EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN-EP/83-137
12 September 1983

A THREE-DIMENSIONAL RIPPLE TRIGGER FOR THE DELPHI
TIME PROJECTION CHAMBER

G. Darbo
INFN, Genoa, Italy

B. Heck and L. Rossi^{*)}
CERN, Geneva, Switzerland

Paper presented at the
Time Projection Chamber Workshop,
Vancouver, B.C., Canada, 23-25 June 1983.

^{*)} Permanent address: INFN, Genoa, Italy.

A THREE-DIMENSIONAL RIPPLE TRIGGER FOR THE DELPHI
TIME PROJECTION CHAMBER

G. Darbo
INFN, Genoa, Italy

B. Heck and L. Rossi^{*)}
CERN, Geneva, Switzerland

(Presented by L. Rossi)

1. INTRODUCTION

The use of a contiguity processor¹ for fast selection of events in the DELPHI detector has already been proposed². This note presents a general description of a contiguity processor and its implementation in gate arrays.

A three-dimensional ripple trigger for the time projection chamber (TPC) and a fast "shower builder" processor for the high-density projection chamber (HPC) both based on contiguity ideas, are presented.

2. CONTIGUITY CONCEPT

In the visual perception, topological analysis and metric evaluation are separated³. The first is performed by the system "eye-cerebral cortex", while the second is performed by the brain.

A similar separation is advantageous also in electronic processors devoted to automatic pattern recognition. The topological analysis (i.e. finding a track) can be made in a highly parallel structure, such as a contiguity mask processor¹, and will end up with clusters of data (i.e. sets of points belonging to a track).

The parametric calculations (i.e. finding the equation of the track) can then be done by a more standard processor that will use the clusters of data as input. Several advantages are evident:

- a) the topological analysis can be highly parallel and very fast since contiguity is a "local" property (i.e. no need for long loops);
- b) the parametric calculations are greatly simplified by the fact that the information has been previously ordered (i.e. one has to calculate the track equation *knowing* which are the points belonging to the track);
- c) the parametric calculations can be done by a multiprocessor structure (i.e. n clusters can be given to n processors).

The topological analysis based on contiguity criteria can be applied to a wide variety of cases.

There are, however, some general characteristics of the pattern under analysis that make it more suitable for this treatment:

- i) The information (lines, clusters, contours, etc.) should be contiguous (the contiguity can be defined mathematically¹ -- in this paper we shall deal with the intuitive concept). Provided the

^{*)} Permanent address: INFN, Genoa, Italy.

contiguity criterion is satisfied, the parametrization of the lines (clusters, contours, etc.) can be any.

- ii) The number of pixels of the image can be high. In fact a line search based on local properties of the image is performed in a time linearly depending on the number of pixels (N); different approaches for line searches^{4,5} need a time:

$$t \propto N^m \quad \text{with} \quad m \geq 2 .$$

3. THE TWO-DIMENSIONAL CASE

3.1 Generalities

Let us first present a simple example. We consider a TPC and we imagine picking up the signal from the pads (or group of pads) integrating along the full drift length. We will then have only the $\rho\phi$ projection of the tracks with a given granularity (see Fig. 1). The event can then be represented in a $m \times n$ matrix (m = detector layers along ρ , n = number of binning along ϕ), where a track will touch contiguous nodes. Let us now consider only a part of the event (see Fig. 2) with a track plus some uncorrelated noise. The big dots represent the hits memorized in a matrix node, where the switch -- if closed -- represents a connection that is established around each hit node. The contiguity mask that generates such connections is programmable (some possible choices are shown in Fig. 2); it can be different in different regions of the matrix and it can be changed in between two events, if necessary.

With an appropriate contiguity mask, one can take inefficiencies into account, or can select a p_T cut [further details are given elsewhere⁶]. The fast selection of events can be done by applying a logic "1" at the bottom of the array and counting how many clusters of logic "1's" (i.e. how many tracks) are present at the top. The hits are now connected in track segments, and we know where these track segments end, so through a reading-back operation we can rearrange the information in k vectors, where k is the number of tracks (and the hits are the elements of the vectors). The reading-back operation should proceed as follows:

- 1) A word, composed of all 0's but a 1 in front of node A (see Fig. 2) is presented to the 12th row of the matrix; the coding of this word represents the address of the first hit belonging to the connected track segment.
- 2) The content of the 11th row of the matrix is read out and coded and it represents the address of the second hit belonging to the connected track segment.
- 3) Operation (2) is repeated for all rows down to the first. Operations (1), (2) and (3) should be done for all tracks, so the time needed for the building up of the track vectors increases linearly with the track multiplicity.

Fig. 1 $\rho\phi$ projections of tracks produced in an e^+e^- interaction (z is the incoming beams direction).

Fig. 2 A 12×16 contiguity matrix with a track and some uncorrelated noise. Three types of masks (A, B, and C) are also shown. Mask type A is used in the 12×16 matrix.

Fig. 3 A 4×4 contiguity matrix, where all the logic circuits are shown. Symbols on the right help to correlate Fig. 3 with Fig. 2 and to understand the connection with the TPC output signals.

3.2 Circuit implementation

The logic circuit equivalent to the matrix shown in Fig. 2 is presented in Fig. 3 (for a 4×4 array only, the correspondence between the symbols of Figs. 2 and 3 is shown separately together with the connection with the detector). In the flip-flop N (FFN) the information coming from the pads is then memorized, while the other flip-flops (F-F) contain the contiguity mask pattern that should be loaded together with the event pattern [for instance through appropriate shifts (left, right, up, and down) of the FFN information into the contiguous F-F].

To obtain the full processor, one should add to the circuit shown in Fig. 3 the way of addressing the various rows (essentially through multiplexers and demultiplexers).

A more sequential contiguity processor ($8 \text{ rows} \times 16 \text{ columns}$) has already been realized with standard integrated circuits (ICs) and has been shown to work with the expected performances¹.

A breakthrough in this technique can be obtained with the adoption of gate arrays⁷. The repetitive structure of a contiguity processor is, in fact, well suited to being implemented in these semi-custom-built integrated circuits. We evaluated the size of the matrix that one can put into a typical fast gate array (between 10^3 and 6×10^3 equivalent gates, e.g. MCA 1200 ECL or MCA 2500 ECL or SCX 6360), which is between 4×4 and 8×8 . In fact each node -- see Fig. 3 -- is made up of three RS flip-flops, one 4-input OR and four 2-input ANDs, and to build up such a structure one needs approximately (see Ref. 7) 50 gates.

Another limit comes from the number of I/O pins: if fully parallel input on a $m \times n$ matrix is required we should have

$$\begin{array}{lll} \text{INPUT} & \left\{ \begin{array}{ll} m \times n & \rightarrow \text{for writing all FFN} \\ 1 & \rightarrow \text{for resetting all flip-flops} \\ 2 \times (m+n) & \rightarrow \text{for connections with other gate arrays} \\ & \text{(needed for the ripple of the contiguity} \\ & \text{mask network)} \end{array} \right. \\ \\ \text{OUTPUT} & 2 \times (m+n) \rightarrow \text{for connections with other gate arrays} \\ & \text{and read-out.} \end{array}$$

This gives 49 I/O pins for a 4×4 matrix and 129 I/O pins for an 8×8 matrix (the number of pins of a gate-array chip ranges from 28 to 144) plus some instruction and control pins and power lines.

A fast (ECL) gate array can work up to frequencies of ~ 200 MHz with a gate propagation delay of ~ 1 ns, but the number of gates per chip is limited to 2×10^3 (e.g. MCA 2500), while a CMOS gate array can work at 40 MHz, with a gate delay of 2 ns and a maximum density of 6×10^3 gates/chip (e.g. SCX 6360).

After the event has been loaded (during the drift-time in the case of the TPC) the writing of connections will require -- for instance in the case of mask A of Fig. 2 -- 4 clock cycles; we then have to allow 2 clock cycles for the decoding of the end-point of each track crossing the detector. This means that after 100-300 ns we will know how many tracks crossed the detector.

The reading-back operation that will build up the vectors of the tracks requires several microseconds and is mainly limited by the speed of the external electronics. Assuming, for instance, a 100-ns/word speed, a 20-track event, and a 15-layer detector we have:

$$t(R/O) = 100 \text{ ns} \times 20 \times 15 = 30 \text{ } \mu\text{s} .$$

The track vectors should then be ready for a fast high level trigger.

4. THE THREE-DIMENSIONAL CASE

4.1 The time projection chamber example

The TPC is a device that detects space points along the tracks produced in an event. A trigger working on space points instead of one working on projections is then conceivable⁶.

A possible sketch of a three-dimensional contiguity processor is shown in Fig. 4a. A memory mapping the contiguity matrix size (one 8-bit word per node) should contain the third (= z) coordinate of the corresponding node (that represents a point in the $\rho\phi$ plane).

Fig. 4 a) Sketch of a three-dimensional contiguity processor for the TPC (see text for details). b) TPC layout and interaction vertex constraint.

The z-coordinate should be calculated inside the gate array through the measurement (counting clock pulses) of the time of arrival of the pad pulse, that sets each node flip-flop (FFN). This information is then stored in the appropriate location of the "image memory".

The approach allows only one hit per pad to be recorded; Monte Carlo calculations should be performed to evaluate the efficiency of such a trigger and the required ϕ binning. Multihit capability on the same pad is possible, by increasing the complexity or the number of chips. We think that it is necessary to achieve a space accuracy of a few centimetres in the outer layer of the TPC in such a way as to be able to match track and calorimeter information in the third-level trigger.

A condition of the vertex of the interaction (see Fig. 4b) could be added, inhibiting the writing of the nodes in certain time (space) domains.

4.2 An alternative three-dimensional time projection chamber trigger

Since the magnetic field in the TPC is parallel to the z direction, all tracks will in first approximation be straight in the ρz projection. Therefore already a linear extrapolation of the TPC track can be used to check whether the track comes from the intersection point or not. This fact is used in the proposed three-dimensional TPC trigger.

As described previously, the TPC trigger will use signals from many small elements. Radially the division is given by the 16 different pad rows. Azimuthally individual pad signals will be ORed together to give good resolution at a moderate price. And the division into elements in polar angle θ will be produced by a time-slicing mechanism, which will be described below.

The treatment of the signals in the $\rho\phi$ projection, as explained for the two-dimensional case, will be repeated again in the three-dimensional trigger, however for individual time slices completely separated and not integrated over the entire drift-time. Figure 5a shows one quarter of the TPC in the ρz projection. Indicated are several areas C_i , which all point to $\rho = z = 0$, and which have an opening angle of $\Delta\theta^1$. In the TPC these areas form three-dimensional cones ($0 \leq \phi \leq 2\pi$), and have a different shape depending on their average θ angle. Each individual track coming from the interaction point should be found in one of these cones. For practical reasons, one will have to define the cones with a moderate overlap, in order not to lose tracks which cross from one cone to its neighbour.

Figure 6 shows a block diagram of the proposed trigger. The contiguity processors (CP) are identical to those for the two-dimensional case. They are repeated for each half TPC n times. However, they are loaded from "distribution switches" with the corresponding signals p_r (= OR of pad signals), where the index r indicates the index of the 16 pad rows. The distribution switches are advanced according to the scheme in Fig. 5b. They rotate slowly for signals from an outer radius, and rapidly for small radii. This corresponds exactly to the wanted cones, defined in Fig. 5a, which then contain each time the complete information on individual tracks.

Fig. 5 a) Sketch of an alternative three-dimensional contiguity processor for TPC (see text for details). One quarter of the TPC [in the ρz projection, with cones C_i ($i = 1, \dots, n$) of $\Delta\theta$ width] is shown. b) Timing of the different control signals, for advancing the distribution switches of Fig. 6.

Fig. 6 Block diagram of the loading mechanism for the processor shown in Fig. 5.

4.3 The high-density projection chamber example

The high-density projection chamber (HPC)⁸ is a device that detects showers occurring in dense material interleaved with gas employing a technique similar to that used in the TPC. In the TPC case we built clusters in three dimensions, the same operation is needed in the HPC in order to assign to a shower the many pieces of information related to it. To clarify the problem, let us give some parameters of the HPC proposed for DELPHI⁹

total number of words to be read out	$\approx 3 \times 10^6$
number of non-zero words per event	$\approx 10^4$
mean number of non-zero words per shower	$\approx 5 \times 10^2$

We have then to build up three-dimensional clusters of about 500 elements each.

There are two important differences compared to the TPC case:

- i) The shower is not necessarily emerging on the external layer of the detector and does not start necessarily at its internal layer.

- ii) A shower is a true three-dimensional object, while a track can be reduced to a two-dimensional object with appropriate transformations.

The design of a contiguity processor for the HPC data treatment requires then some modifications to the schemes previously described:

- i) The detection of a contiguity cluster should *not* be done between the first and the last layer of the detector, but amongst any two layers that are separated by more than N layers (N to be chosen according to the physics requirements regarding showers). This requirement implies a small change in the addressing scheme of the matrix nodes.
- ii) We should add, for each node, a counter that measures the size of the shower along z. This counter should be incremented by each contiguous pad pulse exceeding a threshold (see Fig. 7). As for the three-dimensional TPC trigger we accept, in this scheme, only one shower per pad. Monte Carlo calculations are needed to estimate the required granularity and the trigger efficiency. Systems allowing > 1 shower per pad could be envisaged, increasing the image memory size and the number of cluster counters or simply duplicating the single-hit scheme.

Contiguity treatment of HPC data is intended in order to know which are the constituents of each shower; this information should then be used in third-level trigger for fast calculation of energy deposit and shape of the showers and for correlation with data from other detectors.

4.4 Implementation in gate arrays

The circuit layout for three-dimensional TPC and HPC triggers differs slightly, but, since the overhead for a mask production is (at present) not negligible (30-50 KSF), one can think of designing a chip that contains the two options.

A scheme of the type shown in Fig. 7 should contain both (TPC and HPC) cases.

Fig. 7 Sketch of a three-dimensional contiguity processor for HPC (see text for details).

The number of gates per node is approximately⁷

50 gates (NODE MATRIX),
 10 × N gates (IMAGE MEMORY),
 10 × M gates (CLUSTER COUNTER MATRIX).

Some gates should then be added for multiplexing/demultiplexing data and for control of the process. The time needed to build up the clusters of the showers and to transmit them to the following treatment depends on the required granularity; if the numbers given in Section 4.4 are taken

20 showers/event,
 30 hits/shower,
 7 layers of 256 elements each;

we then have

$$100 \text{ ns} \times 20 \times 30 = 60 \text{ } \mu\text{s} .$$

What have been described up to now are particular cases of contiguity processors; in general a gate array containing a contiguity matrix can be organized as follows:

- a $m \times n$ contiguity matrix that performs the "connection operations" (like an ALU performs arithmetic and logic operations);
- some distributed intelligence around, which controls the connection operations and can execute a very simple program (as in a microprocessor of the first generation).

4.5 A quasi-realistic example

Let us assume a ϕ binning of 1.4° ($\equiv$ 256 steps in 360° ; $\approx$ 0.7 cm in the inner layer of the TPC and $\sim$ 3 cm in the outer layer; $\approx$ 4.5 cm in the HPC) is chosen. We then have to group pads together (OR-ing them either at the output of the shaping amplifier or at the output of the FADC, as is shown in Fig. 8).

Since we want to have approximately the same accuracy in all three coordinates, we should then choose a clock that divides the drift space into $\sim$ 2 cm (for TPC) and $\sim$ 5 cm (for HPC) and an image memory counter that can go up to $140/2 = 70$ ($\Rightarrow$ 64) for TPC and less for HPC. The image memory will then be built up with 6-bit counters ($N = 6$). The cluster counter matrix will most likely be made of 2-bit counters ($M = 2$).

We can then estimate the number of gates per node to be:

$$50 + 10 \times 6 + 10 \times 2 = 130 .$$

A 4×4 matrix can then be implemented in an ECL gate array, while an 8×8 matrix is slightly beyond the present CMOS technology.

The number of chips to be fabricated in these two extreme cases can be estimated as follows:

- a) TPC $\Rightarrow$ 16 layers of 256 ϕ -binning $\Rightarrow$ 256 4×4 matrices,
 - b) HPC $\Rightarrow$ 7 layers of 256 ϕ -binning $\Rightarrow$ 112 4×4 matrices.
- These numbers will obviously scale if a different granularity is chosen or if the multihit capability is required.

The cost is about 100 SF/chip and is not supposed to decrease in the near future. The overhead for mask production depends on computer aided design (CAD) facilities accessible to the DELPHI Collaboration.

Fig. 8 Possible connection between the HPC pads and the contiguity processor.

5. CONCLUSIONS

We have shown the advantage of using contiguity processors implemented in gate arrays for fast trigger and fast pattern recognition in DELPHI. We believe that the potentialities of contiguity processors have not yet been fully investigated.

REFERENCES

1. G. Darbo, Thesis, University of Genoa, AA 1979/80.
G. Darbo and S. Vitale, Nucl. Instrum. Methods 190 (1981) 81.
2. L. Rossi, A trigger for DELPHI, CERN Internal Note, 26.4.82.
3. E.C. Zeeman, The topology of the brain and visual perception *in* Topology of 3-manifolds and Related topics (Prentice-Hall, NY, 1962), pp. 240-256.
4. M. Brafman et al., SLAC-PUB 2033, October 1977.
5. M. Conversi and C. Verkerk, Nucl. Instrum. Methods 158 (1979) 217.
6. G. Darbo et al., A high parallelism structure for real time pattern recognition applied to colliding beam experiments, paper presented at the Padua Conference on Specialized Processors in Elementary Particle Physics, March 1983.
7. See, for instance, Motorola MECL-10000 Macrocell Array design manual and Siemens Gate-array design manual.
8. E. Albrecht et al., IEEE Trans. Nucl. Sci, NS-30 (1) (1983) 142.
9. O. Ullaland, DELPHI 82/54 ELEC, December 1982.