FERMILAB-Pub-96/030-E
E789

Use of a Track and Vertex Processor in a Fixed-Target Charm Experiment

M.H. Schub et al.

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

February 1996

Submitted to *Nuclear Instruments and Methods*

Use of a Track and Vertex Processor in a Fixed-Target Charm Experiment

M. H. Schub[*]
University of Chicago, Chicago, Illinois

T. A. Carey, R. G. Jeppesen[†], M. J. Leitch, J. C. Peng
Los Alamos National Laboratory, Los Alamos, New Mexico

C. N. Brown, Y. B. Hsiung
Fermi National Accelerator Laboratory, Batavia, Illinois

D. M. Kaplan[‡], C. Lee, G. Miller, J. Sa
Northern Illinois University, DeKalb, Illinois

P. K. Teng
Institute of Physics, Academia Sinica, Taipei, Taiwan, R.O.C.

Abstract

We have constructed and operated a high-speed parallel-pipelined track and vertex processor and used it to trigger data acquisition in a high-rate charm and beauty experiment at Fermilab. The processor uses information from hodoscopes and wire chambers to reconstruct tracks in the bend view of a magnetic spectrometer, and uses these tracks to find the corresponding tracks in a set of silicon-strip detectors. The processor then forms vertices and triggers the experiment if at least one vertex is downstream of the target. Under typical charm running conditions, with an interaction rate of $\approx$ 5 MHz, the processor rejects 80–90% of lower-level triggers while maintaining efficiency of $\approx$ 70% for two-prong D-meson decays.

[*]Present address: University of Minnesota, Minneapolis, MN.

[†]Present address: Science Applications International Corporation, 2950 Patrick Henry Drive, Santa Clara, CA 95054.

[‡]Present address: Illinois Institute of Technology, Chicago, Illinois

1 Introduction

One way to enrich the charm content of the data recorded in a fixed-target hadron experiment is to trigger on events containing vertices separated from the primary-interaction vertex. In a high-rate experiment, the trigger decision must be made very quickly. Various approaches for rapid secondary-vertex triggers have been proposed recently, including a Cherenkov multiplicity-jump trigger [1] and a Cherenkov detector with novel optics [2]. An approach that has previously been used to advantage is an impact-parameter trigger implemented via a fast processor using information from a silicon vertex detector [3].

We describe here a fast processor which uses information from a silicon vertex detector. In the E605/E789 spectrometer [4] (see Figure 1) we can reconstruct fully only those charm decays for which the daughter particles are reconstructed in the downstream spectrometer. Our processor therefore begins by finding tracks in the downstream spectrometer in the bend view (the bend is in the vertical plane), and then looks for silicon-detector tracks that correspond to tracks in the downstream spectrometer; this drastically reduces the combinatorics of track finding in the silicon detector. (The bend view of a typical hadron-pair event is shown in Figure 2.) Since our targets are small in the vertical and longitudinal directions (see Table 1), we can use the impact parameters of silicon-detector tracks in the bend view to decide whether tracks originate in the target. For each pair of tracks satisfying a given impact-parameter requirement, the processor calculates the intersection point of the tracks in the bend view, and triggers data acquisition if this vertex point lies a given distance downstream of the target and is consistent with the vertical position of the target within limits.

2 The Nevis Laboratories Data-Driven Processor System

The E789 track and vertex processor was constructed by modifying the E605 processor [5], adding components to find silicon-detector tracks and vertices. The processor was constructed using the Nevis Laboratories data-driven processing system [6], which consists of a set of synchronous processing modules and a protocol for interconnecting them. Each module implements some simple operation such as addition of two 16-bit quantities, comparison of a 16-bit quantity with upper and lower limits, or computation of an arbitrary function of an 8-bit quantity via table lookup. Modules are implemented in ECL 10,000, with a clock period of 50 ns. Typically, on each clock tick, data advance synchronously through the pipeline from one register to the next in many modules in parallel. Between registers, the data pass through processing logic (e.g. arithmetic logic units), which must thus be designed to have $< 50\,\mathrm{ns}$ settling time. Each module's output register is followed by a transparent latch, providing temporary storage for one datum in case the dataflow needs to be interrupted.

The processor algorithm is determined by the interconnection of the modules, by hard-wired plug-in "patches" on the modules, and, for some modules, by information downloaded into module control registers or memories. By modifying the input file used by the downloading program, one can update information on spectrometer alignment and magnet fields, change the values of cuts made by the processor, change the way the processor treats various trigger types, and enable or disable certain features of the processor algorithm.

2.1 Communication Protocol

The modules communicate synchronously over cables that carry 16 bits of data, 4 signals that control the flow of data, and 4 bits of "Name" information. The control signals are called "Valid", "Complete", and "Hold" (the fourth control signal, "Block Reset", is not used in our processor). On each clock cycle, the input register(s) of a given module will either store or ignore the data presented to them depending on the combination of signals that accompanies the data. If the input data are accompanied by a Valid signal, and if the Name bits match a pattern determined by a plug-in patch, then the data are clocked into the register, and a latch associated with the input register is set to indicate that valid data are present. For modules with memories, Name can be used as part of a memory address; this can be useful in a loop, where the Name bits can serve as a loop index, allowing modules to use different areas of memory on each pass.

The features of the processor communication protocol are best illustrated by considering the operation of a representative two-input processing module. Some two-input modules can produce a valid output only if they have valid data on both inputs. If such a module has valid data on only one of its inputs, it sends a Hold signal to the module supplying the data; the Hold signal tells the source module not to overwrite any valid data in its output latch until the Hold signal is turned off on some future clock cycle, when the two-input module has received valid data on its other input. A module that is holding valid data in its output latch due to a Hold signal may need to pass a Hold signal back through its input registers to the modules connected to its inputs. Hold signals can therefore be passed module-by-module against the flow of data, until they encounter an empty register (i.e., an "invalid" datum "annihilates" Hold). When a two-input module has valid data on both its inputs, it turns off any Hold signal it may have been asserting, and any data being held due to that signal can be released to flow down the pipeline. The Complete signal indicates that no further valid data can be expected; when a two-input module has received Complete on both its inputs, it passes a Complete signal on to the next module in the pipeline. A Complete signal on the output of the final module in the processor tells the data-acquisition system that the processor has finished processing the current event. The data-acquisition system then reads out the event data or not, depending on the processor's decision.

2.2 Module Types

Eleven different module types are used in the E789 processor. (In addition, a twelfth module, the Block Buffer, interfaces the processor to the data-acquisition system.) Several of these can perform arithmetic functions. The Arithmetic Operator (denoted by $+$ or $-$ in processor diagrams) can be programmed to perform a number of arithmetic functions, including addition and subtraction, and can also be programmed to act as a "switch" (S or Sr) to merge data streams.[1] The Comparator module ($<>$) performs an arithmetic comparison of its input value with upper and lower limits stored in internal registers, and changes the Name bits according to its decision. The Table module (T) has a downloadable memory which can be addressed by one or both input registers, or some combination of bits from both. The Normalizer (N) has internal memories addressed by its input register, along with arithmetic circuitry that can add, subtract, etc. the outputs of the internal memories.

Three module types are specifically designed to facilitate trackfinding. The Associator module (A) is designed to process associated hits from a pair of adjacent drift chambers with 1/2-cell-offset drift cells. It takes an ordered stream of hit data from such a chamber pair, compares each hit pair to see if both hits could have come from the same particle track, and produces a single output word for each associated pair. The output word is quantized in quarter-drift-cell units. The merged drift-chamber hit streams that are input to the Associators are formed by Ordered Merge (OM) modules, which reside physically in the drift-chamber TDC crates rather than in the processor itself. The Map module (M) is used to represent a list of addresses as an array of bits, allowing a predicted position to be compared with all entries in the list in a single clock cycle. Each word received at the "write" input causes a single bit in the array to be set. Each word arriving at the "read" input addresses the bit array, causing the addressed bit and several surrounding bits to be output. This provides a means of checking whether a drift chamber has a hit in the vicinity of the hit position predicted from hits in other chambers.

Block Buffer modules (BB) serve to store data for input to the processor and for output to the data-acquisition system [7]. Buffer modules (B) provide temporary FIFO storage internal to the processor. They pass data straight through from input register to output register when they are not receiving any Hold signal, and when they are receiving Hold they buffer their input data in their internal memories. They are used to prevent lockups that can occur in loops if Hold signals are allowed to propagate all the way

[1]Sr denotes a switch that "reciprocates," i.e. it accepts data alternatively from each input.

around the loop.

Several modules provide loop control. In the process of track finding, one often needs to loop over all possible combinations of hits in a pair of chambers or silicon planes. To do this, one must store all the data from each of the two chambers; this function is served by the List module (L). Data presented to the write input of the List module are stored as a list in memory, and a given data word may be retrieved by sending an integer index to the List module's read input. To loop over all combinations of hits in both chambers, one must generate all possible combinations of indices; an Indexer module (I) does this by keeping track of how many valid data words have appeared on each of its inputs and outputting a set of all possible pairs of integer indexes that can correspond to the inputs, with each index pair packed into a single 16-bit data word. The Indexer module is designed so that it can begin generating index pairs even before it has received a Complete signal. Finally, since some data may not pass a cut on a given pass through the loop, one must be able to retrieve the index pairs of data that have passed; this function is served by the List/Counter module (Lc). The List/Counter, like the List, stores all the data that arrive at its write input register, but instead of retrieving the data by index, it retrieves the data by position in the list: the second input on the List/Counter counts the input data, and retrieves the corresponding index pair only if the Name bits associated with the input data match a value determined by a patch on the List/Counter.

A concrete example of a loop structure is provided in Figure 3. Here we loop over all combinations of associated hits from the Y1/Y1$'$ drift-chamber pair and the Y2/Y2$'$ drift chamber pair. One List module for each drift-chamber pair stores all the associated hits from that pair. Meanwhile, an Indexer counts the hits from each chamber pair while beginning to send out index pairs, which are passed through and stored by the List/Counters. Passed-through index pairs are used to address the Lists during the first pass through the loop. Calculations are made on the drift-chamber data, and decisions are made by Tables at the bottom of the loop structure. Data from these Tables, along with decisions made by those Tables, are passed back up to the List/Counters, which output indices to recall from the Lists those data for which a favorable decision was made. The recalled data make another pass through the loop, and another set of computations and cuts is made on the next pass. Because the Name bits associated with each datum indicate which pass through the loop it is currently making, it is possible for data from both passes to make their way through the loop at the same time.

3 The Track Processor Algorithm

The portion of the E789 processor that finds tracks in the downstream (i.e. non-silicon) spectrometer was originally built for Fermilab E605 and is described in Ref. [5]. For E789, we removed the mass calculation shown in [5] and rearranged modules and patches so as to loop over pairs of hits in Y1/Y1$'$ and Y2/Y2$'$ (rather than Y2/Y2$'$ and Y3/Y3$'$). A diagram of the modified track processor is shown in Figure 3.

For reasons of speed, cost, and complexity, and because the spectrometer magnets bend tracks only in the vertical plane, the processor uses information from only those drift chambers, hodoscopes, and silicon-strip detectors that measure the vertical (y) coordinate. The track processor uses hits in the Y1/Y1$'$ and Y2/Y2$'$ drift chambers and a single-bend-plane approximation for each of the two analysis magnets to predict hit positions in the Y3/Y3$'$ drift chambers. To suppress out-of-time hits, only drift-chamber hits that have corresponding hits in the neighboring scintillator-hodoscope planes are used. The processor then checks for the presence of hits in the Y3/Y3$'$ drift chambers within a reasonable distance of the predicted hit position. If such hits are found, the processor computes the vertical position of the track at the downstream end of the beam dump, using a single-bend-plane approximation for the SM3 magnet and a two-bend-plane approximation for the SM12 magnet, and accepts the track only if it appears to be outside the beam dump (which is an intense source of background muons centered in the spectrometer aperture). These checks are done in the first pass through the track processor's loop. On the second pass, the processor can, if so directed by the data downloaded into it, demand that tracks have hits in the proportional tubes behind the hadron absorber, if we wish to trigger on muons, or it can demand that there be energy in the calorimeter cell that the track hits. On the third pass, the reciprocal of the momentum of the track and the track angle at the target are calculated. The angles at the target are passed on to the vertex processor, allowing the vertex processor to select only those silicon hits that fall near projections of a downstream track.

4 Track Processor Implementation

The implementation of the algorithm is shown in Figure 3. Drift-chamber data pass first through Comparators, which remove hits with unphysical drift-time values, then through Associators, which associate hits in each chamber pair, and then through Normalizers. The Normalizers express the drift-chamber hit positions in units of hodoscope counter widths, with enough fraction bits included so that each drift-chamber hit can be reconstituted later. The Map modules that come next check for hodoscope hits corresponding to the drift-chamber hits. Normalizers then reconstitute the original drift-chamber hit information and use the Map output to tag (in the Name field) those drift-chamber hits that correspond to hodoscope hits.

The processor then loops over pairs of hodoscope-tagged hits. The Normalizers and Arithmetic Operators are used to calculate the position of the track at a different detector location on each pass through the loop. For the first pass, the positions of the track at the Y3/Y3$'$ drift chambers and at the downstream end of the beam dump are calculated, and the right-hand Map module is used to check that there are hits in Y3/Y3$'$ near the predicted locations. The bottom Table checks that the track passed both the dump and Y3/Y3$'$ tests. For data that pass both tests, it sets the Name bits to indicate that the data are ready for the second pass through the loop and passes this information back to the top List/Counter. On the second pass through the loop, the track position at the proportional tubes is calculated, and the left-hand Map is used to check for proportional-tube or calorimeter hits. The upper Table checks for proportional-tube or calorimeter hits or both (depending on downloaded settings) to deterimine which data will make a third pass through the loop. In the third pass, the right-hand Arithmetic Operator calculates the track angle at the target, which is then passed on to the vertex processor. Hit pairs that have survived to the third pass are written out to the data-acquisition system through the Switch and Block Buffer at right. These track data can be used for further on- or off-line analysis and diagnostic checks on processor performance.

5 The Vertex Processor Algorithm

A diagram of the E789 silicon detectors is shown in Figure 4. Half the detectors measure the vertical (y) coordinate, while the others are tilted to provide stereo information for three-dimensional reconstruction. The vertex processor uses only the y detectors.

The vertex processor uses the projected angles of tracks from the track processor to select only those silicon-detector hits that could be associated with a downstream track originating near the target. It then loops over all combinations of the selected hits in the first and last y planes. It checks whether the line defined by each hit pair has an angle consistent with a downstream track, misses the target by a specified minimum impact parameter in a direction consistent with a decay downstream of the target, and is confirmed by hits in one or both of the other y planes. (Downloaded settings determine whether a hit in one or both confirming planes is required.) It then forms all pairs of tracks meeting the above requirements and finds each intersection vertex (y_v,z_v). The vertex processor triggers data acquisition if at least one vertex satisfies downloaded y_v and z_v cuts.

6 Vertex Processor Implementation

A diagram of the vertex portion of the processor is shown in Figure 5. Silicon-hit data pass first through Normalizers, which convert the hit positions to a format suitable for masking with downstream-track angle information.[2] The angle information from the track processor enters the vertex processor through a Comparator, which rejects drift-chamber tracks at angles outside the acceptance of the silicon detectors. Normalizers convert this angle information to a format suitable for masking the silicon-detector hits. The Maps then tag silicon hits that match the downstream-track angle information. The subsequent Normalizers implement the masking decision and convert the silicon-hit data back to their original form.

The angle-masked hit addresses from YS1 and YS4 now enter the loop. In the first pass through the loop, a set of Normalizers and a pair of Arithmetic Operators compute the impact parameter and angle for all combinations of YS1 and YS4 hits. Only tracks that meet a downloaded impact-parameter cut, have angles close to the angles of downstream tracks, and have both hits in the same silicon spectrometer arm are marked for subsequent passes through the loop. A list of track impact parameters, including those that do not meet the impact parameter cut, is written out to the data-acquisition system. This is useful in monitoring the target position on-line, to ensure that it agrees with the position assumed by the processor-downloading code.

On the second pass, the same Normalizers and Arithmetic Operators calculate, for each track candidate that survived the first pass, the strip numbers at which hits would be expected in YS2 and YS3. A pair of Maps then checks for hits in YS2 or YS3 at the expected positions. A Table selects tracks that have a hit in one or both of YS2 and YS3. On a third pass through the loop, these selected YS1/YS4 hit pairs are retrieved, and their track angles, impact parameters, and numbers of hits are passed on to the next section of the vertex processor, the "Vertex Endgame." Also on this pass, the selected YS1/YS4 hit pairs are written out to the data-acquisition system (via a Reciprocating Switch) for processor monitoring.

The "Vertex Endgame," shown in Figure 6, must calculate the vertex positions for all pairs of silicon tracks that meet the above cuts. To loop over pairs in a simple way, we first combine the number-of-hits, impact-parameter, and angle information for each track into a single data word; this is done by the two Arithmetic Operators at the top. Everything needed can be calculated in a single pass through the loop, so the loop apparatus requires only an Indexer and a pair of Lists. Inside the loop, the y coordinate and the logarithm of the z coordinate of the vertex point are calculated, using the expressions

$$z = \frac{|y_-| + |y_+|}{\theta_+ - \theta_-}$$

and

$$y = \frac{|y_-| - |y_+| + z(\theta_+ + \theta_-)}{2},$$

[2]In addition these Normalizers eliminate hits from some stereo planes also present in the data streams.

where y_+ and y_- and θ_+ and θ_- are the impact parameters and angles for the top and bottom tracks, respectively. The sums and differences in these formulas are done by Arithmetic Operators, while the multiplication and division are done using an Arithmetic Operator plus a Normalizer downloaded with a logarithm table.[3] A pair of Comparators cut on the y and $\ln z$ of the vertex. A Table interprets the Comparators' decision and, if appropriately downloaded, can apply a cut on the total number of hits on the two tracks. A List/Counter is used to write out to the data-acquisition system a set of indices of track pairs satisfying the cuts, for processor-monitoring purposes. If at least one track pair satisfies the cuts, the processor sends a "read" signal to the Event Generator Source (EGS) [7] of the data-acquisition system to initiate event readout.

It is worth pointing out a couple of features of the processor that are necessary to prevent pathologies in the data flow that may occur for certain rare input data. The first is the "Event Alignment" apparatus. Both the track processor and the vertex processor send event alignment information to the Vertex Endgame; this information is denoted by EA1 and EA2 in the processor diagrams. These signals are necessary because it may sometimes happen that there are no data for a particular drift chamber or silicon detector. In such cases, it is possible for the Complete signal that comes in for that detector component to propagate through the processor and reach the data-acquisition system while data are still being read in for other detector elements; this can result in the event being read out and data for a new event being input to the processor while data for the current event are still propagating through. To prevent this from happening, a Table in the Vertex Endgame is set to pass Complete to its output only when it sees Complete on both of its inputs, EA1 and EA2. (The inputs for this Table come from the bottoms of the loops in the track and vertex processor, so that they receive Complete only when data from all detector elements have been processed in the loops.)

The other pathology occurs when too many silicon tracks are found, causing the internal counters in the Vertex Endgame Indexer to overflow. This is prevented by putting a modified Buffer module at the output of the Vertex Processor. The Buffer is modified to ignore tracks beyond the maximum number which the Indexer can accomodate.

7 Diagnostic Tools

A number of diagnostic programs were used in designing and debugging the processor. During the design phase, a module-level simulation program was invaluable; this program allowed the user to test a specific processor configuration by defining which modules would be used and how they would be patched and interconnected, and then simulating the action of the processor on input data. During the debugging phase, a number of

[3]Addition and subtraction of logarithms are used to avoid multiplication or division of a variable by a variable, which would have required more complicated modules.

programs were used to test individual modules in isolation, and the assembled processor was tested with a program that could run the processor through a specified number of clock cycles and then read out its status, which could then be compared to the predictions of the simulation program. In this way, the user could locate exactly on which clock cycle a module gave the wrong output, what the output was, and which module produced it. In another valuable debugging technique, the simulation was used to predict the processor's output for a large number of events, and the prediction was compared to the output of the real processor on the same event data. With a large amount of data, one could detect processor problems that occured too rarely to be noticed by hand-checking events.

8 Performance

Table 1 summarizes some of the charm running conditions from E789's 1991 run, along with the processor's performance under these circumstances. The interaction rates are approximate averages; the instantaneous interaction rate varied considerably, but was typically about twice the long-term average interaction rate. The processor was set up to prescale a fraction of all input events; these events could be used to monitor the processor's performance. The average number of 50-ns clock cycles per event quoted in the table was determined by running the processor simulation on samples of these events. A typical distribution of the number of clock cycles per event is shown in Figure 7. The rejection factor is the ratio of the number of events processed to the number of these events that pass the processor algorithm. The higher rejection in the 1000 A Au running reflects the smaller target and tighter cuts that were employed.

The efficiency is the fraction of prescaled events passing off-line reconstruction cuts and having a downstream vertex and a mass near the D mass which are also passed by the processor. Note that the efficiencies shown in Table 1 are dominated by the efficiencies of the individual silicon-detector planes required by the algorithm, which were typically in the 90–95% range. Since we trigger only on events with tracks in both arms, each having hits in YS1 and YS4, the processor efficiency is proportional to the fourth power of the detector-plane efficiency. (A somewhat more elaborate design [8] could have eased these requirements and resulted in higher efficiency; due to time constraints we opted to build the simpler version described here.)

Figure 8a shows the the top- and bottom-arm impact-parameter distributions (before cuts) found by the processor. A major contribution to the widths of these distributions is the vertical size of the target. Figure 8b shows the effect of the impact-parameter cuts on these distributions. Figure 8c shows the y and z coordinates of vertices found by the processor for events that passed the processor cuts. The effect of the vertex z cut is visible; the triangular envelope is a consequence of the impact-parameter cut.

The processor was used for the E789 charm running [9] but not for the beauty running [10], which featured a considerably higher interaction rate and a different analyzing-magnet current. The beauty magnet settings were close to a singularity in the single-bend-plane calculation used for downstream tracks, and this degraded the processor rejection. As it happened, trigger rates for the beauty running were low enough to be recorded directly on tape, with no additional rejection from the processor needed.

References

[1] A. M. Halling and S. Kwan, Nucl. Instr. and Meth. **A333** (1993) 324.

[2] G. Charpak, Y. Giomataris, and L. Lederman, Nucl. Instr. and Meth. **A306** (1991) 439;
D. M. Kaplan *et al.*, *ibid.* **A330** (1993) 33;
G. Charpak *et al.*, *ibid.* **A332** (1993) 91.

[3] M. Adamovitch *et al.*, IEEE Trans. Nucl. Sci. **37** (1990) 236.

[4] J. A. Crittenden *et al.*, Phys. Rev. **D34** (1986) 2584.

[5] Y. B. Hsiung *et al.*, Nucl. Instr. and Meth. **A245** (1986) 338;
D. M. Kaplan, "A Parallel, Pipelined, Event Processor for Fermilab Experiment 605," in *Proc. Symp. on Recent Developments in Computing, Processor, and Software Research for High-Energy Physics*, Guanajuato, Mexico, May 1984, R. Donaldson and M. Kreisler, *eds.*, Fermilab, 1984, p. 31.

[6] W. Sippach, G. Benenson, and B. Knapp, IEEE Trans. Nucl. Sci. **NS-27** (1980) 578.

[7] J. A. Crittenden *et al.*, IEEE Trans. Nucl. Sci. **NS-31** (1984) 1028.

[8] C. Lee *et al.*, IEEE Trans. Nucl. Sci. **38** (1991) 461.

[9] M. J. Leitch *et al.*, Phys. Rev. Lett. **72** (1994) 2542.

[10] D. M. Jansen, *et al.*, Phys. Rev. Lett. **74** (1995) 3118.

Table 1: Typical charm running conditions and performance of E789 processor.

SM12 current (A)	Target material and dimensions (Δy[mm] $\times$ Δz[mm])	Interaction rate	Average clocks/event	Rejection factor	Efficiency
900	Be 0.16×1.8	1.4 MHz	239	24	0.74 ± 0.05
900	Au 0.16×1.8	4 MHz	250	5.3	0.70 ± 0.04
1000	Au 0.11×0.8	6 MHz	239	13	0.71 ± 0.04

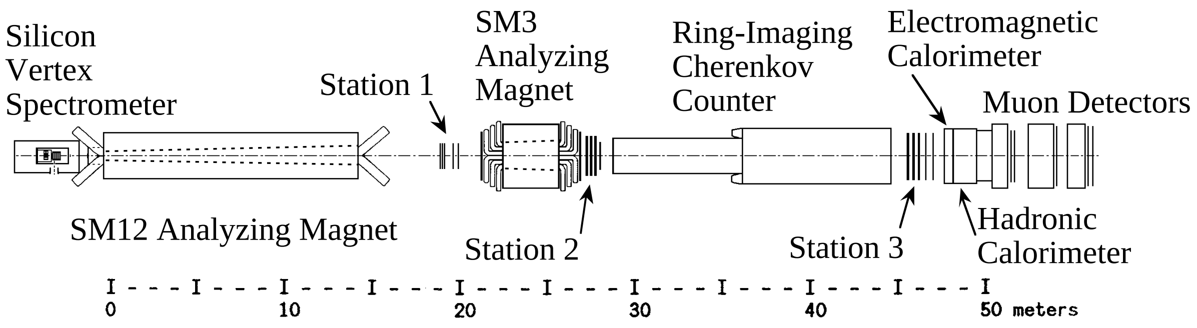

Figure 1: Plan view of E789 spectrometer.

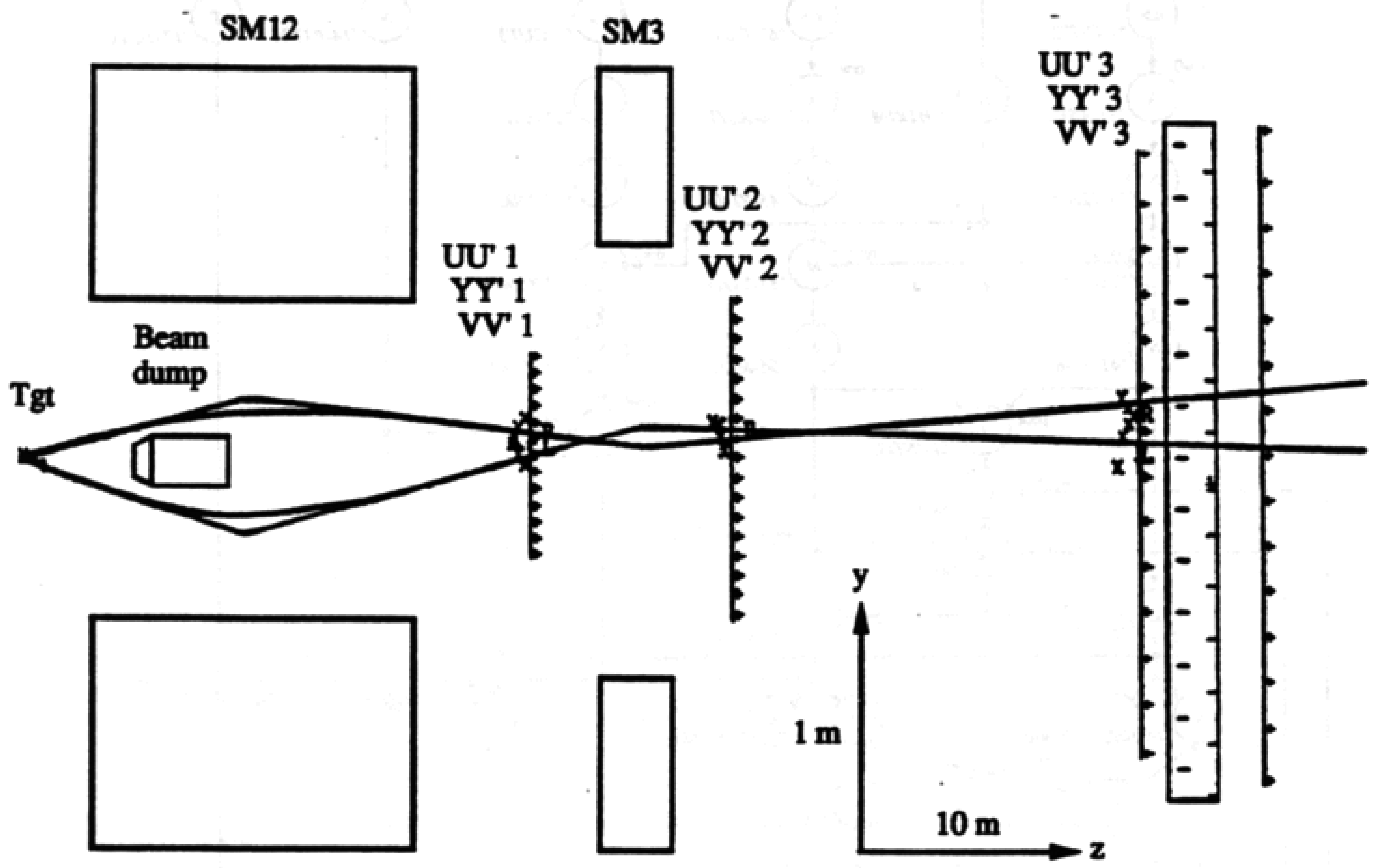

Figure 2: Bend (y-z) view of a typical event as reconstructed in the downstream spectrometer: "X"s indicate drift-chamber hits, and "L" and "R," hodoscope hits on the left ($x > 0$) and right ($x < 0$) sides of the spectrometer; both processor (single-bend-approximation) and off-line-reconstructed tracks are shown. Only 1/3 of the drift chambers measure in the bend view; stereo-plane hits appear offset from the track by an amount proportional to x.

Figure 2: Bend (y-z) view of a typical event as reconstructed in the downstream spectrometer: "X"s indicate drift-chamber hits, and "L" and "R," hodoscope hits on the left ($x > 0$) and right ($x < 0$) sides of the spectrometer; both processor (single-bend-approximation) and off-line-reconstructed tracks are shown. Only 1/3 of the drift chambers measure in the bend view; stereo-plane hits appear offset from the track by an amount proportional to x.

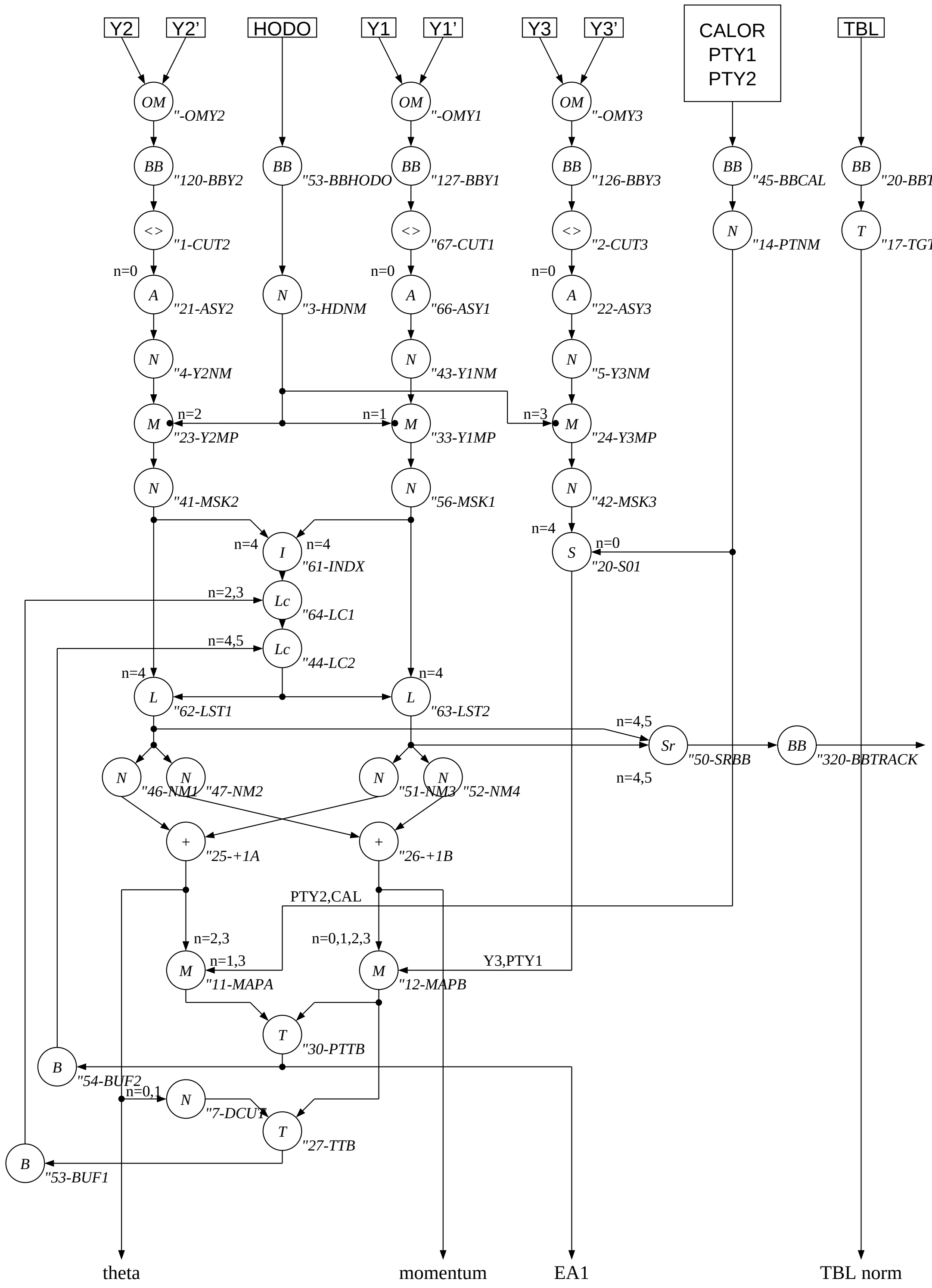

Figure 3: Schematic diagram of the E789 track processor.

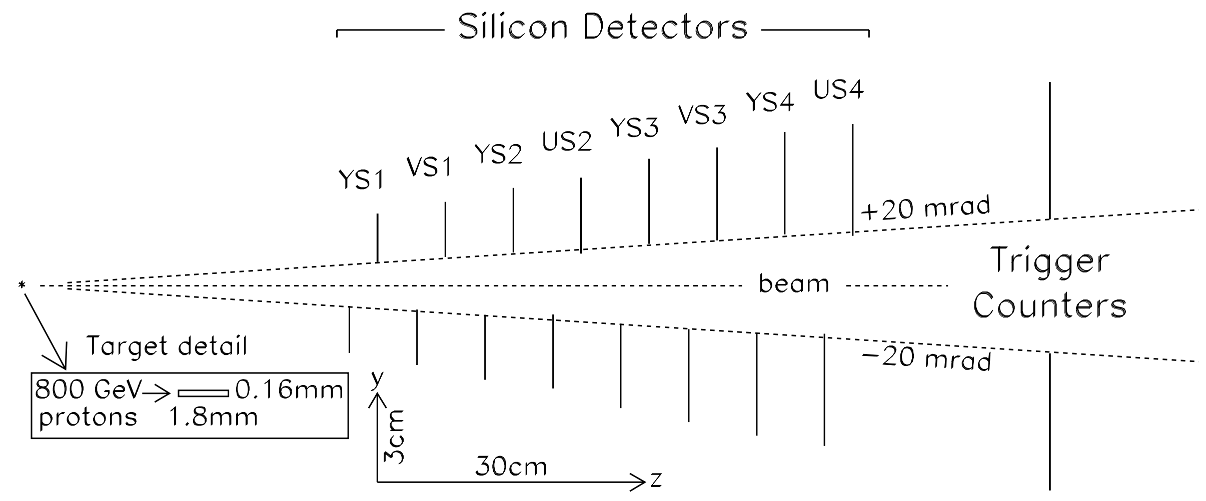

Figure 4: Elevation view of E789 silicon microvertex detector; target dimensions are indicated for the 900 A configuration.

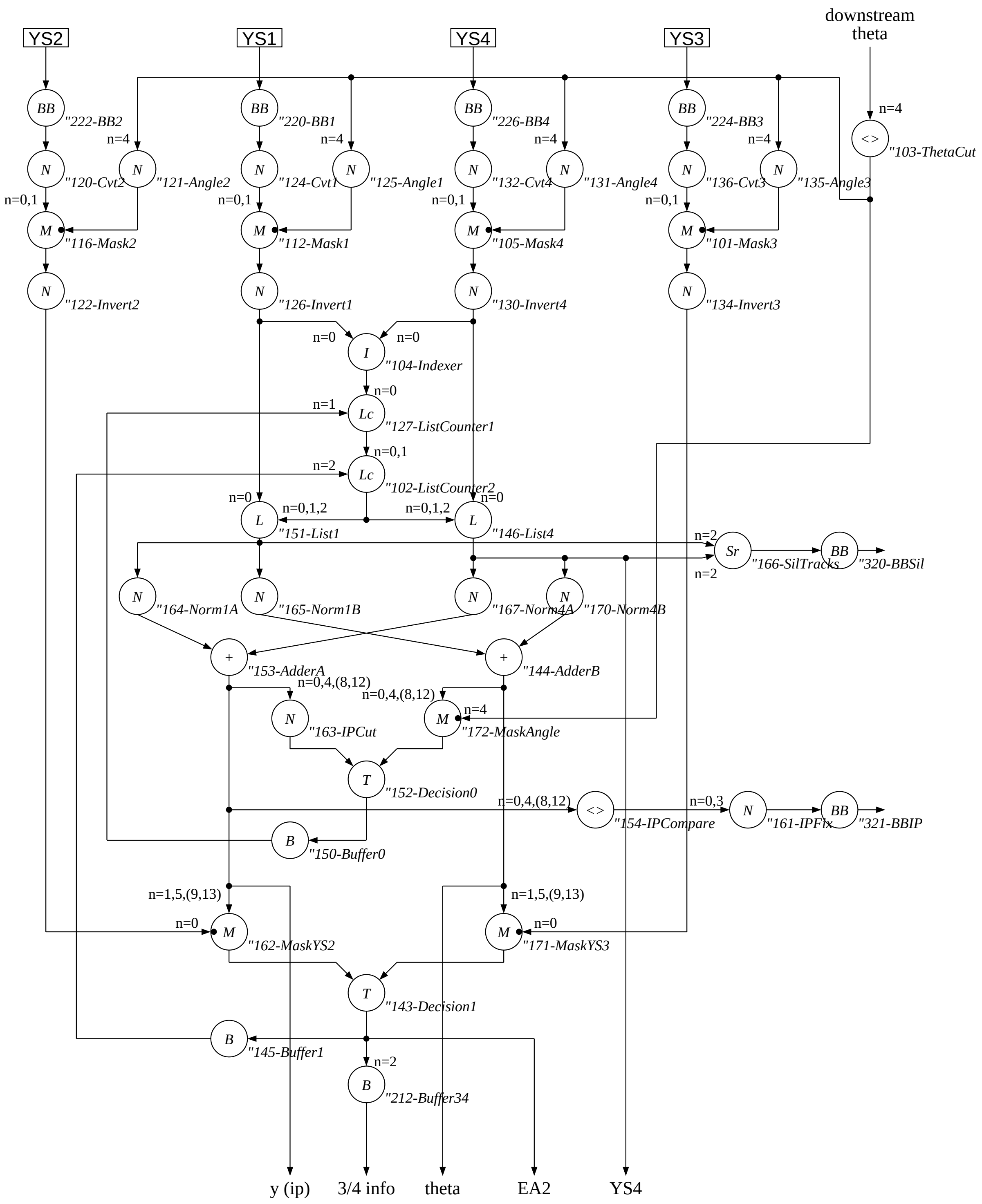

Figure 5: Schematic diagram of the E789 vertex processor.

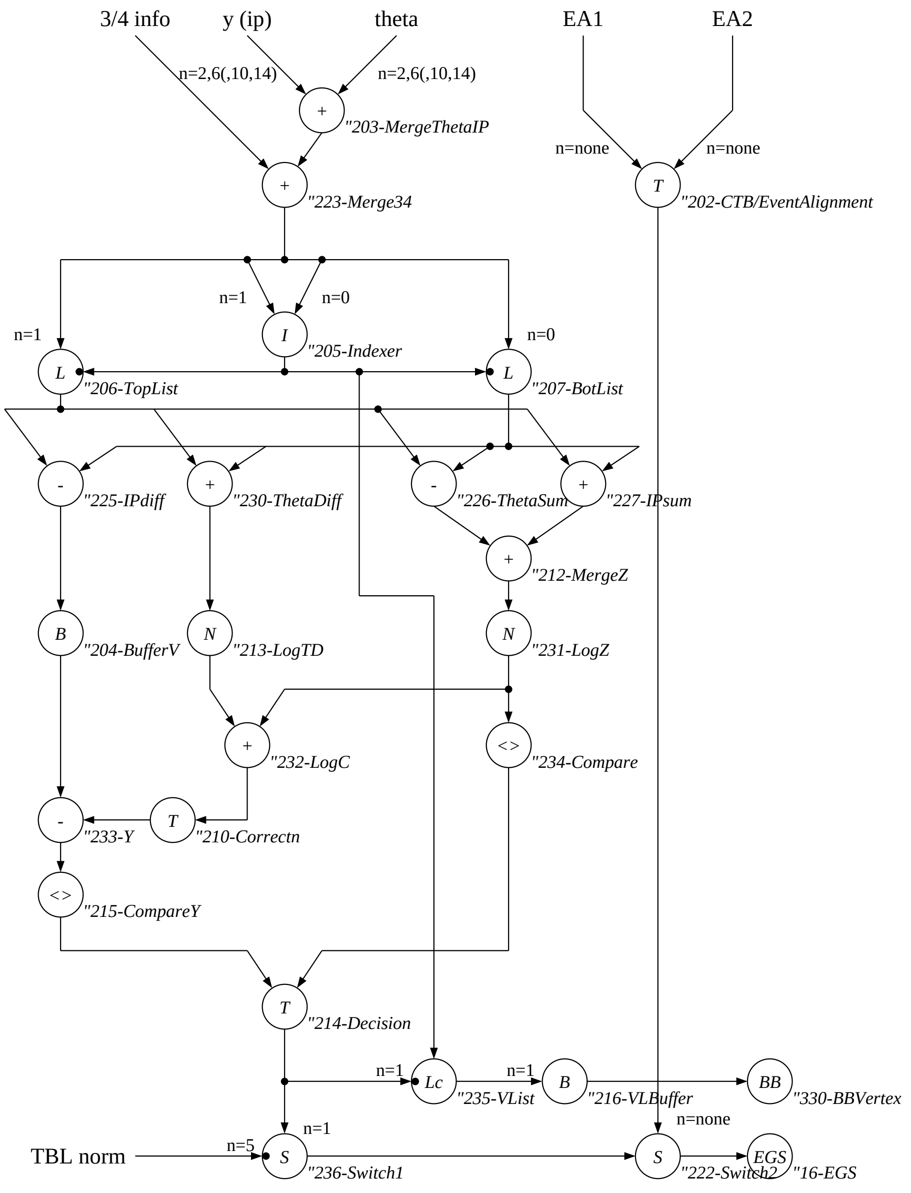

Figure 6: Schematic diagram of the "Vertex Endgame".

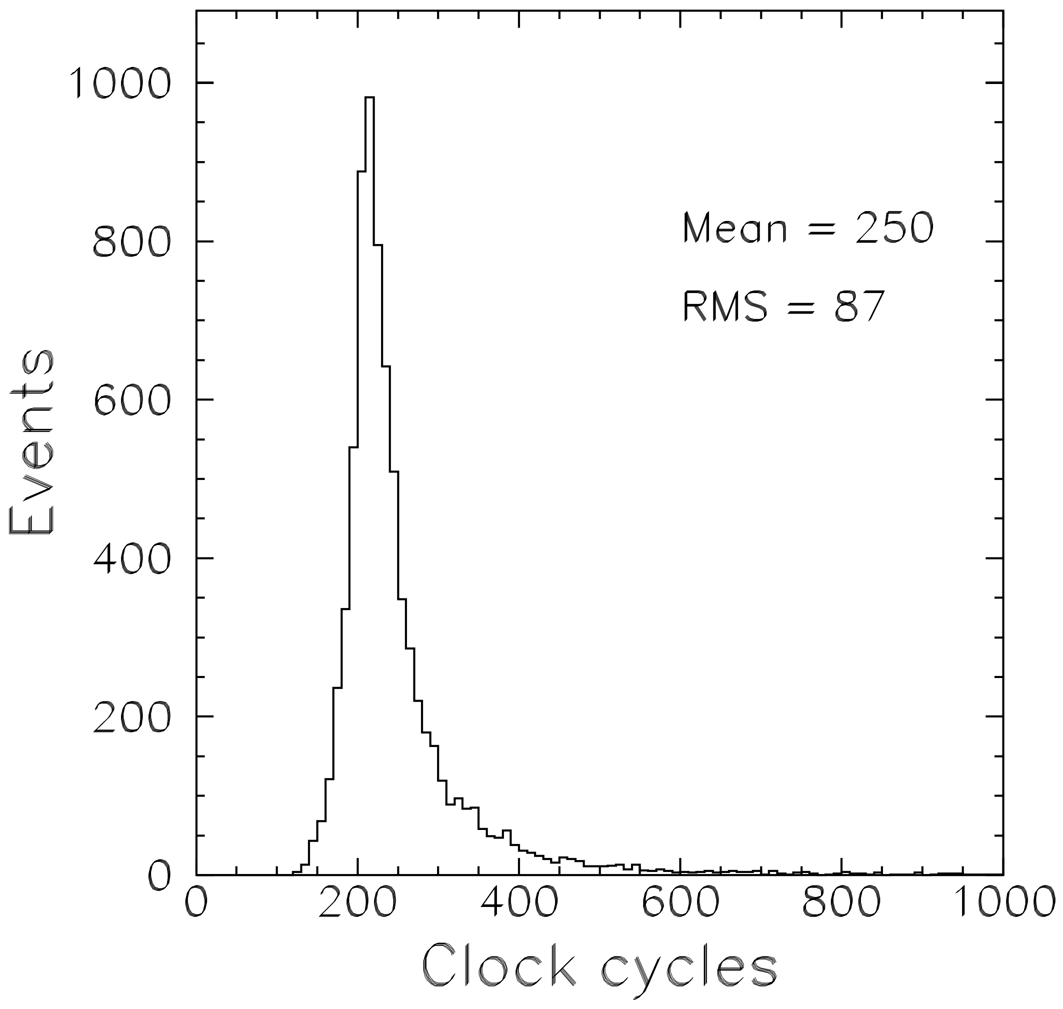

Figure 7: Distribution of number of clock cycles to process an event for the 900 A beryllium-target charm data; the mean value of 250 cycles implies an average processing time of 12.5 μs/event.

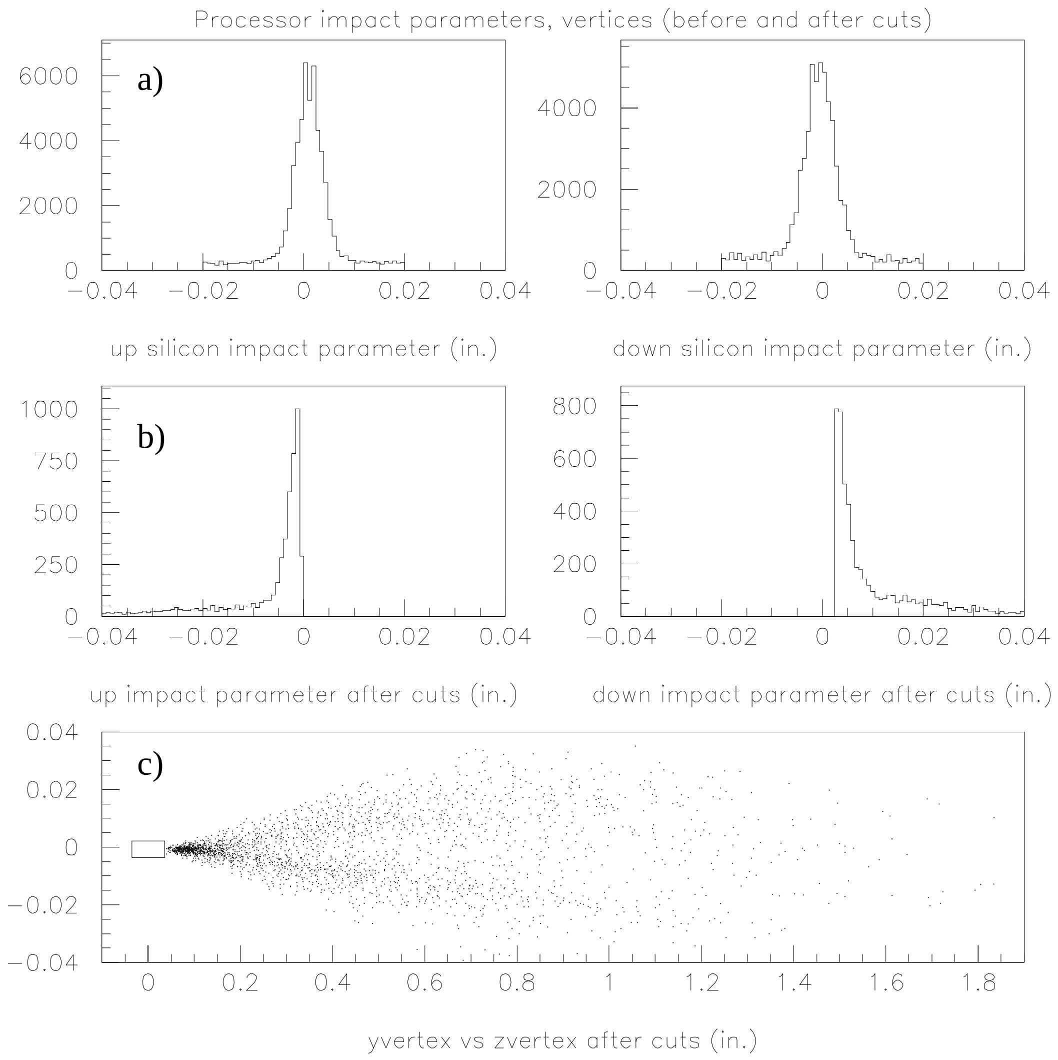

Figure 8: Processor impact-parameter distributions for top and bottom silicon arms for the 900 A Au running condition a) before and b) after cuts; c) y versus z of vertices found by the processor for events that pass the processor cuts; also indicated are the size and location of the target (rectangle).