TR     P       PT       PZ
      PHI0     Z0      CHI2
 1   0.275   0.275    0.0
     0.772   0.0      5.9
 2  -0.589   0.399   -0.434
     0.608  -1.0      1.0
 3   1.226   1.224   -0.056
     0.957  -2.6      3.2
 4   1.233   1.202   -0.272
     1.294   1.9      1.0
 5   0.430   0.384    0.193
     4.631  -0.3      2.9
 6  -0.369   0.357   -0.092
     4.561   1.0      3.5
 7  -1.790   1.755    0.354
     5.427   0.2      1.6
 8  -1.074   0.699   -0.816
     0.577  16.1     77.6
 9  -0.439   0.251    0.360
     1.896   0.4      3.0
10  -1.943   1.878   -0.499
     3.626  -0.2      0.3
11   0.375   0.328    0.183
     3.972  -3.9      3.0
12   0.363   0.357    0.067
     4.133  -3.4      1.4
13   1.361   0.723    1.153
     2.644   0.1      1.1
14   0.243   0.242   -0.018
     3.404  12.1      7.1
15   0.247   0.245   -0.029
     0.032   3.2     12.8
16  -0.177   0.175   -0.030
     5.459  -9.0     41.4
```

Fig. 7. Typical event in the R,ϕ-plane.

event with 10 or more tracks (fig. 7) is limited to 25 μs taking into account many trials combining driftchamber hits.

The used algorithm shows that the processor has to calculate several trigonometric functions like sin, cos, ctg etc., and the execution time needed for these calculations are rather long if they are done by software.

Furthermore the processor has to do also multiplications and divisions, each of them consuming some microseconds. Only from this point of view one would need a millisecond for computing one track instead of the required 25 μs.

To complicate the situation much more, we have to take into account that if the processor has calculated a ϕ-value or a range of values within the next chamber, and the processor looks into the memory asking for stored ϕ-values in this range, all memory cells are to be addressed sequentially to get the answer by read and compare. In FORTRAN software this is done by time consuming DO-loops. So if the track following program is written in FORTRAN, the execution time for one track is in the order of 10–20 ms which is a factor of about 1000 too long.

We can solve this problem by taking a fast special processor using several tricks. Fast means fast hardware together with fast software.

The first trick is to use CAMs (content addressable memories) to store the ϕ-values of the driftchamber hits (fig. 8). A CAM element is addressed by associating input data with some or all bits of the stored words. This approach differs from the conventional memory organisation where an arbitrary number which has no relation to the stored data is assigned to each address location. Because of its organisation the CAM is uniquely suited to do parallel searches by data association. A search can be done on an entire data word (full association) or on special fields by masking data columns (masked association).

The used CAM elements consist of 16 identical memory cells together with the necessary addressing and mode control logic to perform several modes of

INPUT I	CONTENT Q	MATCH M
0	0	1
0	1	0
1	0	0
1	1	1

FULLY ASSOCIATED: $M = I_{ij}$ XOR Q_{ij}

MASKED ASSOCIATED: $M = I_i$ XOR Q_i
OR
$M = I_j$ XOR Q_j

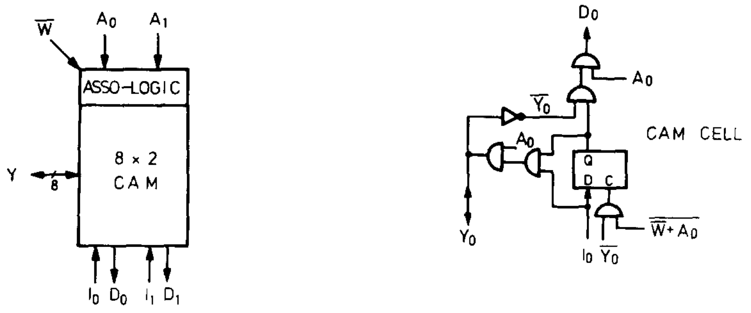
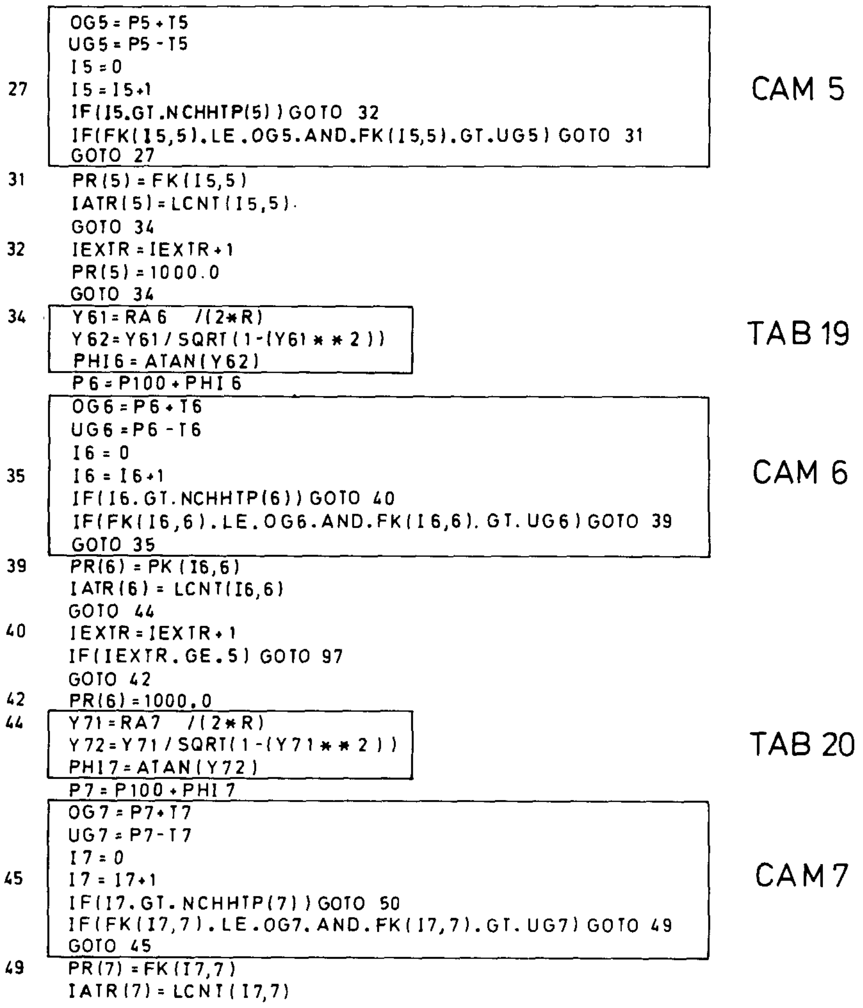


Fig. 8. CAM: Signetics 10155.



CAM 5
TAB 19
CAM 6
TAB 20
CAM 7

```
      OG5 = P5 + T5
      UG5 = P5 - T5
      I5 = 0
27    I5 = I5+1
      IF(I5.GT.NCHHTP(5))GOTO 32
      IF(FK(I5,5).LE.OG5.AND.FK(I5,5).GT.UG5) GOTO 31
      GOTO 27
31    PR(5) = FK(I5,5)
      IATR(5) = LCNT(I5,5)
      GOTO 34
32    IEXTR = IEXTR+1
      PR(5) = 1000.0
      GOTO 34
34    Y61 = RA6   /(2*R)
      Y62 = Y61 / SQRT(1-(Y61**2))
      PHI6 = ATAN(Y62)
      P6 = P100+PHI6
      OG6 = P6+T6
      UG6 = P6 - T6
      I6 = 0
35    I6 = I6+1
      IF(I6.GT.NCHHTP(6))GOTO 40
      IF(FK(I6,6).LE.OG6.AND.FK(I6,6).GT.UG6)GOTO 39
      GOTO 35
39    PR(6) = PK (I6,6)
      IATR(6) = LCNT(I6,6)
      GOTO 44
40    IEXTR = IEXTR+1
      IF(IEXTR.GE.5) GOTO 97
      GOTO 42
42    PR(6) = 1000.0
44    Y71 = RA7   /(2*R)
      Y72 = Y71 / SQRT(1-(Y71**2))
      PHI7 = ATAN(Y72)
      P7 = P100+PHI7
      OG7 = P7+T7
      UG7 = P7-T7
      I7 = 0
45    I7 = I7+1
      IF(I7.GT.NCHHTP(7))GOTO 50
      IF(FK(I7,7).LE.OG7.AND.FK(I7,7).GT.UG7) GOTO 49
      GOTO 45
49    PR(7) = FK(I7,7)
      IATR(7) = LCNT(I7,7)
```

Fig. 9. Printout of the FORTRAN algorithm.

operation which are associate, masked associate, read and write. In the associate or masked associate mode the information presented to the data input, which in our case is a ϕ value or range of ϕ values calculated by MONICA with respect to the next chamber, is compared simultaneously with the stored data words which consist of all ϕ values measured by the TDCs dedicated to the chamber. If any or all of the stored data words match the information on the data inputs the corresponding output line is affected.

The great advantage using CAMs for our problem is to compare calculated and measured ϕ values within one computer cycle. If we use a normal RAM, all measured ϕ values have to be searched sequentially to find out whether the input and the stored word are the same. The time for comparing is greatly reduced in our case.

The second trick to reduce the program run time is the use of look-up tables in which complex calculations like combinations of angular functions, multiplications and divisions are stored, which in our case are functions of one variable only. The use of these tables is possible because the number range of the used data is limited. All ϕ values are between 0 and 2π, they are limited to a spatial resolution of about 0.2 mm, that means we have some 15 to 16 bits of data word length. Also the angular range is limited because of the minimum transverse momentum of the particles. So the processor has to add or subtract only, all other arithmetic functions are stored in different tables with more than 120 kbytes. Each of the precalculated values can be accessed within one computer cycle. This is also a great reduction in runtime. In fig. 9 one can see a section of the FORTRAN printout each framed part of which are the contents of a table or a CAM access. The tables are computed on a large computer, the results are burned into PROMs.

2. Hardware

Since two years ECL processors are available, built in 4-bit wide slices which can be cascaded to n times 4-bit processors. The processors are microprogrammed machines with cycle time of less than 100 ns. Therefore we have designed a 16-bit fixed point processor with access to CAMs, tables and other peripherals, which can execute a program with up to 250 microinstructions within 25 μs. This was the runtime limit for the one track program. We have translated the resulting FORTRAN program into something more than 200 microinstructions including the CAM and table calls.

Basically, a microprogrammable device is a machine in which a sequence of microinstructions is used to execute various commands required by the machine.

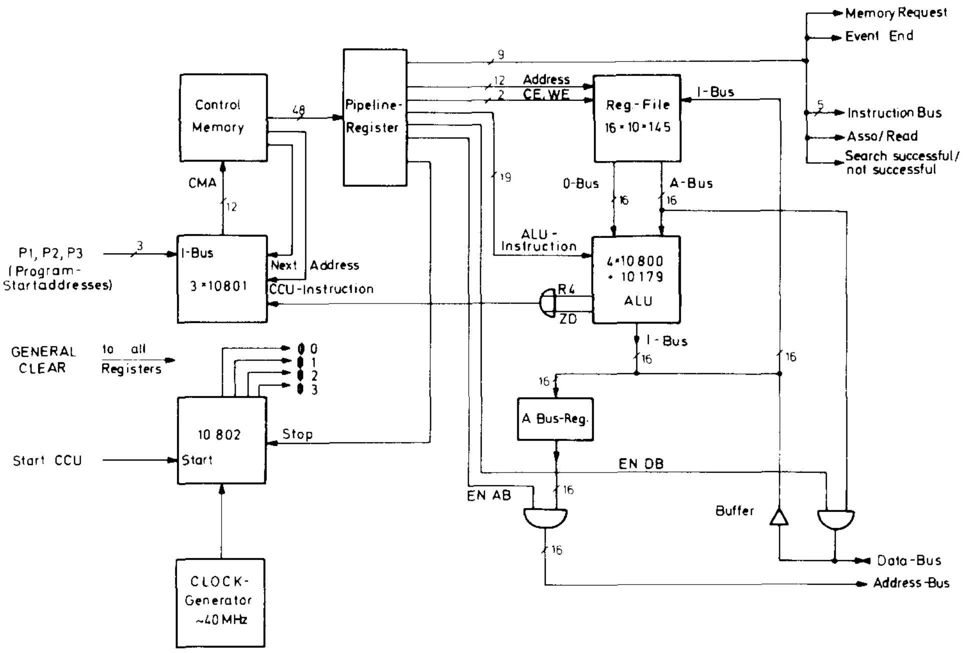


Fig. 10. Computer block diagram.

A microinstruction has two primary parts:

(1) the definition and control of all microoperations to be carried out within one computer cycle, and

(2) the definition and control of the address of the next microinstruction to be executed.

The microprogram memory holds the various microinstructions. Each instruction word is broken into various fields defined by the configuration of the machine; the fields can consist of different numbers of bits. Fields are named e.g. branch address, next address control, carry control, ALU source operand control, ALU function control, ALU destination control etc. The microprogram format has to be defined by the user of the special purpose processor, i.e. defined by the special application.

In our case (fig. 10) the computer hardware is build up as a 16-bit ALU, a register file of 2 X 32 words, a microprogam sequencer delivering a 12-bit address to the control memory the size of which is 1k X 72 bits [4]. There are 3 microprograms for different track following algorithms, each containing some 200 microinstructions. The clock is generated with a cycle time of 100 ns. The computer is housed in a CAMAC crate using a special dataway. It is connected to the outside world through CAMAC highways which are not too slow because the incoming data, organized as branch-, crate-, module- and subaddress numbers, have to be converted into useful numbers, namely R and ϕ-values. Therefore during the fetch of the next data word from the TDCs the conversion is done by the processor. For a more detailed description of the hardware see ref. 5.

3. Software

In our case writing software means that a microprogram for a bit-slice processor has to be developed. The configuration of a bit-slice hardware depends on its application, i.e. it is impossible to write a general assembler, editor and simulator for all the different configurations. On the other side it is a very hard job to develop and debug a microprogram of more than hundred instructions without any software support. In order to realize a more convenient method to program the processor one has to develop a microassembler that runs on a standard computer, in our case an AEG 80-60. The microprogram for the processor is written in two steps. First the mnemonic code for the microword fields has to be defined, symbols describing the functions, operand sources, registers etc. This set of symbols has to be modified if another processor configuration is used. In the second step one writes the microprogramm, i.e. the tracking algorithm in terms of these mnemonics and runs the assembler program on it. The assembler produces a binary pattern which is the object code for the bit slice processor.

Therefore development and syntax debugging can be done using the full power of a big computer and one avoids to produce the object code manually which is very tedious and difficult to survey.

However, to fully test the microprocessor hardware and software, one needs an additional tool to provide functional debugging. One would like to communicate with single parts of the whole processor equipment to test its functional behaviour by the aid and the comfort of a big computer. In order to realize this an interface between the computer and the microprocessor has to provide the communication. This interface between MONICA and the AEG 80-60 is a serial port working as a PROM simulator (fig. 11). The microprogram object code is stored during the test phase on a disc of the 80-60. To run the program the 80-60 loads the first microword serially into the PROM simulator which transfers the information to MONICA in a paralle mode. MONICA performs the required operation and in return sends the next

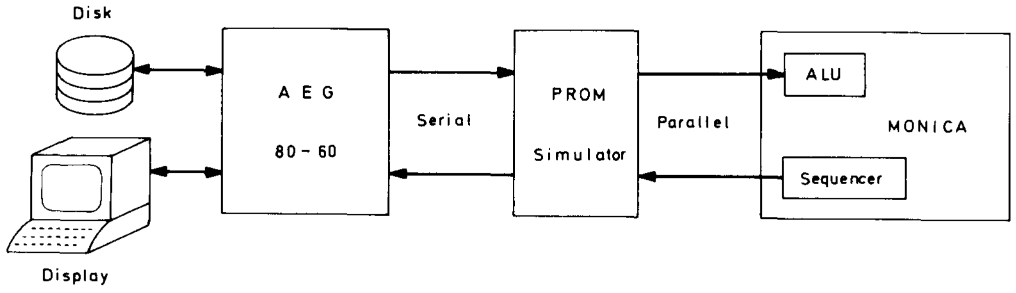
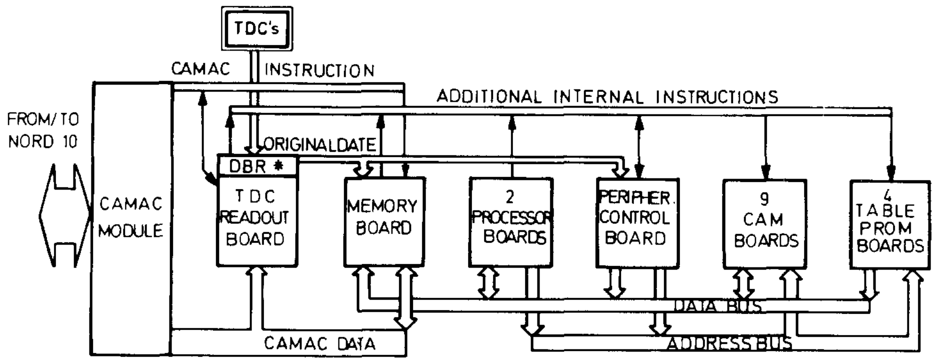


Fig. 11. Software development with a PROM simulator.



* DBR = DATA BUFFER REGISTER ON THE TDC READOUT BOARD

Fig. 12. MONICA block diagram.

address from the microprogram sequencer through the output bus to the PROM simulator and thereby to the 80-60. Now the next microinstruction is fetched and the procedure will be repeated. In addition a special debug program, written in FORTRAN IV, provides the possibility to test specific parts of MONICA by setting up small sequences of microinstructions. E.g. one can test whether a CAM is read out correctly when associating with a special data word and whether the ALU operates reasonably well.

Also the tracking algorithm can be tested by the aid of this debug system. Breakpoints and the possibility to receive all register contents at any program point allow efficient checks for software bugs.

The PROM simulator module is designed for different processors having an instruction word length of up to 96 bits. All the serial to parallel conversion circuitry is built in using the right protocol.

4. Results

Summarizing all features we are lead to the block diagram in fig. 12. In a typical event MONICA reads up to 150 words from the TDCs, converts the addresses into R and ϕ values, the drift time into $\Delta\phi$ values, accomplishes corrections due to individual TDC slopes and offsets as well as to gas dependent drift velocities and loads the resuling $(\phi \pm \Delta\phi)$ values into the CAMs. After all the processor starts the program with about 200 instructions and when having finished it sends an interrupt to the experiment's on-line computer (NORD 10).

Tests on MONICA's software were carried out with hit patterns from Monte Carlo and real PETRA events.

Fig. 13 shows $\phi_{\text{true}} - \phi_{\text{MONICA}}$, the difference of the track tangents yielding a halfwidth of 10 mrad. Fig. 14 displays the radius deviations

$$\frac{R_{\text{true}} - R_{\text{MONICA}}}{R_{\text{true}}} ,$$

showing a halfwidth of 10–15%.

The tests have shown that nearly 80% of the tracks are correctly reconstructed, up to 14% are wrong and up to 6% are not found. This is due to the fact that very often tracks are within jets, i.e. they are very close together.

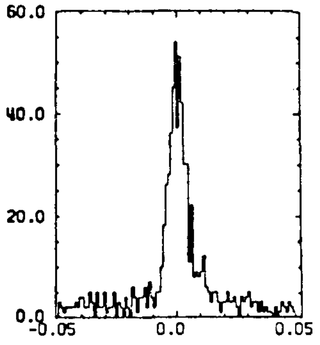


Fig. 13. $\phi_{\text{true}} - \phi_{\text{MONICA}}$.

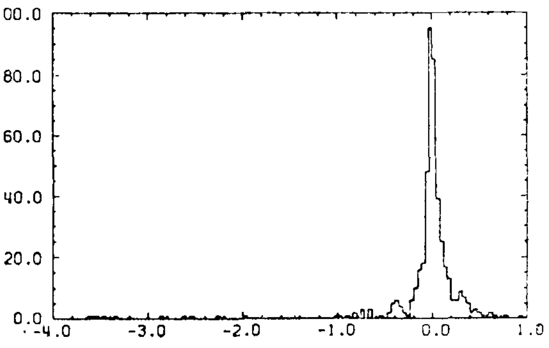


Fig. 14. $(R_{\text{true}} - R_{\text{MONICA}})/R_{\text{true}}$.

At present MONICA is built in the experiment's on-line data acquisition. The device is still in a test status in order to optimize the tracking efficiency. We expect to use MONICA as an event filter for the experiment which allows to gain additional interesting physics in the near future.

References

[1] C. Verkerk, Special purpose processor, Proc. 1974 CERN School on Computers, CERN 74-27.
[2] TASSO Collaboration; R. Brandelik et al., Phys. Rev. Lett. 83B (1979) 261.
[3] H. Boerner et al., The large cylindrical drift chamber of TASSO, DESY 80/27 (March 1980); submitted to Proc. Wire chamber Conf. Vienna, Austria (1978).
[4] Data sheets Motorola 10800 series (1978).
[5] P. Schildt, Diplomarbeit (Universität Hamburg, 1979).