Fermilab FERMILAB-CONF-03-496-E DECEMBER 2003

The BTeV Trigger – Recent Developments

Penelope Kasper *
Fermi National Accelerator Laboratory, PO Box 500, MS 122, Batavia, IL 60510, USA

Abstract

BTeV is a collider experiment at the Fermilab Tevatron dedicated to precision measurements of CP violation, mixing and rare decays of beauty and charm hadrons. The detector is a forward spectrometer with a pixel vertex detector inside a dipole magnet. A unique feature of BTeV is the trigger, which reconstructs tracks and vertices in every beam crossing. We present here an overview of the BTeV trigger and a description of recent improvements in trigger timing.

1. Introduction

The BTeV experiment includes a sophisticated trigger system that rejects more than 99.9% of light-quark background events, while retaining large numbers of B decays for physics analyses. The BTeV trigger reconstructs tracks and vertices in every beam crossing and looks for topological evidence of a B decay downstream of a primary vertex. This is achieved by using a pixel vertex detector that has low occupancy, excellent spatial resolution and fast readout; a heavily pipelined and parallel trigger architecture; sufficient memory to buffer the events while waiting for a trigger decision; and a fault tolerant and fault adaptive error handling system. A more detailed description of the BTeV trigger can be found in the BTeV [1] Technical Design Report.

The trigger system consists of three levels. Each level contributes to the reconstruction of events, and successive levels impose more and more re-

* for the BTeV Collaboration
Email address: penny@fnal.gov (Penelope Kasper).

fined selection criteria to select B events and reject light-quark background events. The trigger is designed to run at an initial (peak) luminosity of $2\times10^{32}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ with a 132 ns, 264 ns, or 396 ns bunch crossing interval, corresponding to an average of 2, 4, or 6 interactions per crossing respectively. Data from all detector subsystems are temporarily stored in Level-1 buffers as trigger decisions are made. The full data rate from the detectors is ~800 GBytes/sec. The Level-1 buffers consist of commodity SDRAM with approximately 1 TByte of memory.

The Level 1 trigger rejects at least 98% of background events while retaining about 60% of B events that would pass final analysis cuts. The Level 2 trigger refines the track and vertex reconstruction and rejects 90% of the remaining light-quark events. At Level 3 we perform a complete analysis of the data. This is comparable to the offline analysis performed by other high-energy physics experiments. Level 3 rejects 50% of the remaining events and reduces the output event size. The output data rate from Level 3 is 200 MBytes/sec.

2. Level 1 Pixel Trigger

2.1. *Algorithm*

The pixel vertex detector consists of an array of 30 stations of silicon pixel planes perpendicular to the beamline and distributed over approximately 125 cm along the interaction region. Each station contains two planes: one with the pixels oriented so the narrow pixel dimension is horizontal and one with the narrow pixel dimension in the vertical direction. The Level 1 pixel trigger employs a two stage algorithm[2] consisting of a segment-finding stage followed by a track and vertex finding stage.

Data from the pixel detector front ends are sent to FPGA pixel processors that group individual pixel hits into clusters and translate the row and column information into x-y coordinates. Hit clusters from three neighboring pixel stations are then routed to FPGA-based hardware for the segment finding stage of the Level 1 trigger algorithm. This stage finds the beginning and ending segments of tracks in two separate regions of the pixel planes, an inner region close to the beam axis and an outer region close to the edge of the pixel planes.

The segments found in this stage are then sorted by a custom switch according to beam crossing number and routed to a programmable embedded processor in the track and vertex farm. The baseline design uses Digital Signal Processors (DSPs) for the embedded processors. In the segment matching phase inner segments are projected outwards and outer segments are matched to inner segments to form complete tracks based on their proximity to these projected trajectories.

After the segment matching phase, the algorithm then proceeds to the vertex finding phase. It begins this phase by processing the tracks found from the previous phase, calculating their momentum from the track curvature and knowledge of the magnetic field in the spectrometer magnet, and calculating their transverse distance from the beam axis. It then loops over all tracks with $p_T < 1.2$ GeV/c that appear to originate close to the beamline to search for primary interaction vertices.

Once all the primary vertices are found, the algorithm searches for detached tracks by looping over all tracks not attached to any primary vertex and calculating the impact parameter with respect to each primary vertex. A detached track is then assigned to the primary vertex for which its impact parameter is smallest. A Level 1 vertex trigger accept is generated if there are at least 2 detached tracks with $p_T > 0.5$ GeV/c, associated with the same primary vertex, and directed towards the instrumented arm of the spectrometer.

2.2. *DSP Timing Studies*

A critical issue in the Level 1 trigger is the timing of the Level 1 algorithm[3] that runs on the DSPs. The execution speed directly determines the total number of DSPs required in the Level 1 farm and therefore its cost and complexity.

The starting point of our studies was a variant of the original C language version of the code. All DSP timing studies were done on a Texas Instru-

ments (TI) C6711 DSP Starter Kit (DSK) board with a 150MHz TI TMS320C6711 DSP. Much of the Level 1 optimization work has focused on the segment matching phase because it is the most time consuming part of the DSP algorithm. Although the current version is eight times faster than the original, it still takes more than 50% of the DSP processing time.

In the segment matching algorithm, internal segments are checked against an entire list of external segments to find possible matches that form a complete track. This is an $O(n^2)$ process that consumes a significant portion of CPU time. However, one could make use of the fact that external and internal triplets belonging to the same track have approximately equal slopes in the non-bend view. By sorting triplets into several bins based on their slopes, each internal triplet need only be checked against external triplets in one or two bins instead of an entire list, thereby reducing processing times significantly. A simple and novel hardware implementation known as a *hash sorter* has been proposed[4]. The sorting can be done in an FPGA before the data is sent to the embedded processor.

A test design has been implemented on the preprototype Level 1 track and vertex hardware. This hardware, which forms the basic unit of the track and vertex farm, consists of four DSPs, three FPGAs and two microcontrollers. Segment data is received by this hardware after going through an event-building switch. One of the FPGAs, acting as a Buffer Manager sends all segment data for one crossing to one of the four DSPs. The hash sorter function has been compiled in our current Buffer Manager FPGA device (Xilinx Virtex II XC2v1000). The logic cell usage is $\sim$7% and the memory block usage is $\sim$10%.

To test the hash sorter, we applied it to the track and vertex portion of the Level 1 trigger algorithm. Our timing studies were done on 11 simulated segment data files with a fixed number of interactions per crossing ranging from 1-11. Execution time of the trigger algorithm in CPU cycles on a 1.13 GHz Pentium III-M was measured and averaged over the $\sim 2,500$ crossings in each data file.

These measurements were done with and without the hash sorter and the results are shown in Fig. 1.

Since the hash sorter affects only the performance of the segment matching routine of the algorithm, we show the results for this portion alone on the left-hand plot. Results for the entire algorithm are shown in the right-hand plot which also includes times for the entire algorithm excluding the segment matching routine. These results indicate an improvement of $\sim 5\times$ for the segment matching routine and an overall improvement of $\sim 2\times$ for the entire algorithm. They also show that, while total times are dominated by the segment matching routine when the hash sorter is not used, the time consumed by the other portions of the algorithm become more dominant when the hash sorter is used. Although these studies were done on a Pentium III, performance on other platforms should exhibit a similar trend of acceleration with hash sorting.

3. Level 2/3 Trigger

The Level 2/3 trigger is a farm of commodity processors running Linux. After a Level 1 accept, all data for an event will be transferred from the Level-1 buffers to a Level 2/3 processor. If the event passes the Level 2 trigger it is then processed by Level 3 in the same farm node. The distinction between Level 2 and Level 3 is that Level 2 uses only the pixel data (or possibly pixel plus forward tracking data), whereas Level 3 uses the full event data.

The basic requirements for the Level 2 trigger are (i) a rejection factor of 10 on light quark crossings, (ii) an acceptance higher than 90% on relevant heavy quark decays, (iii) and trigger timing that must be 10 msec or less per bunch crossing. We do not anticipate that memory utilization will be a critical issue.

The input data to Level 2 consists of all Level 1 tracks and vertices and the raw pixel hits. The Level 2 algorithm performs a Kalman filter track fit on the Level 1 tracks and refits the primary vertices. It then searches for other primary vertices and detached secondary vertices. It also looks for high p_T single detached tracks, corresponding to decay modes with only one charged prong (e.g.

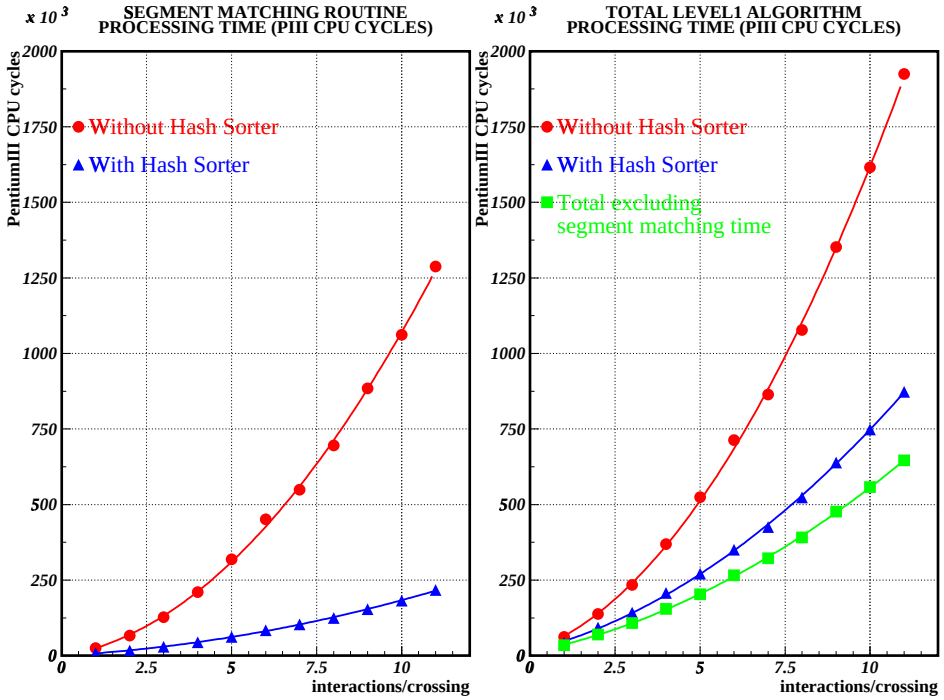

Fig. 1. Execution times of the DSP part of the Level 1 trigger algorithm with and without the hash sorter.

$B^+ \to \pi^+ \pi^0$).

A secondary vertex must satisfy the following criteria: (i) tracks must have a track fit confidence level greater than 2.5%, must be detached from the primary vertex by more than 3.5σ and must have transverse momentum greater than 0.5 GeV/c; (ii) all tracks must have the same sign p_z and be pointing away from the primary vertex; (iii)the vertex must have a confidence level greater than 2% and must be detached from the primary vertex by more than 3.5σ; (iv) the vertex must have an invariant mass less than 7 GeV and more than 100 MeV outside the K_s mass. An event passes Level 2 if it has either a detached secondary vertex or a high p_T detached track.

The original Level 2 philosophy was to find all pixel hits on all possible tracks. It started with the Level 1 tracks and vertices but then searched for all pixel hits on the Level 1 tracks between the inner and outer segments, and then searched for other tracks from the unused pixel hits. The current philosophy is to use only the Level 1 tracks identified as being associated with the triggering Level 1 vertex (ie. the vertex with detached tracks). The Kalman fit is done using only the pixel hits in the inner and outer triplets of each Level 1 track. The new algorithm runs 700 times faster with no loss in signal/background. The Level 2 efficiencies for several decay modes of interest are shown in Table 1. The trigger efficiency for B decays is defined as the fraction of the events passing full analysis cuts that pass the trigger. The Level 2 efficiency is defined as the ratio of the number of events passing Level 2 to the number of events passing Level 1. The current code runs about 5 times faster than our requirements so we have plenty of leeway to improve the efficiency by refining the tracking and vertexing algorithms or adding extra algorithms such as a Level 2 muon trigger.

The goal of Level 3 is to achieve another factor of 2 in background rejection and to reduce the size of the event by a factor of 4. The Level 3 algorithm will be similar to the full offline reconstruction. We have prototype code for forward track-

Table 1
Level 2 trigger efficiencies

Event type	Level 2 efficiency
Light quark	7%
$B_s \to D_s K$	85%
$B^0 \to \pi^+\pi^-$	87%
$B^0 \to J/\psi K_s$	78%
$B^- \to \pi^- K_s$	72%

ing, K_s reconstruction, particle ID in the RICH, and electron, photon and π^0 reconstruction in the calorimeter.

In order to be as efficient as possible for known decay modes of interest yet keep the trigger as open as possible to new ideas, we adopt a strategy of first searching through a list of decay modes of interest and saving with highest priority those events which are consistent with one of these decay modes. We then select events with evidence of a heavy quark vertex.

4. RTES

The BTeV trigger system encompasses approximately 5000 CPUs distributed over a three-level trigger architecture. Time critical event-filtering algorithms that run on the trigger farms are likely to suffer from a large number of failures within the software and hardware systems.

To address this concern, we have established the BTeV Real Time Embedded System Collaboration (RTES) [5]. Funded by a $5M grant through the NSF ITR program, RTES is a group of physicists, engineers, and computer scientists working to address the problem of reliability in large-scale clusters with real-time constraints, such as the BTeV trigger. RTES is defining a software infrastructure to detect, diagnose, and recover from errors not only at the system administration level, but also at the application level. This infrastructure must be highly scalable to minimize bottlenecks or single points of failure. It has to be verifiable to make sure that it performs its functions in a timely fashion, extensible by users to acquire new detection/analysis methods, and dynamically changeable so that it can be reconfigured as the system operates. The problem is being approached by using a hierarchy of monitoring and control elements. The architecture is such that lower levels have high data rates, short reaction times, and a narrow view of system components, while higher levels have aggregated data summaries, longer reaction times, and a more global perspective of the system.

The goal of the RTES project is to create fault handling that can be used by all components in the BTeV trigger and DAQ. This subsystem must be capable of accurately identifying problems and compensating for them. This includes application related activities such as changing algorithm thresholds, and overall system activities such as load balancing. As many recovery procedures as possible will be automated. A simple example is the ability of the system to switch to a hot-spare Level 1 processing board when a failure is detected in a board that is actively processing data.

Each university involved in the RTES Collaboration has expertise in some aspect of the problem. In some cases, they already have toolkits that have been used to solve smaller scale problems related to real-time embedded systems and fault management. BTeV has established a prototype architecture for the trigger that is being used as a model for RTES software development. The prototype helps to establish subsystem boundaries, to give a sense of scale, and to identify required interfaces and error conditions. This prototype uses DSPs as the embedded Level 1 processors.

References

[1] http://www-btev.fnal.gov/public/hep/index.shtml

[2] E. E. Gottschalk, "Detached Vertex Trigger", Nucl. Inst. Meth. A473(2001)167

[3] M. Wang, "BTeV Level 1 Vertex Trigger", Nucl. Inst. Meth. A501(2001)214

[4] J. Wu *et al*, "Hash Sorter - Firmware Implementation and an Application for the Fermilab BTeV Level 1 Trigger System", FERMLAB-CONF-03-357-E

[5] http://www-btev.fnal.gov/public/hep/detector/rtes/