Chapter 8

The BTeV Trigger Performance

8.1 Introduction

The proposed broad charm and beauty physics program of BTeV/C0 calls for a trigger that is very efficient for heavy-quark decay events which can be found off-line and is relatively independent of decay mode.

The baseline trigger is a multi-level trigger with several "branches" that can be independently prescaled and then combined into a global trigger. It is described in detail in chapter 4. The main B-physics trigger focuses on the identification of detached vertices at Level 1. The trigger must be capable of reducing the event rate by at least a factor of 100 at Level 1, and another factor of 25-50 at Levels 2 and 3.

Much work has been done on the Level 1 trigger since the EOI. We present in this chapter the results of studies of the baseline algorithm under more realistic experimental conditions including such effects as electronic noise, additional hits due to secondary hadronic interactions and photon conversions, and multiple interactions per crossing. We have also increased the thickness of the pixel detectors in our simulations to account for the material in support and cooling structures.

We present a status report on Level 1 trigger R&D including an overview of our hardware prototype and timing results for Level 1 trigger software running on digital signal processors. We also present alternative Level 1 designs in which the pixel detector is modified to reduce the material budget.

Preliminary studies of the Level 2 trigger indicate that the event rate can be reduced by a further factor of 10 through the improved measurement of track parameters and rejection of spurious tracks.

8.2 Advanced Level 1 Trigger Studies

The goal of the trigger is to reconstruct tracks and find vertices in every interaction, up to a luminosity of $2 \times 10^{32}\text{cm}^{-2}\text{sec}^{-1}$. It must be capable of reducing the trigger rate by a factor of about one hundred. We employ parallelism both at the subevent level and at the event level. The detector layout has "stations" consisting of three planes of pixel detectors. Processors organize the hits in each triplet of planes into minivectors or "station hits". Still at the subevent level, individual processors work on the station hits from a "ϕ-slice"[1] of the detector to form tracks. At the event level, tracks from individual slices are brought together in a vertex finder node. The corresponding algorithms for these various stages are described in detail elsewhere [1],[2]. Here we report on:

- Trigger diagnostics and 'software instrumentation';
- Performance studies of this baseline algorithm, in presence of noise backgrounds, both random and correlated;
- Performance studies on the baseline trigger in the presence of multiple interactions per crossing.

These studies are performed using a full hit-level simulation and a full pattern recognition algorithm. Although the final criteria for triggering will depend on the luminosity, the algorithm that we have implemented runs well on crossings which have as many as 3 interactions in them.

8.2.1 Trigger Diagnostics and 'Software Instrumentation'

The Level 1 trigger simulation uses MCFast output "hits" and the MCFast geometry as input data. This is a genuine reconstruction package including pattern recognition (i.e., Monte-Carlo generator information is never used to assemble tracks or vertices). However, back-pointers to this Monte-Carlo information are extensively used to monitor the performance of this pattern recognition. In addition, the Kalman filter from MCFast is also used to compare the crude errors on the track parameters to the nominal tracking resolution one could achieve with the detector. Finally, a preliminary version of the $B^o \to \pi^+\pi^-$ analysis presented in chapter 11, section 4, has been run in conjunction with the trigger simulation, allowing us to study and refine the trigger efficiency and rejection. We have measured the trigger efficiency on a variety of other states presented in chapter 11.

8.2.2 Uncorrelated Noise Study

Here we investigate the effects on the trigger algorithm of adding random noise to the pixel detector. This study was performed by throwing random hits from a Poisson distribution of average value <N>, uniformly distributed over the area of each plane. Electronic noise or neutrons could cause this kind of distribution.

Fig. 8.1 shows an event with no random noise. Fig. 8.2 shows the same event, but with an average of 25 noise hits distributed randomly over each of the 93 pixel planes.

[1] See Chapter 4 for definition of this particular event parallelism

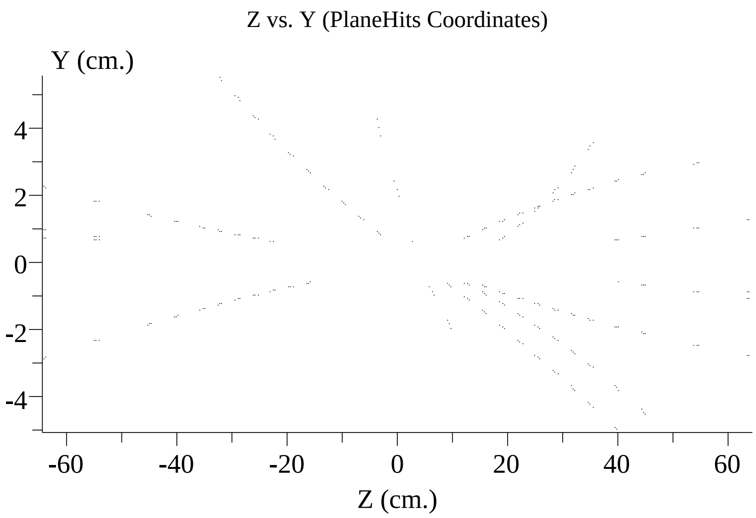


Figure 8.1: Pixel hits in Z-Y projection with no random noise from an MCFAST simulation

Fig. 8.3 shows the efficiency for minimum bias events, which we want the trigger to reject, and signal events (in this case events with a decay $B^o \rightarrow \pi^+\pi^-$), which we want the trigger to accept, plotted against the average number of random noise hits per plane. Three tracks with 3 σ detachment were required in this example.

We believe that it should be possible, and the experience with existing detectors bears this out, to achieve random noise levels below 10^{-5} if hot channel suppression is included in the pixel readout system. The result of our study is that the efficiency and rejection of the baseline trigger are preserved up to $<N> \approx 100$, which corresponds to a contribution of random background to the occupancy of 10^{-4}. We expect the detector to have a much lower noise level than this. However, it was observed that processing time did get somewhat longer when the number of hits rose above a mean of 25. A real estimate of the processing time in the DSP used to reconstruct the minivectors has been made and is described in section 8.6.

8.2.3 Correlated Noise Study

In this study, we used GEANT to simulate the events and turned on all its physics processes. This led to additional hits from hadronic interactions, photon conversions, decays in flight, and curlers (a.k.a loopers). Fig. 8.4 shows that average number of hits per plane rises by about 20% from $\approx$ 8 hits/plane to $\approx$ 10 hits/plane. A degradation of 1.5 is observed in the overall rejection capability of the trigger.

There are other sources of correlated noise. For example, beam-gas interactions or beam halo particles might leave long tracks in the detector. Although the time between crossings is long, there may still be background hits from previous crossings whose source and distribution we are not currently modeling. None of these is expected to be a serious problem but more work does need to be done to prove this definitively.

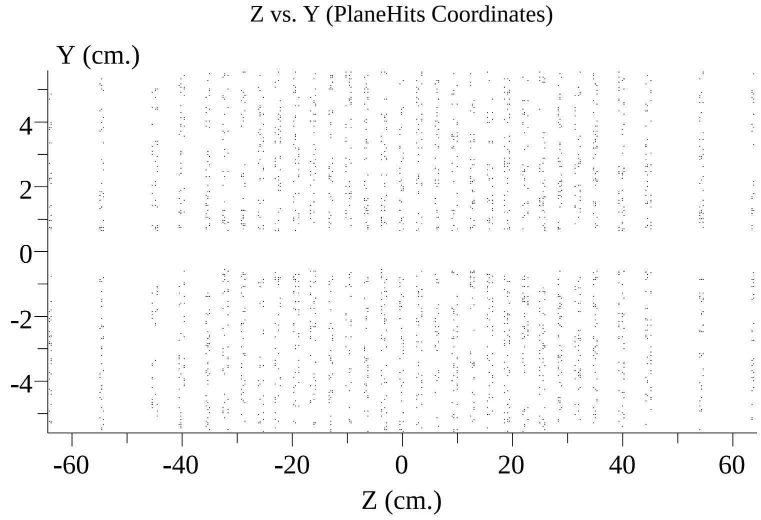


Figure 8.2: Pixel hits in Z-Y projection with an average of 25 random noise hits from an MCFAST simulation

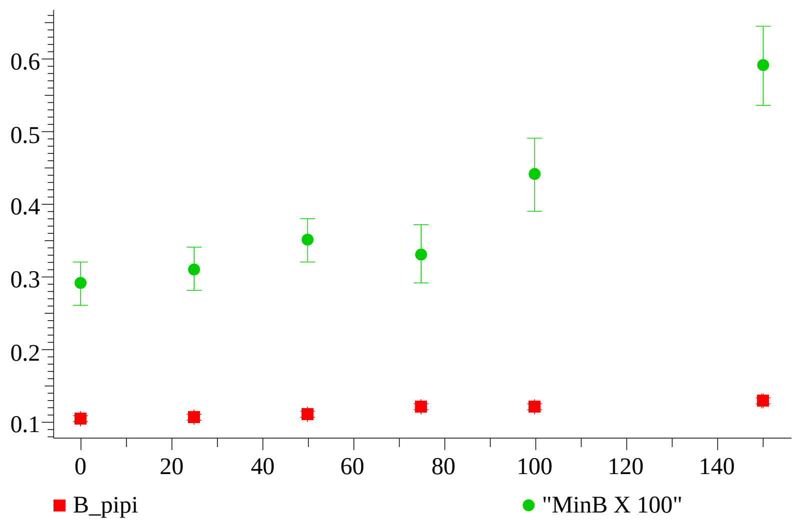


Figure 8.3: Efficiency vs average number of random noise hits for minimum bias and B events. Three detached tracks were required. The acceptance for B events includes all events, even those not falling into the geometrical acceptance.

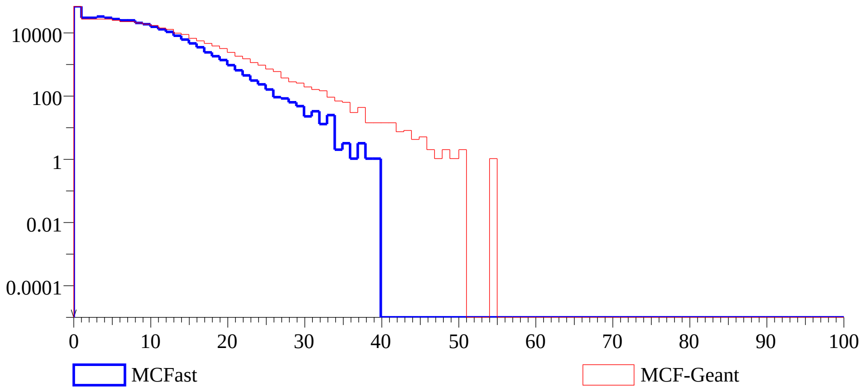


Figure 8.4: Hit multiplicity for minimum bias events in GEANT simulation and MCFAST simulation

8.2.4 Multiple Interactions per Crossing Study

While BTeV will begin running at a luminosity of $5 \times 10^{31}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, in order to achieve the desired statistical precision, we will eventually want to run at a luminosity of at least 2×10^{32}. Because the Tevatron will have only 108 bunches, this means that the experiment, and in particular the trigger, must be able to handle multiple interactions per crossing. The lower luminosity creates an average of 0.5 inelastic interactions per crossing, while the higher luminosity generates an average of 2.0.

As shown below, even in the presence of multiple interactions, the trigger continues to give good efficiency for states with B decays while providing good rejection against minimum bias events because the occupancy of the pixel vertex detector is very low and the luminous region of the Tevatron is long.

In order to achieve good results when there are multiple primary vertices, we had to make only a small refinement to the trigger algorithm: we put an **upper limit** on the impact parameter (with respect to the primary vertex) which is used to define secondary vertex track candidates. This prevented tracks not used in estimating a given primary vertex (but associated to this interaction) from automatically appearing as 'detached' when compared to another primary vertex. Here we present only a brief summary of the results of an extensive study [3].

Performance on Single Interaction Crossings

The average number of tracks per interaction (primary and secondary) which pass through at least three stations of the pixel detector is 30 for minimum bias events and 38 for $B^o \rightarrow \pi^+\pi^-$ events (which will be called B events from now on). After a set of cleanup cuts on the tracks, the average multiplicity of tracks that the trigger uses in the primary vertex search is ≈ 12 for minimum bias events and ≈ 20 for B events. Two additional cuts are placed on tracks which are candidates for use in the primary vertex: they must have a transverse

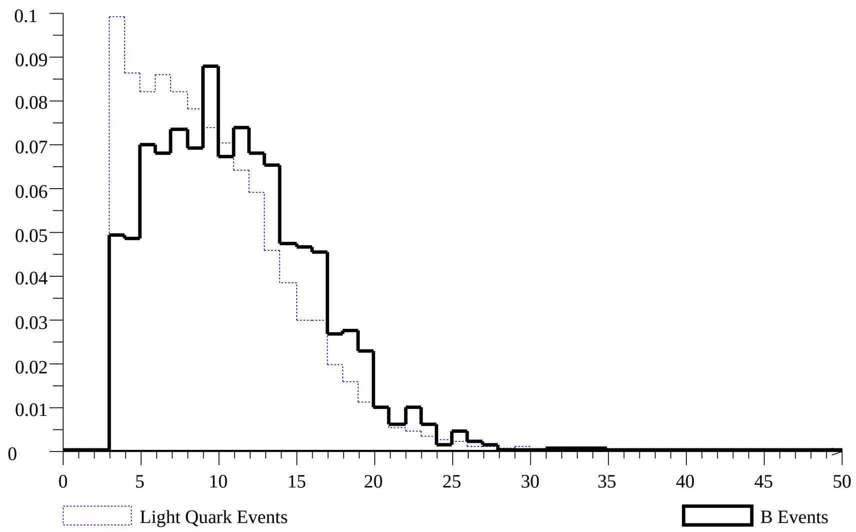


Figure 8.5: The track multiplicity distribution in the primary vertex, for single interaction crossings. The average is 9.6 for minimum bias events, and 11.1 for $B \to \pi\pi$.

momentum of less than 1.0 GeV/c; and they must intersect the nominal beam line within the luminous region. With these requirements, the resultant multiplicity of primary vertex tracks found by the trigger algorithm in a crossing with exactly one interaction is given in Fig. 8.5 for minimum bias and B events. The Z resolution (i.e. along the beam axis) of the primary vertex is given in Fig. 8.6. Note that this distribution has typically more tails for the B sample because of pattern recognition failures in events with multiple vertices: if many tracks from a B decay are mistakenly taken as primary tracks, the B decay vertex could be wrongly taken as the primary. Nevertheless, since some tracks are still likely to be detached, these events could pass the trigger.

To form a primary vertex requires at least 3 tracks. As a consequence of this cut and the transverse momentum cut, the probability to find a second, false, primary vertex is very small, only 0.05%, in minimum bias events. The 'leftover' track multiplicity after primary vertex formation is very different for minimum bias and B events. If one applies an additional detachment cut on the normalized impact parameter $\frac{\sigma_b}{b}$ of 3 and a transverse momentum cut $P_t > 0.5$ GeV/c one has reconstituted the analysis we presented in 1997 [1]. To deal with multiple interactions, we added the requirement that the 'detached track candidate' not be **too far from the primary vertex** – the impact parameter b must be less than 0.2 cm from the primary. This cut also allows us to reject tracks from K_s decays. In addition, further cleanup cuts on these detached tracks are now applied. For instance 4 station hits (or minivectors) on a track are required. Then, one obtains the distribution in Fig. 8.7.

Table 8.1 shows the trigger acceptance for minimum bias events and B events for single interaction crossings. Our goal for Level 1 was to accept fewer than ≈1% of minimum bias events, which is achieved in all conditions. Other triggers based on global properties of the

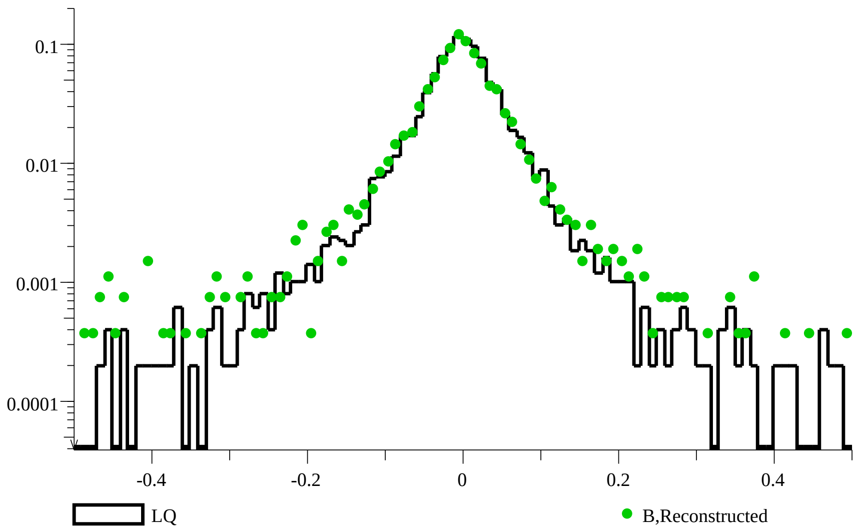


Figure 8.6: The primary vertex Z resolution, for single primary vertex events. The width (σ) of the core of these distributions is approximately 270 μm , while the r.m.s is about 760 microns, reflecting the presence of tails in the distributions.

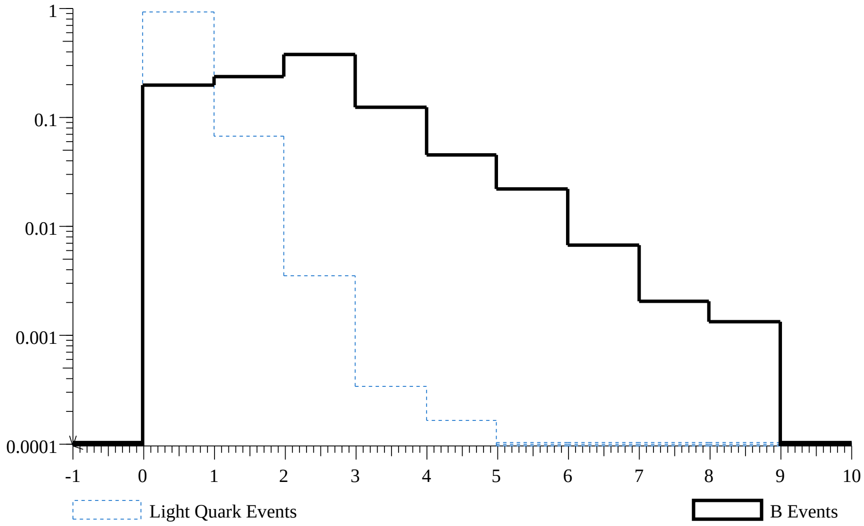


Figure 8.7: The detached, $P_t > 0.5GeV/c$ track multiplicity for both samples. The average track multiplicity for minimum bias (B) is 0.6 (1.5).

Table 8.1: The trigger acceptance (%) for single interaction crossings, for the minimum bias sample and the $B \to \pi\pi$ sample, versus selection criteria (number of tracks detached from the primary, detachment cut, P_t on each detached track, and impact parameter upper limit in cm). The errors are statistical only.

Criteria	Minimum Bias	$B \to \pi\pi$
2 detached tracks ($4\sigma; P_t > 0.5$ GeV/c; $b_t < 0.2$)	0.9 ± 0.1	54.9 ± 0.9
3 det. tr. ($3\sigma; P_t > 0.5$ GeV/c; $b_t < 0.2$)	0.05 ± 0.02	18.5 ± 1.2
3 det. tr. ($3\sigma; b_t < 0.2$; $Sd_{Pt} > 1.0$)	0.17 ± 0.04	27.0 ± 1.5

vertex and its ensemble of detached tracks have been studied. For instance, the total scalar transverse momentum Sd_{Pt} summed over all these detached tracks could be used, as in the third line of Table 8.1.

Performance on Multiple Interaction Crossings

Fig. 8.8 shows the number of primary vertices reconstructed by the trigger algorithm for an average of 0.5 and 2.5 interactions per crossing, corresponding roughly to luminosities of 0.5×10^{32} and $2.5 \times 10^{32}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$. These plots suggest that the number of reconstructed vertices is well correlated to the number of generated vertices even in these busy events, which can be seen as follows: The distribution can be qualitatively understood as a Poisson distributed number of interactions with a fixed vertex finding efficiency of about 62%. The efficiency is low because the trigger is deliberately made insensitive to very soft collisions such as quasi-elastic and low multiplicity interactions which are unlikely to contain B's.

Fig. 8.9 shows that the primary vertex Z resolution is independent of the number of primary vertices per crossing. No degradation of this resolution is observed as the luminosity increases. Each interaction can be studied independently because the average distance between primary vertices is much larger than the resolution.

Fig. 8.10 shows the 'multiplicity distribution' of detached tracks for different numbers of reconstructed vertices when the average number of interactions per crossing is ≈ 0.5 (corresponding to a luminosity of $\approx 5 \times 10^{31}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$), and a luminosity 5 times this amount. The main point is that there is not a large tail of events in which the number of detached tracks begins suddenly to become troublesome as the number of reconstructed vertices increases. We observe that the efficiency for detecting B events does not degrade as the number of actual or reconstructed vertices increases. Note also that the detached tracks must be associated to the same primary vertex in order to satisfy the trigger.

The total yield of B events per unit time does not degrade as additional detached tracks are found in other primary vertices, provided of course one does not explicitly reject these multiple primary vertex crossings. If one does, the yield follows what one would expect from the simple Poisson law. The probability of triggering on a crossing with at least one inelastic interaction is shown vs luminosity (in units of $10^{32}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$) in Fig. 8.11. (Here, we have used a trigger requirement of at least 3 tracks each missing the primary by $\geq 3\sigma$.) This

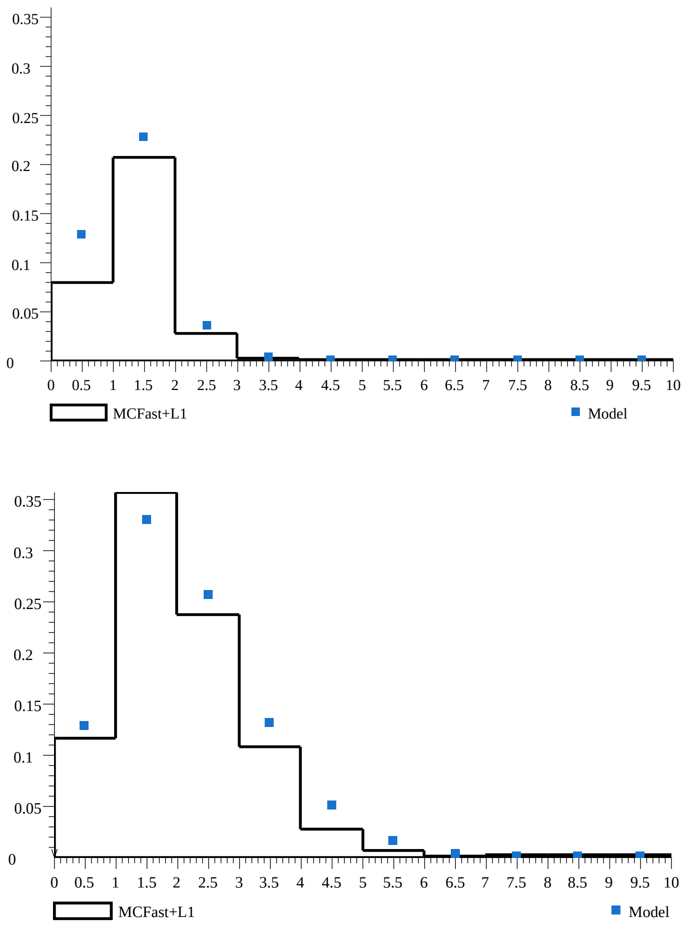


Figure 8.8: The Level 1 primary vertex multiplicity distribution running at luminosities of 5×10^{31} (top plot) and 2.5×10^{32} (bottom plot). These histograms are normalized to the total number of 132 ns crossings in the run. For instance running at 2.5×10^{32}, the probability to reconstruct one and only one primary vertex is about 35 %. Also shown is a simple model assuming a fixed vertex geometrical acceptance and reconstruction efficiency of about 62 %.

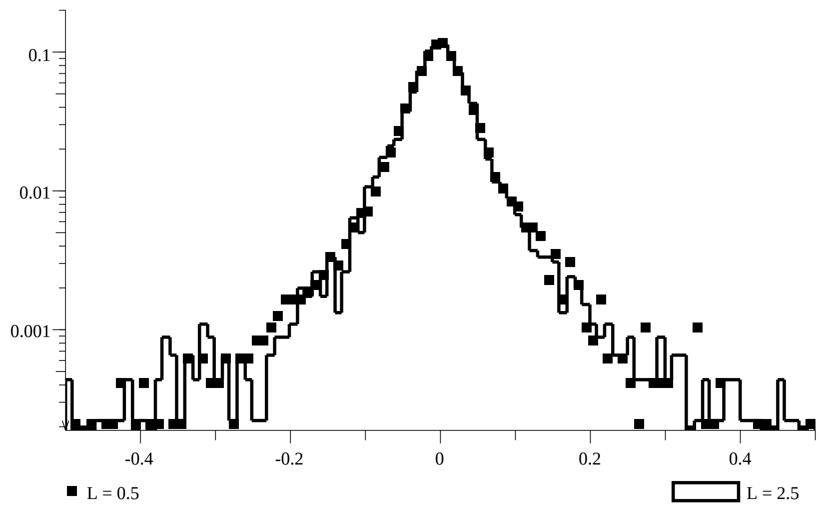


Figure 8.9: The Z position primary vertex resolution plot, running at luminosities of 5×10^{31} and 2.5×10^{32}.

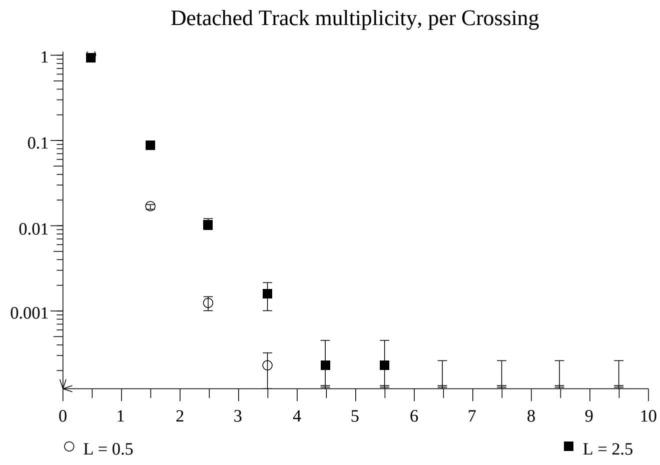


Figure 8.10: The detached (3σ , $P_t > 0.5$) track multiplicity distributions for light quark events running at a luminosities of 5×10^{31} and 2.5×10^{32}. These are normalized to the total number of crossings in the run: the probability to observe 1 detached track at $L = 5 \times 10^{31}$ is $\approx 1.6\%$, and at $L = 2.5 \times 10^{32}$, it is $\approx 8.5\%$

probability is about twice as large if the two detached track at 4σ trigger is used. Ideally, one would expect the yield per unit time to increase linearly and it does increase roughly linearly as long as one does not restrict the number of interactions too severely. The probability for triggering on crossings with inelastic interactions does increase but stays below 1% up to a luminosity of 2.5×10^{32} if we allow at most three reconstructed vertices. For the two tracks at 4σ, in the worst case, the probability is slightly above 1%. There are many additional requirements (such as the presence of a true secondary vertex rather than tracks missing the primary) that can be imposed at Level 1 to reduce the trigger rate even more.

Our results on the performance of the baseline trigger in the presence of multiple interactions per crossing are quite encouraging. In essence,

- The trigger rate is not dominated by unresolved multiple vertices up to at least 2×10^{32} .
- A Level 1 rejection factor of about one in a hundred is possible with a Level 2 bandwith that can be achieved (see chapter 9).
- One gains sensitivity by running above an average of 1 interaction per crossing provided we accept crossings with up to three interaction vertices.

Thus, we believe that it is possible to achieve a rejection against minimum bias events better than 1/100 at Level 1 up to a luminosity of 2×10^{32} with good efficiency for B events we feel confident to reconstruct offline. One can work above 2×10^{32} provided one can handle the data rate of falsely accepted minimum bias events into the Level 2 trigger or one can reduce the rate into Level 2 by adding more requirements on Level 1. However, there is more work to be done in making the simulation more realistic and in refining the algorithm. Also, we need to estimate the required CPU power at each stage of the reconstruction, to balance the data flow in the pipeline. We have started looking into the queuing problems at the front end of the triggering system. We look forward to reporting further on this in the future.

8.3 Efficiency and Rejection of the BTeV Level 1 Vertex Trigger for Selected Physics States of Interest

The results are documented in the above section and in part 3. Here we summarize our results. The rejection factor on light quark events for the two detached track trigger is about 1/100 for single interaction crossing and is proportional to the luminosity if multiple interactions per crossing are accepted. We summarize the efficiency for this trigger in table 8.2. In all cases, the opposite b-hadron decays following the standard QQ decay table (defined in part3) .

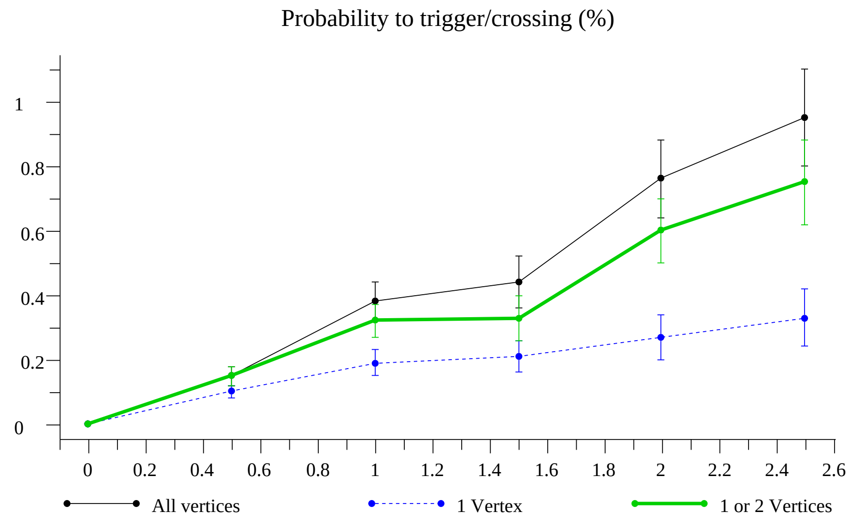


Figure 8.11: The absolute probability for triggering per 132 ns beam crossing. The trigger is based on three detached tracks pointing to the same longitudinal hemisphere, with $Sd_{Pt} > 1.0$. For both plots, the abscissa axis is the luminosity in units of 10^{32} cm^{-2} s^{-1}. The solid line is for events with any number of primary vertices; the dotted line, for events with one and only one reconstructed vertex; and the thick gray line for events with 1 or 2 reconstructed vertices.

Table 8.2: The two detached track at 4σ trigger efficiency for various B decays.

Final State	Efficiency (%)
$B \to \pi^+\pi^-$	55
$B_s \to D_s K$	70
$B^- \to D^0 K^-$	60
$B^- \to K_s \pi^-$	40
$B^0 \to K^+\pi^-$	54
$B^0 \to J/\psi K_s$	50
$B_s \to J/\psi K^*$	69
$B^0 \to K^*\gamma$	40

8.4 Other Level 1 triggers

While we have emphasized the "vertex/impact parameter" trigger because it is our most crucial and most ambitious trigger, we will also have other triggers at Level 1. These will include muon, minimum bias, and (possibly) electron triggers and perhaps some triggers derived from the same hardware that is used for the vertex trigger but optimized for different physics.

8.5 Status Report on Level 1 trigger R&D and trigger prototype

The Level 1 vertex trigger is a key feature of the BTeV detector design. A proposed scheme for the vertex trigger has been worked out by the Penn group in collaboration with D. Husby at Fermilab. The hardware design for this system is described in detail in references [4] and [5]. Although the design can be implemented with DSPs (digital signal processors) and FPGAs (field-programmable gate arrays) that are available today, an intensive R&D program is needed to develop the complete design, test the viability of the system, and perhaps modify some of the hardware components as new hardware becomes available. The R&D program has three main components:

- Simulations;
- Hardware prototyping;
- Software development.

Simulations of the trigger are an important aspect of the trigger R&D program. We generate pixel hits using the MCFast simulation program, and process the hits using a software implementation of our Level 1 trigger algorithm. This procedure allows us to test the trigger algorithm, and to refine the algorithm as we include more realistic running conditions for the experiment. We also use trigger simulations to learn more about the performance of the Level 1 trigger under "normal" and "extreme" running conditions, and to develop the software that is needed for our trigger prototype (see Fig. 8.12).

Although much of the testing of the trigger algorithm can be done with simulations, there are tests that are best done with a hardware trigger prototype. The trigger prototype is a single-board module that implements most of the data paths and processing steps used in the trigger system. It has FIFOs to load data generated by the MCFast simulation program, and it includes four hit processors, four farm (track or vertex) processors, and the merging and switching circuitry that connects them.

The prototype is used to test data flow through the trigger system. There are three primary data paths that are tested:

1. Single station-quadrant to track finder.

   In this test, the prototype is used to test the path from one quadrant of a pixel station to the farm processors that reconstruct tracks. Pixel hits from the three planes of a station-quadrant (see Fig. 8.13) are divided into eight ϕ slices, and fed to an array of eight hit processors. The prototype is used to test paths through four of the eight processors. These hit processors execute the segment-finding algorithm that finds minivectors in the vertex trigger. The minivectors are sent through the sorting network to the farm processors. Each farm processor runs the standard track-finding code to produce track records that are written to a FIFO, which is used to read out and verify test results. The results are used to investigate hit-processor throughput, study the effects of extreme event sizes, and test the ability of the switching network to sort hits.

2. 31 stations to track finder.

   This tests the flow of data for all 31 pixel stations that make up a single ϕ slice. The four hit processors execute code that mimics the data generated by 31 hit processors. We use the results from this test to study the performance of the switching network under a full load.

3. 32 track finders to vertex finder.

   This tests the flow of data from 32 track processors to the vertex processing farm. The four hit processors execute code that mimics data generated by the track processors for all ϕ slices. We use the results to study the switching network, and the vertex finding algorithm.

For each of the three tests, a performance monitor keeps track of statistics on the data paths that are used during each cycle. These statistics are sent to a host computer for graphical display and analysis. The host computer is also used to configure, monitor, and control each of the processors and FPGAs in the prototype.

The design of the trigger prototype entails both hardware and software development. The design of the hardware has been completed. The software that is being developed for the prototype consists of code that is used to implement test scenarios, and the three programs that implement the Level 1 vertex trigger algorithm. The first of these three programs runs on hit processors (each hit processor is a Texas Instruments DSP) and finds track minivectors. The second and third programs run on farm processors (each farm processor is an Analog Devices DSP) and perform track and vertex reconstruction, respectively. The Texas Instruments and Analog Devices DSPs differ in their hardware architecture. Each DSP requires software for data flow and buffer management, and the software to implement the Level 1 trigger algorithm.

The software development for the DSPs encompasses three different aspects:

- Data flow and buffer management;

- Trigger algorithm;
- Optimization.

The code that handles the data flow and buffer management is specific to the hardware architecture of each DSP, and depends on the design details of the trigger prototype. The code that implements the vertex trigger algorithm is based on the trigger code used in our MCFast simulations, but significant modifications are required to produce code for the DSPs. Compared to the trigger code in MCFast, the software for the DSPs is optimized to take advantage of the DSP architecture. This optimization requires assembly language programming to achieve the performance (see timing results in the following section) that we require for the Level 1 vertex trigger.

8.6 Results on Baseline Level 1 Algorithm Timing on Digital Signal Processors

With 100 gigabytes of pixel data coming into the Level 1 trigger every second, a beam crossing every 132 ns, and an average of 2 interactions per crossing, the time required for each pipeline step in the trigger is critical. Timing issues are studied by performing a variety of trigger simulations. These include Monte Carlo studies of the pattern recognition and reconstruction algorithms, and simulations of the trigger hardware. Furthermore, the timing of both the trigger hardware and software algorithms will be extensively tested with the trigger prototype described in the previous section. In this section, we describe timing studies for algorithms on DSPs (digital signal processors).

The BTeV trigger utilizes two types of DSPs: a fixed-point DSP, and a floating-point DSP. The fixed-point DSP is made by Texas Instruments and belongs to the TMS320C6X family of DSPs. The floating-point DSP is an Analog Devices ADSP-2106x SHARC. The trigger code for both the Texas Instruments DSP and the Analog Devices DSP is developed with the aid of simulation software provided by each company. We also use this software to obtain the timing results presented in this section.

Simulations of the Level 1 design show that the first stage of the trigger, the segment finder, requires the most computational power compared to other parts of the trigger. A large number of operations are performed by the segment finder, due to the number of combinations of pixel hits that must be sampled to find the hits that define a mini-vector. To address this problem, we introduce considerable *parallelism* in the trigger architecture. This is a key feature of the Level 1 trigger design. The parallelism begins with the organization of the data readout from the pixel detectors. Pixel hits are read out in parallel for each quadrant of a pixel plane. The hits from the three planes that belong to a pixel station are brought together in a *quadrant processor*, which represents one of 124 identical circuit boards that make up the first stage of the trigger. Additional parallelism is introduced by subdividing the pixel hits in a quadrant into 8 ϕ slices, for a total of 32 ϕ slices per pixel station. The hits in each ϕ slice are processed in parallel by one of the 992 TMS320C6X DSPs

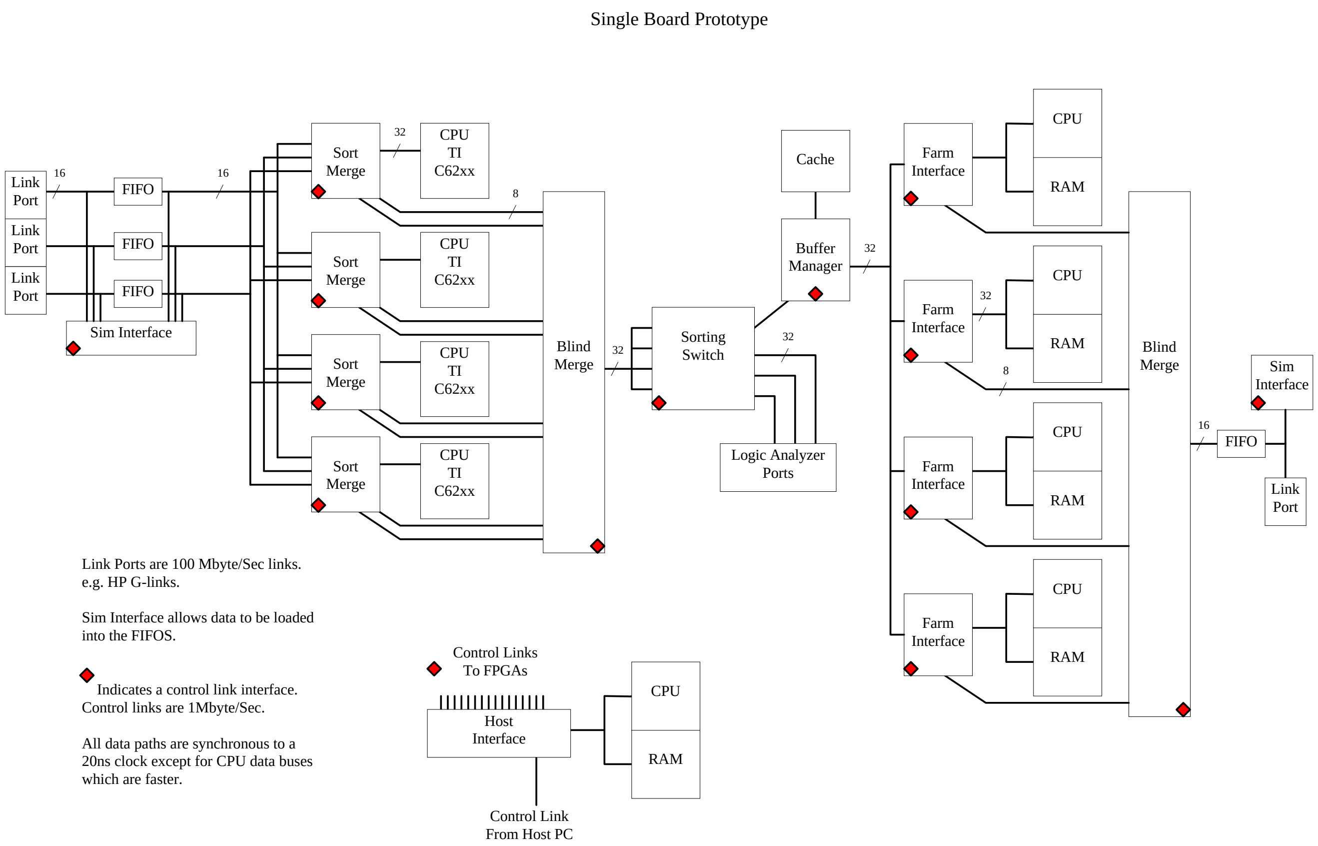


Figure 8.12: Schematic of Single Board Trigger Prototype

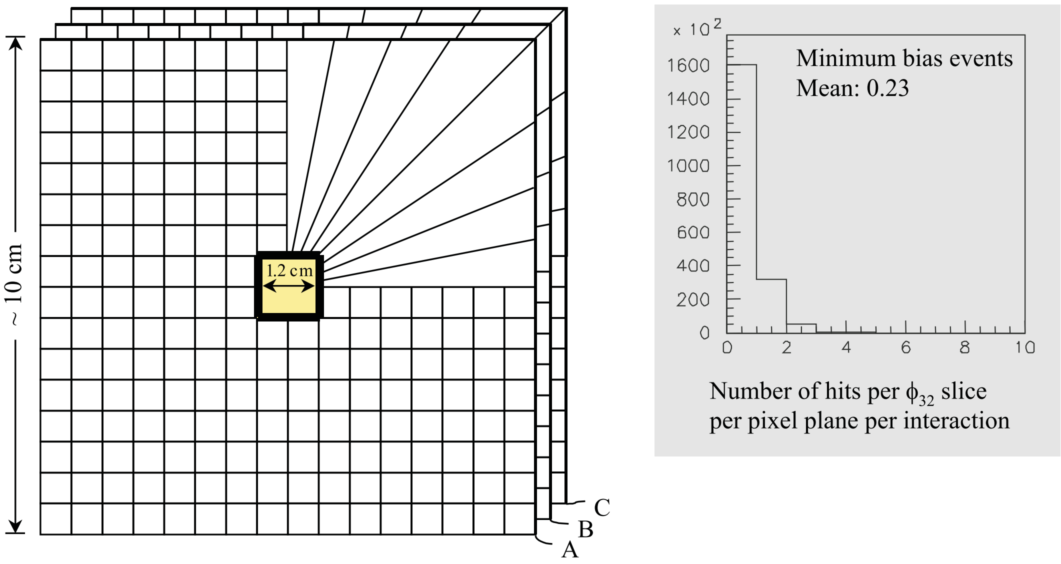


Figure 8.13: Trigger view of one quadrant in a pixel station with planes A, B, and C. The histogram on the right shows the number of hits for each plane per ϕ slice for minimum bias events.

(a total of 992 DSPs for 31 pixel stations, each with 32 ϕ-slice DSPs). Fig. 8.13 shows the "trigger view" of one quadrant, and shows that the subdivision of hits into ϕ slices reduces the mean number of hits per pixel plane to a value of 0.23 hits per ϕ slice for minimum bias events.

Even with the small number of hits per ϕ slice, the timing for the segment finder is critical. Table 8.3 shows timing results for the segment finder algorithm developed for the TMS320C6X DSP. In the table we choose four cases to present our results: (i) For a given ϕ slice and with three planes (A, B, and C) in a pixel station (as shown in Fig. 8.13), we have a single hit in each plane for one track passing through a pixel station. This is the (1,1,1) case in Table 8.3. (ii) For two tracks passing through the same ϕ slice we have two hits per plane, or (2,2,2). (iii,iv) With three and four tracks in the same ϕ slice we get the (3,3,3) and (4,4,4) cases, respectively.

The timing results for these four cases are divided into three categories that represent different levels of optimization for the segment finder algorithm. The first category involves *C* code that has *not* been optimized (the code is compiled without any optimization), and we use this to establish a reference measurement for subsequent measurements involving different levels of optimization. For the second category we allow the *C* compiler for the TMS320C6X DSP to optimize the algorithm, and for the third case the programming is done directly in assembly language. Although programming in assembly language is more cumbersome, we can optimize the code so that all hardware components in the DSP are performing useful operations most of the time. Our timing results in Table 8.3 show that

Number of hits per ϕ slice in planes (A,B,C)	Time for non-optimized segment finder(ns)	Time for compiler optimized segment finder(ns)	Time for assembler optimized segment finder(ns)
(1,1,1)	1915	1630	135
(2,2,2)	5705	3670	315
(3,3,3)	14015	7560	1340
(4,4,4)	28915	13840	2885
weighted average	530	395	37

Table 8.3: Timing results in nanoseconds for the segment finder algorithm running on the TMS320C6X DSP.

the *C* compiler for the TMS320C6X DSP is not very effective at optimizing the segment finder algorithm; the compiler achieves only a minor performance gain for the (1,1,1) case, and at best a factor of two improvement for cases with more hits in a ϕ slice. Table 8.3 also shows that we are able to achieve almost an order of magnitude improvement in the timing by optimizing the algorithm in assembly language. This does not include additional improvements that can be achieved for the (3,3,3) and (4,4,4) cases, which have not yet been fully optimized.

To obtain an estimate of the average time required to find mini-vectors in minimum bias events, we compute a weighted average by using the results in Table 8.3 weighted by the distribution of hits in Fig. 8.13. We use a value of zero for the (0,0,0) case, since the DSP does not perform any operations when there are no hits in a ϕ slice. The average time for each level of optimization is shown in the last row of Table 8.3. Although these results are somewhat oversimplified, they do indicate that an assembly language implementation of the segment finder algorithm is necessary so that the first stage of the trigger stays within a budget of 132 ns per beam crossing. The average time of 37 ns for the segment finder is encouraging, since it suggests that we are close to having a viable implementation for the segment finder in our Level 1 design.

The track and vertex processor timing are also under study. We hope to present results soon.

8.7 Options to the Baseline Level 1 Vertex Trigger

8.7.1 An Alternative Using Two Plane Stations

The choice of three-plane pixel stations in the baseline detector is dictated by the baseline trigger design. That trigger uses the three planes to make minivectors that can then be projected to a small portion of the next station, which substantially reduces the combinatorics. Any attempt to use fewer planes per station will have to cope with larger combinatorics when starting the search for tracks.

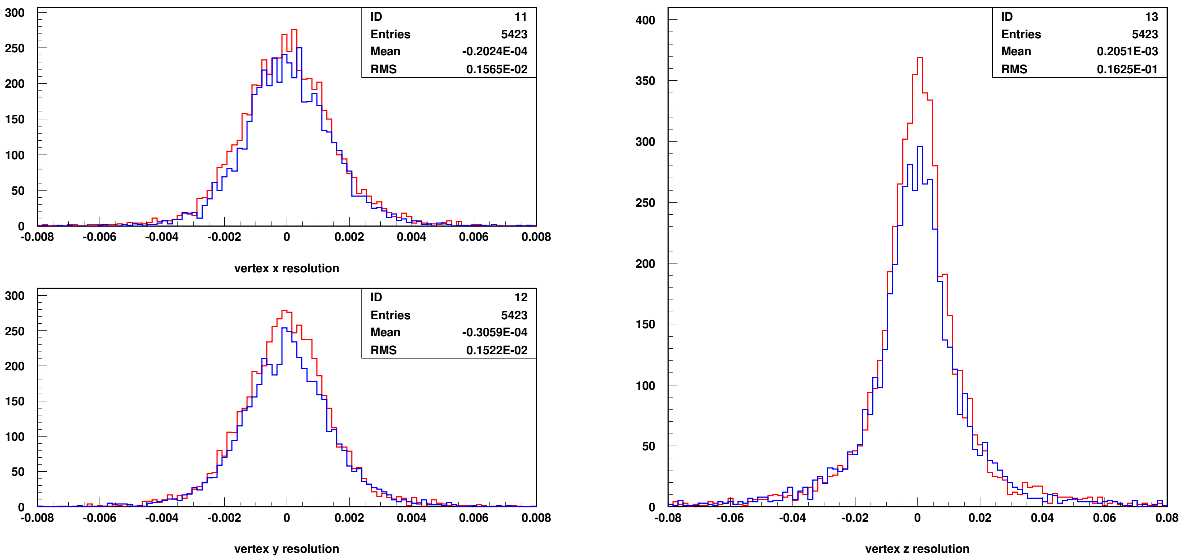


Figure 8.14: Comparison of the vertex resolution for $B^0 \to \pi^+\pi^-$ for two-plane pixel stations and three-plane pixel stations–left top x, left bottom y, right z.

The removal of one plane per station has advantages of lower cost, a simpler detector, and less material with no loss of physics capability. Simulations using the identical pixel detectors arranged in 31 stations of three planes and 34 stations of two planes showed almost no difference in the effective resolution and efficiency of the two detectors on charged particle states. A full offline analysis was done for several interesting physics channels, $B^0 \to \pi^+\pi^-$, $B_s^0 \to \Psi K^*$ and $B_s^0 \to D_s^+\pi^-$. In the case of $B^0 \to \pi^+\pi^-$ the secondary vertex resolution is a good figure of merit and is shown in figure 8.14. The vertex resolution is almost identical in the two detector configurations. The 5% difference in efficiency may be recoverable with optimization of station placement.

The time resolution for the B_s decays $B_s^0 \to \psi K^*$ and $B_s^0 \to D_s^+\pi^-$ (used to study B_s mixing) is also unaffected by the choice of two-plane or three-plane stations. Simulations show that for the first decay mode the time resolution is 44 fs for both two-plane and three-plane geometries, and for the second decay mode the two-plane is slightly better at 54 fs versus 56 fs for the three-plane geometry. These results can be understood by the fact that multiple scattering is an important contributor to the resolution, and the reduction in the material compensates for the lost measurements.

i. Algorithm Design and Simulation

The basic pattern recognition procedure is still finding straight lines. However they are found across stations instead of within a single station. Because we cannot point into the next station we will have to search over a large section of the next station. It is efficient to restrict the search to quadrants of each station and we do so.

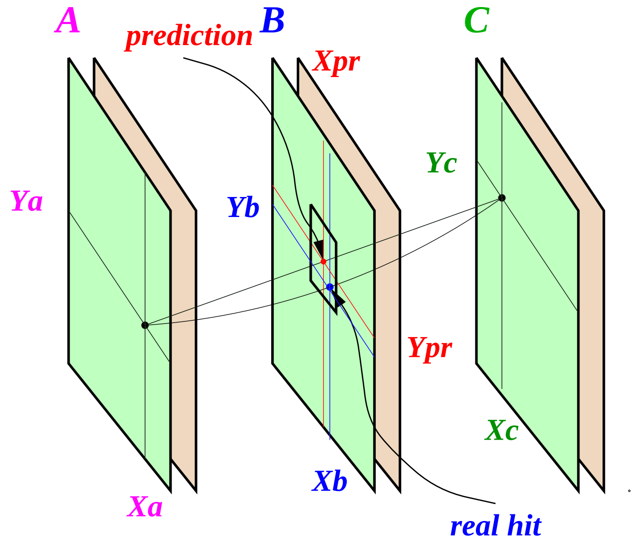


Figure 8.15: Sketch of the basic interpolation algorithm for the planes with fine pitch in x.

We choose three adjacent stations and try to find straight lines by using the hits in the two outer stations called A and C to predict the hit in the middle station B. The prediction is made in both the x and y coordinates since the pixels have 2D information. The algorithm is pictured in Figure 8.15:

$$\begin{aligned} \left|\frac{x_A + x_C}{2} - x_B\right| &< \Delta_x \\ \left|\frac{y_A + y_C}{2} - y_B\right| &< \Delta_y \end{aligned}$$

The size of Δ_x and Δ_y is dictated by the resolution of the pixel detectors, the multiple scattering, and for the y view the bending due to the magnet. In terms of pixel widths, the windows are $\Delta_x = 3$, $\Delta_y = 2$ for x planes and $\Delta_x = 2$, $\Delta_y = 10$ for y planes. The windows are 6 pixels in x planes and 20 pixels in y planes out of 125000 pixels in a quadrant, so the background suppression is excellent. Since the window sizes are larger than the pixels we do not need the full analog information at this stage but only the address of the cluster peak or similar information.

The three stations used to find line segments are overlapped, with each station sending its data to three different line segment finders. At this stage the ensemble of hits is in fact a collection of linked lists. These lists can easily be matched to tracks without combinatorial searches. Track segment fitting is currently less developed, but the momentum and intercept estimators used in the baseline trigger is a fall back option.

ii. Feasibility of Implementing the Design

The basic pattern recognition ability of the pixel detector is more than adequate to find line segments due to real tracks. The main concern is whether the hardware will be fast enough to keep up with the data rate. At a mean of two interactions per crossing we expect to see an average of 8 hits per quadrant and 95% of all quadrants in all events have fewer than 16 hits. If we implement the algorithm using a microprocessor performing loops then the time to complete the search will go like N^3, so we must introduce some parallelism into the problem.

The basic calculation consists of an addition, a multiplication (which can be done as a bit shift), a subtraction, and a test. All of these calculations can be done as integers. Programmable Logic Devices (PLDs, also called FPGAs) contain a large number of simple logic units that can perform hundreds of these calculations in parallel. This is significantly more parallelism than in the baseline trigger design allowing this design to handle the larger combinatorics.

The design can be broken into three sections: an input section that receives and buffers the hits; an interpolation section that does pattern recognition; and an output section that collects line segments to pass on to the next level of processing. The times to complete the various sections scale differently with event complexity. The timing in the input and buffering section scales as the number of hits divided by the number of input paths. If the time for this section is large due to many hits, the hits can be brought in in parallel as long as there are sufficient I/O pins on the chip. The timing of the interpolation section scales with the number of bits used in the interpolation, so it is independent of the size of the event. However, there will be a loss of efficiency should the event exceed the maximum capability of the processor. The timing of the output section scales with the number of line segments found, which makes it stable against random noise and low energy backgrounds.

A design of the line segment finder has been done for a Xilinx PLD. The basic idea is to read all of the hits from the A and C stations into one PLD. These are then used to feed a matrix of interpolation units which will do each hit from the B plane sequentially (see Figure 8.16). We use bit-serial arithmetic to shrink the data bus width and better utilize routing and logic resources. The time do this comparison goes as 1 clock cycle per bit of hit coordinate.

A single chip can check four B plane hits against the interpolations from all A and C plane hits in a quadrant in parallel. Simulation predicts this will take about 120 ns to handle 12 bits of hit information at a clock period of 10 ns. In order to keep up with the data rate 2 chips per station quadrant per view would be required.

The size of a PLD is measured in either gates or configurable logic blocks (CLBs). The design of the interpolation nodes requires 5 CLBs per node so a PLD that can handle a maximum of 16 hits at a time would require $16 \times 16 \times 5 = 1280$. Currently we target a Xilinx PLD that has $8464 = 92 \times 92$ CLBs, so 4 arrays of interpolation nodes can be placed with 40% of the chip left for control logic, input buffers, etc. The new Virtex family chips from Xilinx are more capable and available now.

The I/O capabilities of these PLDs are more than adequate. Seventy-two I/O pins are needed to read in the 12 bits of each coordinate for a hit from each of three planes. The

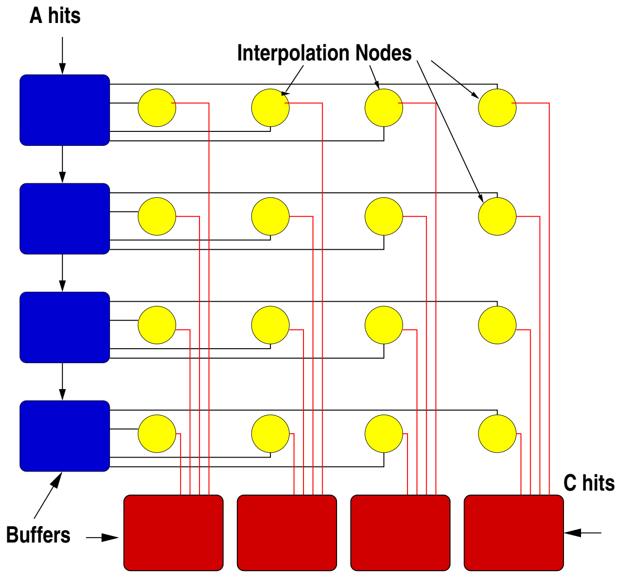


Figure 8.16: Block diagram showing the matrix of interpolation nodes and input buffering for the outer planes.

PLD under consideration has 448 I/O pins. There are more than enough I/O pins to bring in two or three times more hits in parallel. Hits from A and C planes are buffered on the PLD from one event while the previous event is analyzed.

A simple cross check using a Lucent PLD confirms the basic performance. The interpolation portion of the line segment finder was run through the Lucent device simulation and equivalent results were found. The design should work in a variety of devices so while present devices are adequate in both speed and I/O capabilities, we can delay the final choice to take advantage of better and/or cheaper devices as they become available.

8.7.2 The 'Megapixel Trigger'

The Megapixel Trigger is an alternative Level 1 trigger scheme. It benefits from concepts that have been explored in the context of the BTeV baseline vertex trigger and the 2-plane vertex trigger described in the previous section. Like the 2-plane trigger, the Megapixel Trigger is designed for a pixel system with two planes per tracking station. Moreover, the track reconstruction is initiated in a similar manner by finding track segments that span three adjacent pixel stations. Unlike the 2-plane trigger, the Megapixel Trigger limits the track reconstruction to track segments that are found as *inner/outer* tracks, which involve *inner* and *outer* pixel hits (defined below). The inner/outer tracks are used to "seed" the track-finding algorithm. Seed tracks are extended to adjacent pixel planes to find all hits belonging to a trajectory. An important feature of the Megapixel Trigger is that seed tracks are found relatively close to the primary-interaction vertex. This feature can be exploited to extend parallelism in the hardware architecture throughout the Level 1 trigger.

One of the central concepts of the Megapixel Trigger is the distinction between *inner* and *outer* pixel hits, where a single hit corresponds to an (x,y) position measurement made by a clustering algorithm, which combines data from adjacent pixels. Inner hits are defined as hits located in a region close to the Tevatron beams. For hits in a pixel plane, this region is defined as a square with a side of length D centered on the beams and having the same orientation as the square beam hole in the pixel plane. In practice, the magnitude of D is determined by the size of individual pixel sensors, so that the innermost sensors are all within the square region that surrounds the beam hole. The pixel hits from these innermost sensors are the inner hits, and about one third of all hits are inner hits. All of the remaining hits, outside the square region, are outer hits.

The Megapixel Trigger combines hits from three adjacent pixel stations to reconstruct track segments. The track finding is constrained to combinations of three hits, in which the first hit is an inner hit, and the second and third hits are outer hits. By imposing this constraint, the Megapixel Trigger finds only those trajectories that pass through at least three pixel stations, and pass from the inner to the outer pixel region. The 3-station track segments that are found in this manner are referred to as *inner/outer* tracks. The advantages of this approach are that (1) only a limited number of hit combinations need to be considered for pattern recognition, and (2) each trajectory is found *once* by the trigger hardware. By finding tracks only once, the Megapixel Trigger differs significantly from the BTeV baseline trigger, where track segments belonging to a single trajectory are found for *each* pixel station that recorded hits for the trajectory.

The percentage of tracks found as inner/outer tracks is quite high, and is shown in Table 8.4. The table shows the percentage of inner/outer tracks relative to all tracks that pass through at least three pixel stations and have a momentum greater than 1 GeV/c. Results are shown for different types of tracks in minimum bias, B, and charm events for several values of d, where d (which is equal to $D/2$) is the distance from beam center to the border that defines the inner and outer pixel regions. Table 8.4 shows that for most types of tracks the percentage of inner/outer tracks does not vary with d, for the range of d values we have considered. For tracks coming from B decays there is a slight increase in inner/outer tracks for larger values of d, due to the larger p_t for these tracks. From these results we conclude that a large percentage (about 90%) of tracks coming from primary interaction vertices can be found as inner/outer tracks, and used to reconstruct primary vertices in the Megapixel Trigger. Furthermore, we conclude that a large percentage of B-decay tracks (more than 90% for values of d greater than 1.1 cm) can be found, and used to trigger on B events.

After the inner/outer tracks are found, the Megapixel Trigger identifies the *innermost* pixel hits for each trajectory found as an inner/outer track. The goal is to reconstruct the track segments closest to the point of origin. These are the track segments that are least likely to suffer from multiple Coulomb scattering errors, and are required to reconstruct vertices and identify B-decay tracks. For example, Fig. 8.17 shows pixel hits for a simulated $B \to \psi K_s$ event in the $z-y$ projection. In Fig. 8.17a inner/outer tracks are shown as line

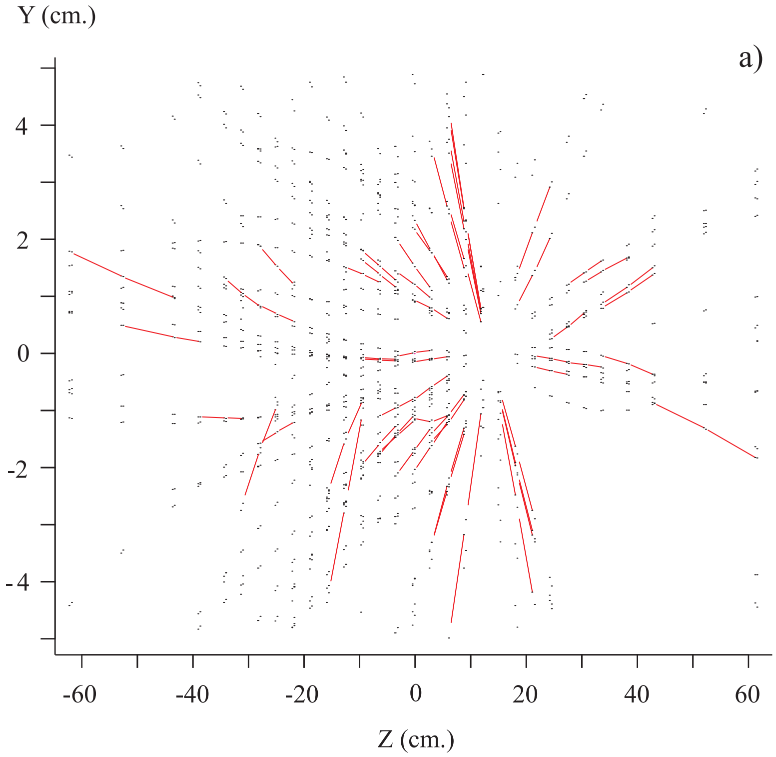
Distance of inner/outer border from beam center	Minimum Bias Tracks from primary vtx.	Generic B Tracks from primary vtx.	Generic B Tracks from secondary vtx.	Generic charm Tracks from secondary vtx.
d = 1.0 cm	91%	91%	87%	89%
d = 1.1 cm	91%	91%	91%	90%
d = 1.2 cm	91%	92%	93%	90%
d = 1.32 cm	90%	91%	94%	90%
d = 1.5 cm	89%	89%	95%	89%

Table 8.4: Percentage of tracks found as inner/outer tracks, where tracks are required to have momentum greater than 1 GeV/c and are required to pass through at least three pixel stations.

segments. There are two segments per track: one from the first hit to the second hit on the track and one from the second hit to the third hit. Fig. 8.17b shows the corresponding track segments for the innermost pixel hits associated with each track that was found as an inner/outer track. In many cases, the inner/outer tracks already include the innermost pixel hits, and no additional step is required to find these hits. For the remaining tracks, we are considering two algorithms to find the innermost hits. In the first algorithm, we extend tracks to adjacent pixel stations to locate the innermost hits by using an approach that is based on the linking of minivectors in the BTeV baseline trigger. In the second algorithm, we reconstruct inner/inner tracks in the first stage of the Megapixel Trigger, and then eliminate track candidates that do not include the innermost hits. For both algorithms, the track segments that are found are subjected to a fitting procedure to determine track parameters.

The hardware architecture of the Megapixel Trigger is organized as a pipeline with parallelism in z (the beam direction), and individual track and vertex processors (digital signal processors, for example) are assigned to a range of z positions in the pixel vertex detector. Processors assigned to a particular range in z receive the data for tracks that originate at a vertex within that range. This can be accomplished with a data-driven architecture that evaluates the apparent origin for each track based on the slope of the track. The purpose is to "funnel" all of the relevant data for an interaction to the same processor, where a trigger decision can be made.

The details of the parallel-pipeline architecture have not been worked out. However, the advantage of this architecture is that the majority of tracks associated with an interaction are funneled to a processor based on the z position of the interaction vertex. This has two consequences. The first is that tracks associated with a primary vertex (call it vertex A), and tracks from B-decay vertices are brought together in a single processor for vertex reconstruction, impact-parameter calculations, and a Level 1 vertex-trigger decision. The second consequence of this architecture is that tracks that occur during the same beam crossing, but are *unrelated* to vertex A, tend to be funneled to other processors. Unrelated tracks



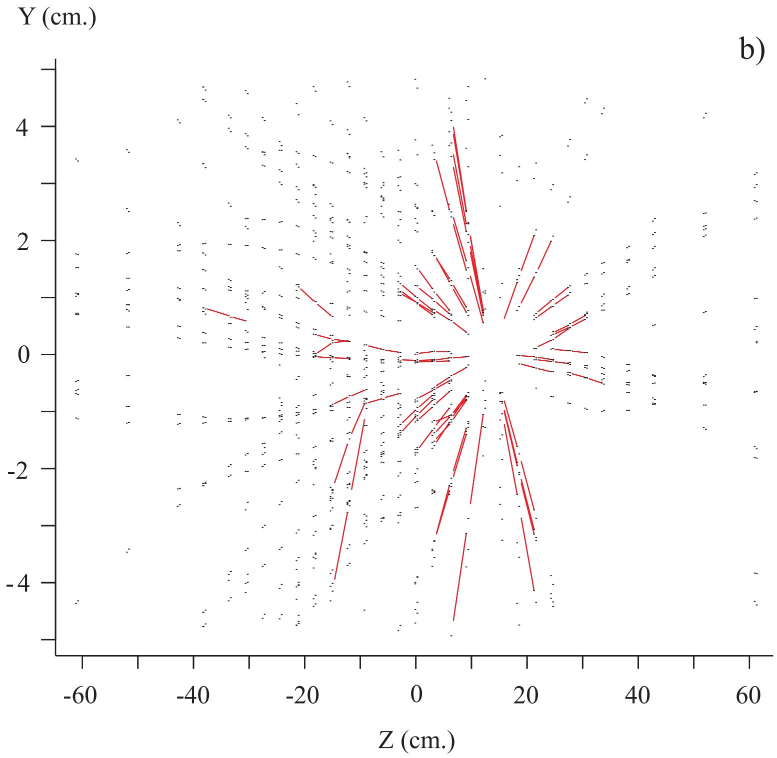


Figure 8.17: Pixel hits (shown as dots) for a $B \to \psi K_s$ event in the $z-y$ projection. The figure shows a) inner/outer tracks (two line segments per track), and b) track segments made up of the three innermost pixel hits for each inner/outer track found in the event.

include those tracks associated with other vertices, and tracks produced by reinteractions in the material of the vertex detector. These tracks are processed independently of tracks associated with vertex A, and do not contribute to the computational load for that vertex.

To conclude, the primary goal of the Megapixel Trigger is to generate Level 1 trigger decisions with high efficiency for B and charm events. A large percentage (more than 90%) of tracks are found *once* as inner/outer tracks. The inner/outer tracks "seed" the track reconstruction, and are used to find the track segments that are made up of the innermost pixel hits. These track segments are close to their point of origin. They are funneled to track and vertex processors by a data-driven architecture. This means that the processing load for a particular vertex is reduced by excluding unrelated tracks from that vertex, and the load for multiple interaction vertices in a single beam crossing is distributed over multiple processors.

8.8 The Global Level 1 Trigger Framework

At the output of the first level trigger there will be a 'global Level 1 trigger' subsystem that will allow us to combine the various individual trigger elements into final triggers (for example, an event with only weak evidence for a secondary vertex but with an indication of a high p_T single lepton might be accepted). It will also have the ability to apply a programmable prescale to the various triggers before they are 'or'ed together to form the final trigger. We will also include the ability to dynamically change the prescales during the course of the store. This device plays a central role in the trigger and data acquisition architecture and is discussed in great detail in chapter 9.

8.9 The Level 2 and 3 Triggers

We have shown that the Level 1 trigger can reduce the trigger rate by a factor of about one hundred. The event rate is then reduced to $\approx$100 kHz in the worst case during the highest luminosity running. While it is technically feasible to move all this data, amounting to a maximum rate of 10 GB/s, to the Level 2 trigger, we discuss below an architecture that allows the Level 2 decision to be made on a subset of the event data, thereby reducing the total bandwidth needed. Only for events passing Level 2 is the complete event assembled. We define further processing done on the complete event to be Level 3. If necessary, the Level 3 software can be quite comparable to the offline analysis in previous-generation experiments.

We anticipate that the data from Level 1 will be dominated by light quark events: the $b-\bar{b}$ production rate is about 20 kHz and not all of these are within the detector or trigger acceptance. The charm cross section is higher, but the triggering efficiency will be poorer by design. Hence, the primary task of the Level 2 trigger will be to reject the remaining light quark events from real collisions at $\sqrt(s) = 2.$ TeV. In addition, Level 2 will provide enough kinematic information about the event so that a meaningful decision on the event decay properties can be quickly taken by Level 3.

Level 1 treats each sub-detector completely independently from the others: muon and pixel detectors (Level 1 vertex) for a given crossing are not treated sequentially in the same processor. They are handled asynchronously in different processing units. Events passing the global Level 1 trigger do not come in sequential order. Level 2 decisions are based on partially reconstructed events. By this, we mean that not all information from all sub-detectors is available in the memory of a given level 2 processor. Possibly, the information on a given sub-detector element (i.e., pixel detector or the forward silicon tracker (FSI)) is partial: only a few planes could be used.

While the Level 1 vertex trigger broadly classifies the events in terms of their transverse detachment, looking for a few detached track with moderately high P_t, the task of Level 2 will be to confirm that the event surviving Level 1 cuts do indeed contain a heavy quark decay. Level 3 will classify these heavy Quark candidates in terms of final states, and make a decision according to an exhaustive trigger table.

In this section, we propose a few algorithms for the Level 2 trigger based on the pixel detector, the two most upstream FSI chambers, and the two most upstream straw chambers (STRAW). The basic idea is to firm up the measurement of the transverse momentum of the tracks, and improve the vertex position resolution using this momentum information. The main software tool for this task is the MCFast Kalman filter (or fitter) to (i) test the hypothesis that a track is real and self-consistent and (ii) improve the track parameters. No attempts at doing particle i.d., search for neutral kaons, π^0, etc. are currently proposed at Level 2. Cuts based on the invariant mass of detached tracks could also be successful, but are obviously final-state specific.

8.9.1 Preliminary studies of the Level 2 algorithm

We have simulated a possible Level 2 algorithm. Note that the vertex pattern recognition, currently done as the last stage, could be done at any time after the first stage, as the Level 2 tracks are genuine, complete 3D tracks with a covariance matrix. Level 1 tracks with at least 3 minivectors are used as seeds for the Level 2 pattern recognition. The pion particle identification is always assumed. The Level 2 algorithm works in several sequential phases:

1. Adding pixel hits. The Level 1 pixel pattern recognition has known deficiencies. For instance, a track may cross a ϕ slice boundary and be partially reconstructed. Level 1 algorithms tend to limit the number of hits found on a track to a minimum, in order to gain time in assembling the track. Despite the very low cell occupancy in the pixel detector, there are still a few confused or fake tracks. Adding new hits and performing a Kalman filter allows us to:

   - Reject spurious tracks. About 10 % of the Level 1 detached tracks do not pass the Kalman filter confidence level requirement.
   - Improve the parameters of the track.

- Compute the best possible estimation of the error on these parameters, such that displacement errors at the vertex are correctly estimated. Errors due to multiple scattering are properly taken into account.

2. Adding hits from the straw chamber or the forward silicon tracker. The momentum resolution $\Delta P/P$ using the pixel information only for forward tracks is about 6 %. These tracks are extrapolated to the location of the most upstream tracking station (about ± 1.0 m from the interaction point). Confirming hits are searched for within a window typically 1.0 cm wide. The left-right ambiguities for straw hits are resolved. The second tracking station, located ± 1.5 m from the interaction point, is also used. The momentum resolution, shown in Fig. 8.18, is around 2% for tracks reaching the first or second station. Note that these two detectors do not cover the same rapidity range, tracks reaching the second station tend to have higher momentum and therefore have smaller sagitta and bigger $\Delta P/P$. They are however measured over a longer lever arm. These two effects partially compensate. The transverse momentum resolution also improves, from $\approx 20\%$ at Level 1 down to a few percent.

3. All Level 2 tracks with a good Kalman fit confidence level are considered for the vertex pattern recognition phase. They are required to be "forward", i.e., have straw or silicon strip hits. However, if they point to these detectors, and no hits are found for a few consecutive planes, they will be rejected. The level 2 primary vertex resolution is shown in Fig 8.19.

The detachment criterion at level 2 is based on an exact calculation of the distance of closest approach of the track to the primary vertex and the error on this quantity. The detached track multiplicity, requiring a 5σ detachment, is shown in Fig 8.20, for events passing the Level 1 trigger (based on 2 detached tracks). Note the level 2 detached tracks are not necessarily the same tracks that were called detached at Level 1. A combined Level 1 and Level 2 rejection factor on light quark events of about 1 in 1,000 is achievable, with an acceptance for $B \to \pi^+\pi^-$ of about 40 %. Finally, at this stage, the $B \to \pi^+\pi^-$ mass resolution is about 98 MeV, assuming pion particle identification. The mass resolution is, of course, significantly improved when information is used from all the forward tracking stations.

These Level 1 and Level 2 studies are based on a full pattern recognition code. The Monte-Carlo generator information is used solely to measure the performance of the code, not to select hits on tracks. Future efforts will be concentrated on improving the CPU performance and tuning the algorithm to minimize the computing needed in the Level 2 processors.

8.9.2 Level 3

The final trigger level will combine information from different sub-detectors to further suppress background events. Particle identification will be available at this stage and could be

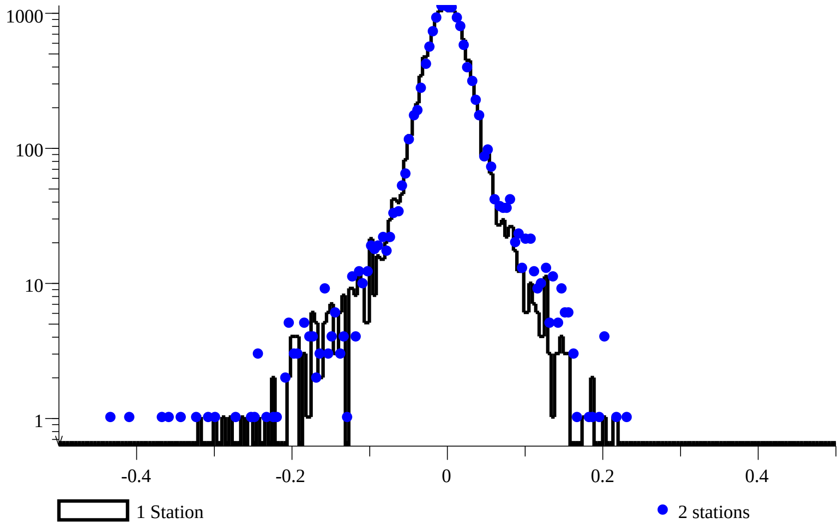


Figure 8.18: The momentum resolution for the light quark sample, at level 2, using the additional pixel hits and the forward silicon strip tracking system. The plot labeled "1 station" refers to track with silicon hits from the most upstream station, the plot "2 stations" uses the two most upstream stations, with at least one hit in either station. The σ of the core distribution is about 2.15 % for "1 station" plot and 2.05 % for the "2 stations" plot.

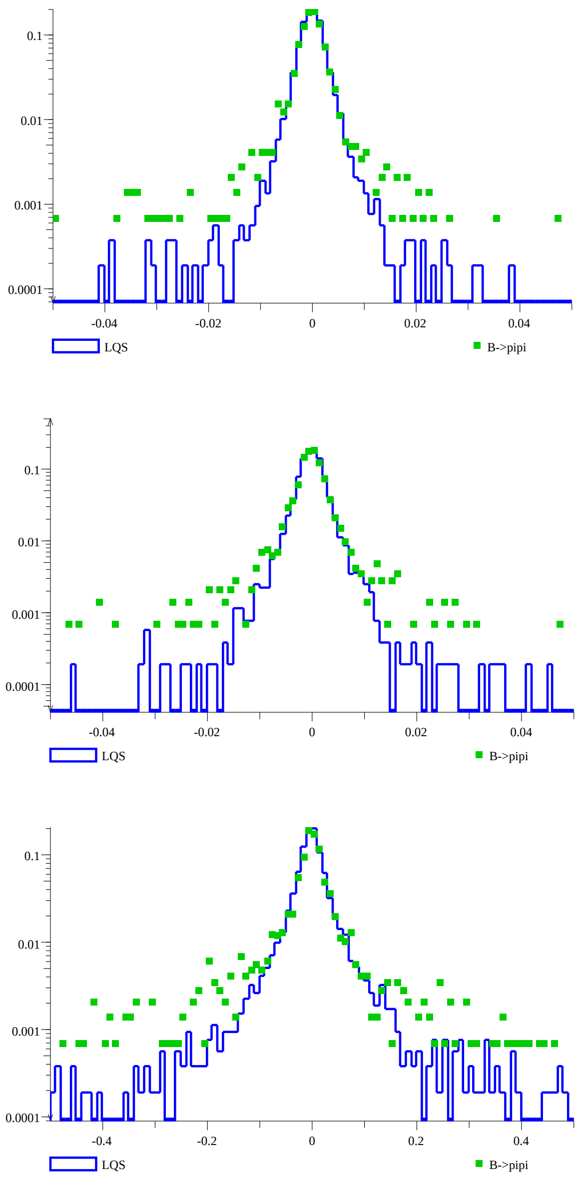


Figure 8.19: The primary vertex resolution along the X (top), Y (middle) and Z (bottom) direction at Level 2, for the light quark sample and for the $B \to \pi\pi$ sample.

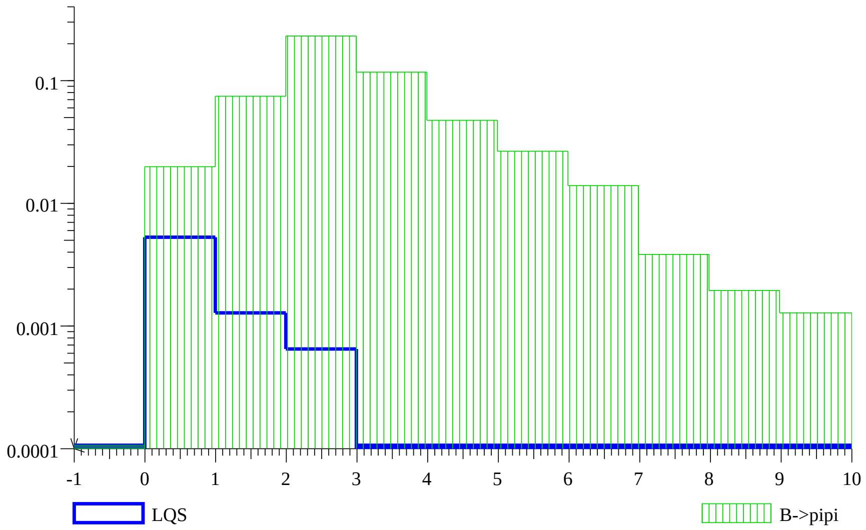


Figure 8.20: The Level 2 detached $(5.\sigma)$ track multiplicity for the light quark event sample and for the $B \rightarrow \pi\pi$ sample selected by the offline reconstruction program. In addition, these Level 2 tracks must have a transverse momentum greater than 0.5 GeV/c. The Level 1 trigger criteria has been applied. The vertical axis is the probability of accepting a beam crossing given the number of detached tracks on the horizontal axis.

used to obtain a very clean direct charm signal for specific final states. In addition, the Level 3 trigger is likely to go through a long list of heavy quark decays to final states of interest. We intend to fully reconstruct the events at this level before the data are sent to a storage system and made available for off-line processing.

Because the event rate surviving this last level may still be a few kHz, the software will probably have to reduce the amount of data per event to archival storage by writing out an event summary which eliminates much of the raw data. The event summary would be around 20 KB so that the output rate could still be as high as 80 MB/s. This results in a dataset size that could be as high as 800 TB/yr. However, initial phases of the experiment will run at much lower rates and will be comparable in dataset size to a current fixed target experiment.

Bibliography

[1] BTeV EOI (May 1997).

[2] These results are based on the work of R. Isik, W. Selove, and K. Sterner,“Monte Carlo Results for a Secondary-vertex Trigger with On-line Tracking,” Univ. of Penn. preprint UPR-234E (1996).

[3] P. Lebrun, “On the Vertex Pattern Recognition in the BTeV Trigger,” BTeV internal note 97/17 (1997). See also BTeV internal note 99/6 and 99/7. (1999)

[4] D. Husby, P. Chew, W. Selove, and K. Sterner, Nucl. Instr. and Meth. A **383** (1996) 193.

[5] For updated vertex trigger documentation, see http://www-ese.fnal.gov/eseproj/trigger/default.htm and E. Gottschalk, BTeV-pub-99/1, presented at DPF99