A Pattern Recognition Scheme for Large Curvature Circular Tracks and an FPGA Implementation Using Hash Sorter

Jinyuan Wu and Z. Shi

Fermi National Accelerator Laboratory, Batavia, IL 60510, USA
jywu168@fnal.gov

Abstract

Strong magnetic field in today's colliding detectors causes track recognition more difficult due to large track curvatures. In this document, we present a global track recognition scheme based on track angle measurements for circular tracks passing the collision point. It uses no approximations in the track equation and therefore is suitable for both large and small curvature tracks. The scheme can be implemented both in hardware for lower-level trigger or in software for higher-level trigger or offline analysis codes. We will discuss an example of FPGA implementations using "hash sorter".

I. INTRODUCTION

In today's colliding detectors such as CMS, strong magnetic field causes large curvatures of tracks in the r-ϕ plane. Track recognition, especially for the middle and low transverse momentum ones, becomes more difficult due to the following two reasons:

1. The large curvature tracks are not restricted in small ϕ region. The tracks may hit many detector sections. Some tracks may even loop back in the opposite portion of the tracker.
2. The "high p_T" approximation for the track equation is not valid globally.

These challenges exist from lower-level trigger stage all the way to the higher-level trigger code and even in offline analysis software.

To overcome the difficulties due to large track bending, we investigated a global track recognition scheme using no approximation in the track equation. Consider a circular track passing through the collision point O as shown in Fig. 1 with initial angle α_0 and curvature $1/R$:

$$r = 2R\sin(\phi - \alpha_0)$$

In a small region of the detector, if the tangent angle α of the track segment (or "cluster", or "doublet") can be measured, the two track parameters, initial angle and curvature can be calculated very simply without any approximation:

$$\alpha_0 = 2\phi - \alpha \qquad c_0 = \frac{25cm}{R} = \frac{50cm}{r}\sin(\alpha - \phi)$$

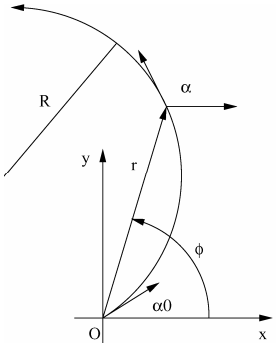


Figure 1: A circular track passing through the collision point O with initial angle α_0 and curvature $1/R$.

After calculating α_0 and c_0 of the doublets, the doublets are stored into an array of bins indexed by these two track parameters, or, one may visualize this as booking a 2-D "histogram". The doublets that belong to a track must all have common parameters. Therefore the doublets from one track will be primarily found in one bin (plus its neighbouring bins to account for the boundary effects). The grouped doublets are then sent to later stages for track reconstruction and multiple track separation.

The processes of binning and grouping of the doublets can be implemented in FPGA using various schemes. The implementation we choose in this document is based on the "hash sorter" [3] that we developed in the trigger system for the Fermilab BTeV experiment [1][2].

There are many techniques of measuring tangent angle α. For example, it can be measured by joining two hits in two detector layers. In this case, we can reach similar formula as in reference [4][5].

This scheme can also be viewed as a special case of the combinatorial version of the Hough transform[5].

II. PRINCIPLE OF TRACK RECOGNITION

We attempt to explain the track recognition scheme through a simulation example.

Imagine a simple detector layout containing 5 super layers. From the inner most super layer 0 to outer most super layer 4, the radii of them are 20, 35 50 75 and 100 cm.

In each super layer, two layers of silicon detectors separated by 1 cm measures the hit coordinate in the ϕ direction with 50 μm sensor pitch.

Generated tracks are shown in Fig. 2. One can see that some of the highly curved tracks with minimum radius $R_{\min} = 50cm$ barely hit all 5 super layers in our example.

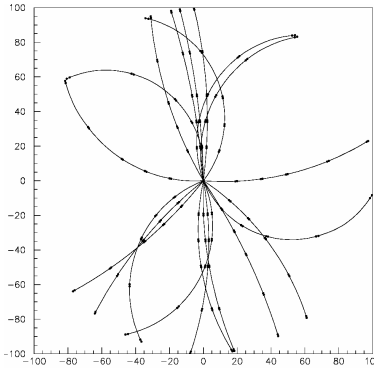
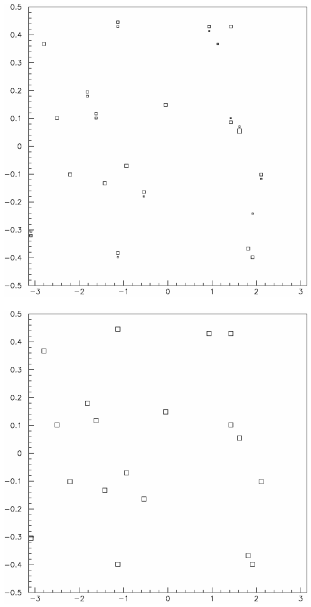


Figure 2: Tracks in a simulated example.

The close-by hits on the two detector layers in a super layer are paired together as "doublets". In our detector configuration, up to 5 doublets can be generated by a charged track. It is also possible that different tracks hit a detector layer too closely so that fake doublets are formed. We will include all possible combinations of the doublets at this stage.



(a)

(b)

Figure 3: Histograms used to explain track recognition scheme. Horizontal axis: initial angle (rad). Vertical axis: curvature c_0 of the circular track. (a) Raw histogram. (b) Non-zero clusters.

The tangent angle α of the track is coarsely calculated for each doublet. Then, through equations given above, the two track parameters, initial angle α_0 and curvature c_0 are found for each doublet.

The doublets are stored into bins indexed by these track parameters. We booked a 2-D histogram as shown in Fig. 3(a) to illustrate this process. Note that in the real implementation, not only the count of the doublets, but also the whole information of all doublets must be stored into a bin and retrieved out later. This is different than just booking a histogram.

The track parameters should be approximately equal for the 5 doublets generated by one track. Therefore, they are concentrated into clusters, which can be seen in Fig. 3(a).

Using simple PAW commands, we checked coincident of doublets in super layer 3 and ones in other super layers for each cluster. The result is shown in Fig. 3(b).

Comparing Fig. 3 (a) and (b), we can see that the entries in a cluster are collected together. Also, isolated fake doublets are eliminated.

Ideally, the doublets in each non-zero cluster should form a track. However, tracks may have similar parameters and may fall into same bin. In our example above, we do see two tracks with $\alpha_0 = 1.6$ and $c_0 = 0.06$ merging together.

The fake doublets may also "contaminate" good clusters. The overlapping will be resolved in later stage(s) when more accurate track parameters can be calculated.

Nevertheless, the number of combinations must be checked by the later stage(s) has been substantially reduced after this process.

There are two possible clustering schemes as shown in Fig. 4. The goal of clustering process is to start from a seeding doublet stored in a bin to find all matching doublets in nearby bins.

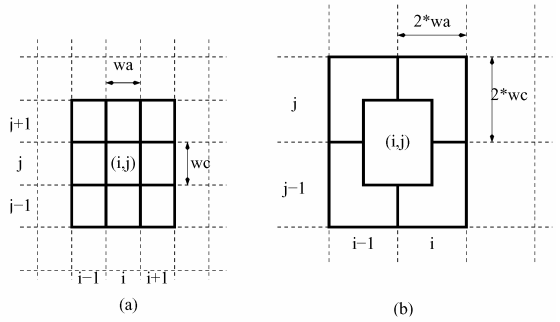


Figure 4: Two clustering schemes. (a) The 9-bin scheme. (b) The 4-bin scheme.

Assume the total width of the measurement errors for α_0 and c_0 are w_α and w_c, respectively. The two clustering schemes differ in the bin size and offset of the bins:

1. In the 9-bin scheme as shown in Fig. 4(a), the size of a bin in the parameter space is $w_\alpha \times w_c$. For a seeding

doublet in bin (i, j), the matching doublets can be found in 9 bins within $i \pm 1$ and $j \pm 1$.

2. In the 4-bin scheme as shown in Fig. 4(b), the bin size is $2w_\alpha \times 2w_c$. The seeding bins and the matching bins are shifted with each other by w_α and w_c in each direction. For a seeding doublet in bin (i, j), the matching doublets can only be found in 4 bins, i.e., (i, j), (i-1, j), (i, j-1) and (i-1, j-1).

The total phase space in the 9-bin scheme is smaller than the one in 4-bin scheme, which will have a better overlapping suppression. However, the number of storage bins must be accessed simultaneously in the 4-bin scheme is a lot less which is suitable for FPGA implementation. In Fig. 3(b) and remaining of this document, we will use the 4-bin clustering scheme.

III. FPGA Implementations Based on Hash Sorter

A. The Hash Sorter

The hash sorter can be visualized as a memory block organized into bins that are indexed by a key number, K. (see Fig. 5).

The data items with a particular key number are stored in the particular bin. The data items stored in the hash sorter can be retrieved quickly later.

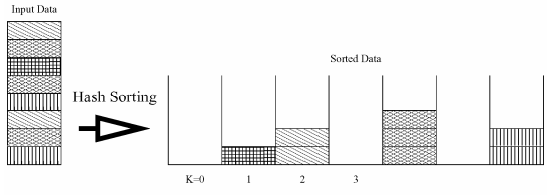


Figure 5: Hash sorting process.

We have developed the hash sorter in FPGA with considerations in the following aspects:

- The hash sorter is designed using pipeline structure so that the writing, reading and refreshing functions are performed in single clock cycle.
- Memories are organized as link lists for efficient usage. Therefore, there is no separate limit on number of data items a bin can store. The only limit is total number of items in all bins.

The logic of hash sorter itself does not require specific operation sequence. However, we normally first push entire set (such as a whole event) of data items into the hash sorter and then pop them out one by one. There is normally a refresh operation after the last popping operation of the previous set of data and before the first pushing operation of the new set.

The block diagram of the FPGA implementation for the track recognition using hash sorters is shown in Fig. 6.

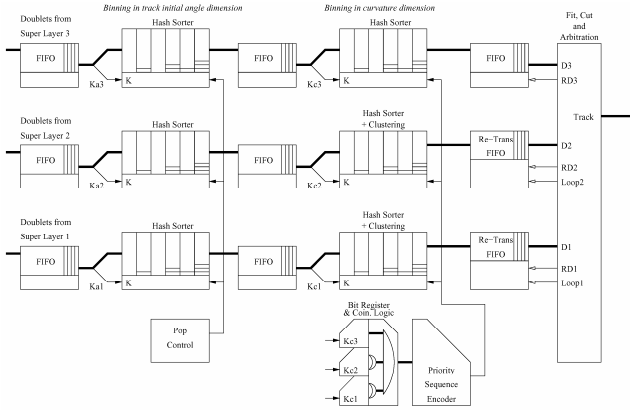


Figure 6: Block diagram of the FPGA implementation of the track recognition scheme.

B. Track Recognition Operations

We have chosen doublets from super layer 3, 2 and 1 in our track recognition example. The hits in other super layers can be collected after the track segments with hits from these three super layers are identified. We use super layer 3 as seeding layer since doublets in this layer provide more accurate measurement on curvature than the inner super layers.

The initial angle and curvature dimensions are divided into 256 and 64 bins, respectively, for the range of $\alpha_0 = [-\pi, \pi]$ and $c_0 = [-0.5, 0.5]$. It is possible to build a hash sorter with 256 X 64 = 16,384 bins to accommodate the 2-D binning with one stage of hash sorter. However, less silicon resource is used if it is done in two stages, one for each parameter. The initial angle hash sorters in the first stage will have 256 and curvature hash sorter 64 bins.

The doublets from super layers 3, 2 and 1 are sent into the input FIFO. The bin index numbers K_{a3}, K_{a2} and K_{a1} representing a small range in the α_0 dimension are picked out for pushing into the initial angle hash sorters.

After all doublets from an event are pushed into the initial angle hash sorters, they are popped out bin by bin. We may visualize these bins as columns in the 2-D histogram. The output doublets are buffered by a set of FIFO and then pushed into the curvature hash sorters.

The pushing operation into the curvature hash sorters are indexed by bin numbers K_{c3}, K_{c2} and K_{c1} in the c_0 dimension.

After doublets from a column (rather than from the entire event) are pushed into the curvature hash sorters, they are popped out bin by bin. These bins represent the rows in the 2-D histogram.

The curvature hash sorters for super layer 2 and 1 need additional internal logic to support clustering function.

As we mentioned above, the data storage in the hash sorter are organized as link lists. The refresh operation does not destroy the data in the previous set. Therefore, in the curvature hash sorters, while popping doublets in column i, the doublets in the previous column, column i-1 can still be retrieved.

The clustering in the c_0 dimension requires reading the row j and j-1, which are simply adjacent bins in the curvature hash sorters.

Normally, not all bins in the curvature hash sorters are occupied. We can use bit registers to record the occupied bins in each super layer.

Coincident logic is performed among the registers to eliminate random fake doublets. The non-zero bits after the coincident logic represent occupied clusters containing doublets from good tracks.

The locations of the non-zero bits are encoded sequentially as indices for the curvature hash sorters.

The doublets popped out from a cluster will be further checked by the "Fit, Cut and Arbitration" functional block.

Sometimes, there may be several doublets for a super layer in a cluster. Then multiple combinations must be checked. The simplest way to produce these combinations is to use re-transmittable FIFO blocks to loop over the doublets in a cluster for super layer 2 and 1.

IV. Test Designs and Silicon Resource Usage

We have test compiled the hash sorter in an Altera Cyclone [6] device (EP1C12).

The compiling results are shown in Table 1. As we can see, the Tiny Triplet Finder fit today's middle sized FPGA comfortably.

Table 1: Silicon Usage of the Hash Sorter

Reference Devices: **EP1C12** $118 (04/2004)			
	Numbers Needed	Logic Cells (ea.) (12,060)	Memory Blocks (ea.) (52)
Hash Sorter	6	143 (3%)	2 (4%)

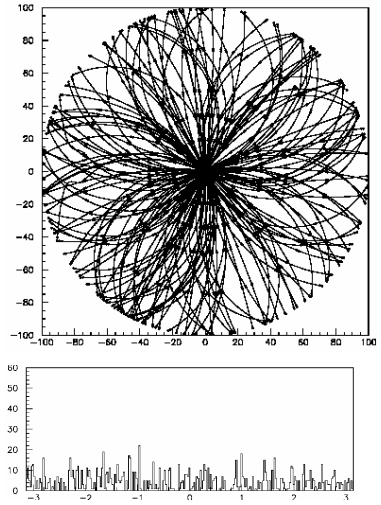
V. Simulation Results

Using a simple simulation, we have evaluated the track recognition scheme for events with numbers of tracks from 100 to 1000. The tracks are generated with even distribution in both α_0 and c_0. The minimum radius of curvature of the track is $R_{\min} = 50cm$.

We assume the z coordinate is not measured, but the region along z-axis of a detector hit is know, since the silicon sensor chips have a finite length. To reduce number of fake doublets we assume that hits can be distinguished into at least 8 regions in z direction effectively.

Programs are written in C to emulate the function shown in Fig. 6.

Cuts on parameters, both explicit and implicit ones, are set as loose as possible so that final missing track rate is controlled to less than 0.1%.



(a)

(b)

Figure 7: A simulation event with 200 tracks. (a) The event display. (b) The initial angle distribution.

A simulated event with 200 tracks and its initial angle distribution are shown in Fig. 7.

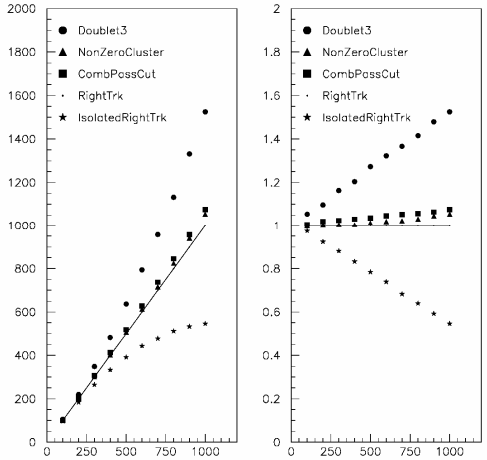


Figure 8: Simulation results with 100 to 1000 tracks/event. Circle: number of doublets on super layer 3. Triangle: number of non-zero clusters. Square: number of combinations that pass all cuts. Connected dot: number of correct tracks after cuts. Star: number of correct tracks with just one doublet per super layer in the cluster. The horizontal axis is number of tracks/event. The raw numbers and their ratios scaled by the numbers of tracks are plotted in the left and right graphs, respectively.

The hits are paired together as doublets. The range for pairing the doublets is chosen so that tracks with $|\alpha - \phi| < 45^\circ$ at super layer 3 will be included.

The simulation results are shown in Fig. 8.

One can see that extra doublets are generated when more tracks are involved in an event.

After two stages of hash sorting process, 3-fold coincident logic in cluster is checked for super layer 3, 2 and 1. The clusters with non-zero coincident are mostly real tracks.

Since in each of the non-zero clusters, there may be several doublets for a super layer, so the numbers of total combinations of doublets to be checked by the later stage(s) will be larger. We simulated a simple fit and cut stage to eliminate extra combinations. The numbers of combinations that pass the cut are in acceptable range.

Most of original tracks we generated in the simulation are found. The missing track rate is less than 0.1%.

VI. Other Fast Trigger Primitives

Useful trigger primitives can be established without full reconstruction of all tracks. An example is shown in Fig. 9.

The event shown here has 200 back ground tracks with the same distribution as in Fig. 7. Two "soft jets" are added, with 8 tracks each and same wide curvature distributions. The initial angles of the two groups of tracks are generated in narrow ranges: [1.00, 1.05] and [1.80, 1.85], respectively.

The "soft jets" are not easy to be seen in the event display. However, when the distribution of the initial angles of the doublets, including both correct and fake ones, is plotted, distinct features of the "soft jets" are obvious.

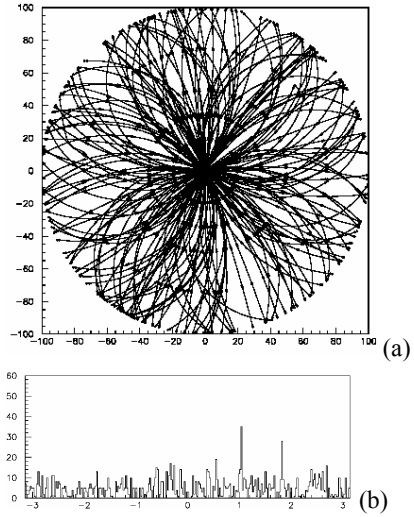


Figure 9: A simulated event with 200 back ground tracks and two "Soft jets". (a) The event display. (b) The initial angle distribution..

VII. Discussions

We should point out that this scheme is insensitive to the asymmetry of the detector configuration. As long as the angle of the doublet can be measured appropriately, this scheme will work in virtually any detector structures.

The offset of the collision point can be viewed as a special case of detector asymmetry. If the offset is known, it is not necessary to tune the collision point to the geometric centre of the detector, (if the centre exists).

The tangent angle measurement is normally very coarse. Therefore, multiple scattering will not affect the track recognition process in most cases.

Some times, tracks from secondary decay are detached from primary vertex. This type of tracks introduces errors essentially in the ϕ measurement. However, large impact parameter of the detached track can be tolerated since the baseline of ϕ measurement is a lot longer than the one of α.

Although we used silicon detector in our example, this scheme can be used in other detector types. There are mature techniques in drift chamber or straw detectors for measuring cluster parameters. The tangent angle of the cluster can be directly used. Sometimes, a cluster may yield two possibilities of the segments due to, for example, left-right ambiguity. In the track recognition process, the correct possibility will be chosen, and the "ghost" possibility can be eliminated, if the event is not too messy.

VIII. References

[1] Kulyavtsev et al., BTeV proposal, Fermilab, May 2000, BTeV-doc-66.

[2] G. Y. Drobychev et al., Update to BTeV proposal, Fermilab, March 2002, BTeV-doc-316.

[3] J. Wu, M. Wang, E. Gottschalk, G. Cancelo and V. Pavlicek [for BTeV collaboration], "Hash sorter: Firmware implementation and an application for the Fermilab BTeV level 1 trigger system," Presented at IEEE 2003 Nuclear Science Symposium (NSS) and Medical Imaging Conference (MIC), Portland, Oregon, 19-24 Oct 2003

[4] P. Schildt, H.-J. Stuckenberg and N. Wermes, "An On-line Track Following Microprocessor for the Petra Experiment Tasso." Nuclear Instruments and Methods, 178, 571, (1980)

[5] R. Fruhwirth et al., "Data Analysis Techniques for High-Energy Physics", 2nd ed., Cambridge, 2000

[6] Altera Corporation, "Cyclone FPGA Family Data Sheet", (2003)