DD

CERN - PRE 89-088

C1

MAX-PLANCK-INSTITUT FÜR PHYSIK UND ASTROPHYSIK
WERNER-HEISENBERG-INSTITUT FÜR PHYSIK

10 JUL 1990

CM-P00062907

MPI-PAE/Exp.El. 216
November 1989

**“Iconic” tracking algorithms for high energy physics
using the TRAX-I massively parallel processor**

NA35 COLLABORATION

presented by

G. VESTERGOMBI

**MAX-PLANCK-INSTITUT FÜR PHYSIK UND ASTROPHYSIK
MÜNCHEN, GERMANY**

8000 MÜNCHEN 40 - FÖHRINGER RING 6

**"Iconic" tracking algorithms for high energy physics
using the TRAX-I massively parallel processor**

NA35 C o l l a b o r a t i o n

presented by

G. Vesztergombi

Max-Planck-Institut für Physik und Astrophysik

Föhringer Ring 6, D-8000 München 40, Fed. Rep. Germany

ABSTRACT

TRAX-I, a cost-effective parallel microcomputer, applying Associative String Processor (ASP) architecture with 16K parallel processing elements, is being built by Aspex Microsystems Ltd (UK). When applied to the tracking problem of very complex events with several hundred tracks, the large number of processors allows one to dedicate one or more processors to each wire (in MWPC), each pixel (in digitized images from Streamer Chambers or other visual detectors), or each pad (in TPC) to perform very efficient pattern recognition. Some linear tracking algorithms based on this 'iconic' representation are presented.

Extended version of talk given at the
Conference on Computing in High Energy Physics
University of Oxford, England 10-14 April 1989

1. INTRODUCTION

Since the birth of particle physics the tracking detectors ranging from cloud chambers through bubble chambers to TPCs (Time-Projection-Chamber) represented probably the most powerful experimental tools to reveal the secrets of physics inside the atomic nuclei. Intensive use of Von-Neumann type sequential computers produced spectacular results in pattern recognition and track reconstruction. Generally the first step in any tracking device is the 'zero-suppression': one stores only those wire numbers which have a hit; measures only selected 'bubbles' in bubble or streamer chamber photographs; makes a list of 'pad-number,time-slot' pairs in the TPC which have some deposited charge. This list of 'symbols' representing individually each hit turns out to be extremely economic in applications where the relative number of active cells is only tiny fraction of the total and it is well adapted for treatment in the usual sequential machines.

There are, however, situations where even the active cell data rate after zero-suppression is prohibitive. For example in heavy ion collisions at the proposed RHIC accelerator one expects up to 10000 particles per interaction; or in LHC (planned at CERN) the number of interactions can reach $10^7 - 10^8$ per second. In order to handle this enormous flow of information one should try to apply more sophisticated methods for data reduction.

The above mentioned symbolic (hit-list) approach has the disadvantage that the 'trivial' topological relations which are obvious on the detector level, such as which points belong to a given track, are practically lost and can be recovered only by tedious number (=symbol) crunching algorithms. If one could exploit this topological information on the 'raw-image' directly one could gain a lot because

- a) simpler 'neighbourhood' algorithms (cluster search, track following) could be applied;
- b) most of the algorithms could be run in parallel providing effective speed increase by orders of magnitude.

In order to exploit effectively the potential merits of this so called 'iconic' approach compared to the 'symbolic' one, cheap and flexible parallel processing architectures are required. One possible candidate for this role is ASP.

2. A S P SKELETON

"Associative String Processor (ASP) microcomputers provide highly versatile components for the low-cost implementation of high-performance information processing systems, and have the potential to achieve cost-performance targets in the range of 100 to 1000 MOPS/\$1000 (Million [12bits-add] Operations Per Second)" as stated in ref. [1], where one can find the detailed description of the ASP architecture. Here we are dealing with a maximally reduced version which includes only the very basic features necessary to effectively use the concept of parallelism and disregards the I/O questions and other technical details.

The ASP skeleton shown in Fig.1a consists of a single string of APEs (Associative Processing Elements) and we restrict our discussion only to single-step 1-bit left-right inter-APE communication, that is, direct data exchange is assumed only between neighbouring APEs along the string. Additional to this inter-APE connection all APEs share bit-parallel 'data-', 'activity-', 'control-buses' and one 1-bit feedback line called Match Reply, or MR. Through the 'data-bus' each APE is connected to the 'central memory' called ASP Data Buffer (ADB). Of course, in a given moment only one word is available from the 'central memory' but depending on the actual situation it can be communicated simultaneously to any subset of APEs, which makes scalar-vector operations rather easy.

Basically, the ASP has SIMD (Single Instruction with Multiple Data) architecture. A given instruction is executed, however, only by a preselected subset of active APEs, where the activation process is steered by the running program itself.

In contrast to more traditional parallel computer architectures, ASP uses content-matching rather than location-addressing techniques. ASP selects APEs for subsequent parallel processing by comparing their 'data-register' and 'activity-register' content with the states of the corresponding 'data-' and 'activity-buses'. Therefore the COMPARATOR is an extremely important part of the single processor element. The APE is sketched in Fig.1b. Each APE incorporates an n-bits 'data-register' and an a-bits 'activity-register', thus the required comparator length is $c = n + a$. In the following we assume $n=64$ and $a=5$.

Each APE is a mini-microcomputer with its own single-bit CPU performing four basic operations: match, add, read and write. As the structure of APEs is very

simple (they are "intelligent memory cells") many of them (up to 1024 in the near future) can be put on a single VLSI chip. The speed is assured by the 20 MHz clock frequency.

3. SYMBOLIC VERSUS ICONIC TRACKING IN MWPC ARRAY

For the sake of simplicity of the argument let us assume an idealized Multi-Wire-Proportional-Chamber array comprised of $M=9$ MWPC planes with y-coordinate read-out. Let the wire pitch equal 2 mm which corresponds to $W=500$ wires on a $1m \cdot 1m$ plane. The chambers are positioned equidistantly along the x-axis at positions $X_1, X_2, \dots, X_M$. We assume 100 % chamber efficiency and signals on the 2 nearest wires for each traversing particle on each plane (in an ideal chamber the particle never hits exactly a single sense-wire of zero thickness).

The particles are assumed to come from all direction and to fulfill the condition that if they hit the first (last) chamber they will go through the whole array until the last (first) plane. Thus for $N=20$ particles there will be exactly $K = M * N = 180$ hits.

For additional simplicity we assume that the wire-pairs having signals from different particles are not overlapping, and there is at least one 'empty' wire between the different hits. This condition is not essential for the applied algorithms, but it simplifies the counting of the necessary computational steps. In the following section we outline one sequential and one parallel algorithm to produce the track candidate list.

3.1. SEQUENTIAL ALGORITHM

For the sequential processing one can use the following strategy:

- i / Define the list of all possible track combinations $N*N = 400$ using the first and last MWPC planes.
- i i/ Search for a hit within a distance $\delta = 2mm$ on the remaining $M - 2 = 7$ planes using the ordered list of $y = (w_1 + w_2)/2$ coordinates where w_1 and w_2 are the 2 wires fired for the given hit. For the comparative studies we count the following computational steps:
 - a) PREDICT the hit coordinate on a given plane for a given track candidate

(1 step);

b) SEARCH for the nearest hit ($L = 5$ steps because in binary search it requires $2^{(L-1)} < N \leq 2^L$ decisions to arrive at the correct value, i.e. $L \sim \log_2(N)$.

c) CHECK that it really belongs to the track (1 step)

iii/ Print out those combinations which have $M = 9$ hits.

All together this procedure requires

$$S = N * N(M - 2)(L + 2) = 19600$$

computational steps.

3.2. PARALLEL ALGORITHM

For solving the MWPC tracking problem we assume an ASP microcomputer with 500 APEs. Thus we can assign to each wire number 'i' one processing element, but the same wires of different planes should share the same APE. If wire 'i' of plane 'j' is fired then in APE 'i' the bit 'j' will be set to 1 otherwise it remains 0. This mapping of the MWPC 'image' onto the ASP data-registers is shown in Fig.2. We call this mapping 'iconic' because the resulting bit pattern is faithfully representing the original two dimensional space relations between the track points.

As a simple illustrative exercise on parallel computing one can demonstrate how to contract the double wire signals in chamber #1 into single ones. The 3 step procedure is shown in Fig. 3.

step 1 : COPY bit#1 of the data-register to bit#1 of the activity-register (A1).

step 2 : SHIFT bit#1 of the activity-register by a step to right (downward in the figure).

step 3 : DELETE bit#1 of the data-register in those APE's where the bit#1 of the activity-register is equal to zero (i.e. $A1=0$).

The "STRING" nature of the APE array is used in the step 2, performing the SHIFT through the neighbour connection. The "ASSOCIATIVITY" is used in step 3, activating only those subset of APE's which has $A1=0$. The activation is performed by help of the COMPARATOR. Using ternary logic it compares the (data + activity) register content to the (data + activity) buses which in the present case

have "x=don't care" setting for all bits except for bit #1 of the activity-bus which is set to zero.

For book keeping purposes one puts the wire number 'i' in the data-register bits{11..19} of APE 'i'.

In general $y = mx + b$ is used to define the track equation. The role of the 'b' intercept will be played in our case by the bit #9. Instead of the slope parameter 'm' we introduce a more convenient quantity

$$DY = m(X_1 - X_M) = y(X_1) - y(X_M)$$

which we call the 'slope-distance'. Within our model experiment DY can have integer values in the range $-W < DY < W$. First let us deal with non-negative slopes only. Including $m=0$ there are $W=500$ possibilities.

Due to the iconic representation it is trivial to select those tracks which have $DY = 0$. One should make an ACTIVATION for those APE(s) which have all the 9 bits set to 1. If one (or more) matches occur then the wire number(s) in bits{11..19} should be printed out for the matched APE(s). That will define 'b' and the controller knows that $m=0$.

The case $DY = 1$ is reduced to $DY = 0$ by a very simple trick. If one shifts bits{1,2,3,4} by one APE right (down in figure) then all tracks with this slope-distance will be masked out in the same way as above. Printing the matched wire numbers one gets 'b' and the controller knows the present position at which 'm' was tested.

It is generally true that after finishing the test for $DY = n$ there exists a shift combination for bits{1..8} which can be written as single byte, setting to 1(0) those bits which should be shifted (left untouched) for preparing the test of $DY = n + 1$ case. Fig. 4 illustrates the procedure up to $DY = 3$. Due to the 2 pixel width track representation it can happen that the same track produces more than one match. These matches however are neighbours in b and DY therefore it is trivial to get rid of them by the neighbour killing technics described in the above illustrative example.

The total number of shifts is equal to $(M - 1) * W/2 = 8 * 500/2$ because the increase of DY corresponds to a rotation around the bit #9 position, the nearer the bit to #9 the smaller number of shift is performed on it.

The number of so called "DY=0" checks is W. Since the tracks can have a negative slope with the same probability the whole procedure should be repeated for $DY < 0$ cases, too. Thus taking into account all possible slopes one finally gets the total number of computational steps equal to

$$W * (M - 1 + 2) = W * (M + 1) = 5000.$$

Of course, it is independent from N, the number of tracks, because all the $W * W = 250000$ possible track candidates are always checked regardless of the image content.

3.3. EVALUATION

The computational speed of the symbolic algorithm is strongly dependent on the number of tracks, N, and in lesser extent on the number of planes, M, whereas the iconic method is sensitive to the total number of wires $W * M$. For comparative studies Table 1. summarizes the number of computational steps required for a representative set of N, M, W values. As is expected the iconic approach wins for larger N even if one is obliged to use more wires to be able to accomodate the larger track densities. In heavy ion experiments, where 100 particles per events are common place, the iconic method easily can reach a factor of 50 in superiority.

In counting the computational steps we neglected the execution time which is required in real hardware. In this respect ASP once more comes out much better, because most of the time is spent on bit-shifting which is a "hard-wired" command taking only 1 cycle of 100 ns. Whereas in any sequential machine, even using assembler code, for a computational step such as y-prediction or one decision during the binary search several machine commands are necessary.

The high speed of parallel computing provides even more advantages. If one connects the ASP directly on-line to the experiment then one can get the tracking results within a few milliseconds, enabling higher level trigger decision and/or a considerable output data rate reduction.

4. HOUGH TRANSFORMATION

In two-dimensional image processing the search for local linear structures is a very frequent task. For example the NA35 streamer chamber picture is digitized providing $1536 \cdot 2048$ gray valued pixels. A sequential code was developed [2] taking small $100 \cdot 100$ patches. On these small patches the curvature of tracks is negligible thus the problem of local track finding is reduced to a search for linear segments. Hough[3] has proposed a procedure for detecting lines in two-dimensional images.

A line (or in more general, a curve) in Cartesian coordinates is represented by a single point in the so called 'parameter space'. If the points (x_i, y_i) belong to the line which passes at a shortest distance d from the origin of the coordinate system and which has a normal with an angle θ relative to the positive x-axis then the value of the expression

$$x_i * \cos(\theta) + y_i * \sin(\theta)$$

is independent of 'i' and always gives d (Fig. 5). If (x, y) is not on this line the value will be different. Thus calculating for each 'black-pixel' in the image for different $\theta_1, \theta_2, \dots, \theta_k, \dots$ values the corresponding $d_1, d_2, \dots, d_k, \dots$ and filling a 2-dimensional (d, θ) histogram, the lines in the original image will stick out as peaks on this 'lego' plot. The problem of LINE-SEARCH in the (x, y) plane is transformed into PEAK-SEARCH on the (d, θ) Hough-transform plane. This form of Hough-transform is ideal for symbolic treatment in sequential machines or for data driven pipelining processors [4]. For the iconic approach, however, one should rephrase it in order to avoid lengthy floating point calculations and reach a form which conserves the topological relations between the image pixels. To each (d, θ) bin of the Hough-transform corresponds a corridor on the original image. Fixing ' θ ' but running ' d ' one gets parallel shifted copies of the starting corridor. Calculating this ' θ -column' of the Hough-transform is equivalent to counting the number of 'black-pixels' in each of these corridors. In the proposed ASP algorithm the trick is that one is able to define these corridors in a natural way on the original image itself.

The Single Angle Hough-transform (S A H) procedure is based on the idea of A. Krikelis [5]. In the 'iconic' representation of the 2-dimensional image each PIXEL has its own COMPUTER (=APE). Comparing to the previous more "economic" mapping in the MWPC case, this is a rather "luxury" way of 'iconic' representation

of the 'image' data, but it pays off in performance. TRAX-I has 16k APEs, thus one can load into it $s=800$ lines of a slice with $n=20$ width (or a square patch of 128×128). It should be remembered that ASP is a STRING, thus the neighbours inside a given line are also neighbours in the ASP but column-neighbours are 'n' positions apart along the string (Fig. 6). The right neighbour of pixel $\{ X = n, Y = k \}$ is the pixel $\{ X = 1, Y = k+1 \}$. For each 'black-pixel' bit#1 is set to 1 in its APE. Counting the number of 'black-pixels' inside a given θ corridor with intercept parameter b_i is reduced to a series of shift and add operations in the following way.

One selects (=activates) the last (=the rightmost along the string) APE belonging to corridor 'i' and assigns bits{11..19} as counter C_i which is empty at the beginning. Then one repeats the following cycle till the 'first' corridor point "reaches" the position of the above assigned 'last' APE with counter C_i :

- a/ If bit#1 is set in the activated APEs: $C_i = C_i + 1$
- b/ Shift one step right the contents of all bit#1 until the next point in the corridor reaches the original position of the last (the so called 'storage-cell') point in the corridor. This number depends only on θ and it is a priori known in the ASP controller, thus filling the (b, θ) plot for the whole column simultaneously. It means the the SAH algorithm has "parallelism of the first power".

Using the associative character of TRAX-I one can create "parallelism of the second power" , too. Exactly this is done by the All Angle Simultaneous Hough-transform (A A S H) algorithm. "All angles" means that one requires only 2 passes: the first for tracks between angles -45° and $+45^\circ$ (y-slice scan) and the second for tracks between $+45^\circ$ and $+135^\circ$ (x-slice scan). The number of θ_i values is determined by the number of columns 'n' in the processed patch, because it is not worth having different corridors for tracks which are starting from the same point of the first column and whose separation is less than 1 pixel in the last column. In this case the number of necessary storage cells is twice as many as the number of pixels in the patch, thus e.g. in y-slice scan the positive and negative θ corridors are collected in the same APE in bits{11..19} and bits{21..29} respectively. In case of the NA35 streamer chamber pictures $n=20$ gives about 2.3° resolution. Fig.7 shows 4 such neighbouring slices. The tracks which are reconstructed as peaks from the Hough-transform matrix are superimposed. It is remarkable how well this procedure

works in this very dense picture fragment.

The AASH routine is both efficient and very fast. The processing speed for a slice of size $s \times n$ is independent of s , and the whole calculation requires only $n \times n$ right-shifts and twice as many additions of $\log_2(n)$ bits. In TRAX-I the setting $s=800$ and $n=20$ requires $N_{slice} = 4400$ computational steps per slice, which corresponds to a processing time $T_{slice} = 440 \mu s$ assuming 100ns cycle time. For the whole 3 Mpixel picture it would take 180 millisecond. As the Hough-transform is done independently for single slices a "parallelism of third power" is also possible. If the length of the APE string is big enough to load all slices at the same time then simultaneous slice processing can bring down the processing time required for the whole image to the level of T_{slice} .

5. CONCLUSIONS

Nowadays massively parallel processing is becoming more and more of a reality both in hardware and software. In particular, ASP architecture on VLSI chips represents the state of art in present technology. TRAX-I with 16k processing elements (providing about 10 GOPS computing power) is being built by ASPeX Microsystems Ltd in Brunel (UK). Software parallelisation in 'iconic' processing comes very naturally. Preparation of a library for basic operations is progressing well, which makes it rather easy to develop programs in high level languages.

In high energy physics using fast tracking or clustering algorithms one can get more "intelligent" input for higher level triggers and/or considerable reduction in raw data output. Within 2-3 years ASP machines with up to 1 million APEs are expected in the price range of 1\$ / APE. These TOPS (Tera Operations Per Second) machines surely deserve the interest of physicists planning "top-energy" experiments on the new generation of accelerators.

TABLES

Table 1. Comparison the number of computational steps between symbolic and iconic algorithms

M=9 W=500 (500 APEs)			
N	Symbolic	Iconic	Symbolic/Iconic
10	4200	5000	0.84
20	19600	5000	3.92
40	89600	5000	17.92
100	630000	5000	126.00
M=9 W=1000 (1000 APEs)			
10	4200	10000	0.42
20	19600	10000	1.96
40	89600	10000	8.96
100	630000	10000	63.00
M=16 W=1000 (1000 APEs)			
10	8400	17000	0.49
20	39200	17000	2.31
40	179200	17000	10.54
100	1260000	17000	74.12

REFERENCES

- [1] R.M. Lea, IEEE MICRO, October 1988, 10.
- [2] F. Puhlhofer, D. Rohrich and R. Keidel, Nucl. Instr. and Meth. A263 (1988) 360-367.
- [3] P.V.C. Hough, U.S. Patent 3069654, 1962.
- [4] W. Krischer et al., Internal Report, CERN-LAA/RT/88-06.
- [5] A. Krikelis, Aspex Microsystems Ltd, June 1988, DARPA.wp/ak 309-006.

FIGURE CAPTIONS

1. Schematic diagram of the ASP skeleton: a) Associative String Processor; b) Associative Processing Element (APE).
2. 'Iconic' mapping of the MWPC hit pattern onto the ASP.
3. Double wire signal contraction in chamber #1 (bit#1 of the data-register): a) copy; b) shift; c) delete (zeros are shown explicitly only to mark the result of the action).
4. 'Iconic' tracking by 'rotation' in a 9 plane MWPC array: a) Input data, no match for any track at $DY=0$ test; b) Rotation to $DY=1$, match for track#1; c) Rotation to $DY=2$, match for track#2; d) Rotation to $DY=3$ position.
5. Hough-transform parameter definition.
6. 'Iconic' mapping of a 6×6 pixel image patch onto the ASP: a) "Black pixels" for 2 parallel tracks in corridor $DY=2$ are shown. Due to "inefficiency" some hit pixels are not blackened; b) The APE's content is shown after 0, 1, 4, 5, 8, 9 shift steps.
7. 4 neighbouring ($n=20$ pixels wide) slices of a real streamer chamber image after digitization (dots). Tracks are identified as peaks in the Hough-transform of the same part of the image (lines).

a)

b)

Figure 1.

Figure 2.

Figure 4.

Figure 5.

Figure 6.

Figure 7