

Data flow analysis of a highly parallel processor for a Level 1 Pixel Trigger

G. Cancelo*, E. Gottschalk*, V. Pavlicek*, M. Wang*, J. Wu*

* Fermi National Accelerator Laboratory

Abstract

The present work describes the architecture and data flow analysis of a highly parallel processor for the Level 1 Pixel Trigger for the BTeV experiment at Fermilab. First the Level 1 Trigger system is described. Then the major components are analyzed by resorting to mathematical modeling. Also, behavioral simulations are used to confirm the models. Results from modeling and simulations are fed back into the system in order to improve the architecture, eliminate bottlenecks, allocate sufficient buffering between processes and obtain other important design parameters. An interesting feature of the current analysis is that the models can be extended to a large class of architectures and parallel systems.

I. BTeV Level 1 Trigger

Fermilab's BTeV experiment has been proposed with a Trigger System in three levels [1]. The Level 1 trigger will perform calculations using data from two detector systems: the pixel detector and the muon detector. The Level 1 must do crude data preprocessing plus Track and Vertex reconstruction while keeping the processing time as low as possible. Level 2 and 3, based on events that passed Level 1, will gather information from the already mentioned sub-detectors plus other sub-detectors to perform full pattern recognition.

The Level 1 Trigger process all events generated by collisions of protons and antiprotons in the Fermilab's Tevatron. The average event interarrival time is about 132 ns at a luminosity of $2 \times 10^{32} \text{ cm}^{-2} \text{ s}^{-1}$. Since the Level 1 processing time will be approximately three orders of magnitude longer, the Level 1 processor will pipeline and process many events in parallel.

Figure 1 shows the L1 Trigger building blocks. The Data Preprocessors have two main building blocks, the Pixel Preprocessor and the Segment Tracker. The Pixel Preprocessor formats and sorts the raw data coming from the Pixel Detector. There are two main modules in the Pixel Preprocessor, the x-y coordinate translator (XYPC) and the Time Stamp Sorter. The XYPC module formats the data; converting groups of Pixel Detector raw hits into x-y coordinate referenced data. The data generated every bunch crossing (BCO) of the accelerator is stamped with a distinctive temporal label called a Time Stamp (TS). The data sorting is done by the TS-ordering function based on the data Time Stamp. The next processing stage is the Segment Tracker, which generates triplets of points describing the beginning and the end of all tracks in each event. Each Pixel Preprocessor and Segment Tracker processes a small geographic portion of the pixel detector.

The Data Router or Switch routes all data that share a same Time Stamp to the same Track and Vertex processor. Each data event is assigned to a single processing node because trigger decisions are event by event basis. The Track and Vertex processors are grouped in larger units of hardware called Farmlets. Processors in a Farmlet share some resources such as data I/O, main buffering and network connections.

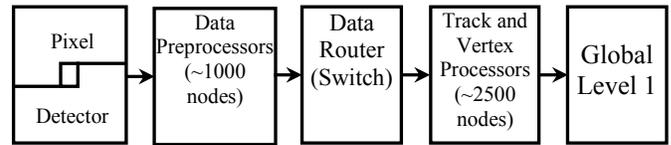


Figure 1. BTeV L1 Pixel Trigger

A key subject in the design of a multi-thousand node processing system is the data-flow. A data flow analysis guarantees that bottlenecks are eliminated and that sufficient processing and storage are allocated. The analysis also generates results that can be used to optimize the system architecture.

The data flow analysis of the BTeV L1 Trigger has been done using mathematical models and simulations. The mathematical models make extensive use of queuing theory. The L1 Trigger data inputs and outputs are described as stochastic processes. Subsystem behaviors are described by a set of differential-difference equations and solved either for their transitory or equilibrium states. Furthermore, the beauty of modeling resides in its generality. The current models are general enough to be applicable to a large class of parallel processing architectures.

The mathematical models have also been validated by behavioral simulation of the Level 1 Trigger Processor. The input to the simulators comes from the simulation of the BTeV detector [1]. The dataflow simulations represent timing and trigger functions as conceived today and as close as possible to their final implementation.

II. Pixel Processor and Segment Tracker (PP&ST) Dataflow Analysis

The dataflow analysis of the L1 Trigger cannot be covered entirely in this paper. Only the main sections of the PP&ST will be shown. A data flow analysis of the Track and Vertex processors is presented in paper N36-61 in this conference. For a comprehensive reading on both subjects please see [2][3]. The queuing model used for the PP&ST is shown in Figure 2.

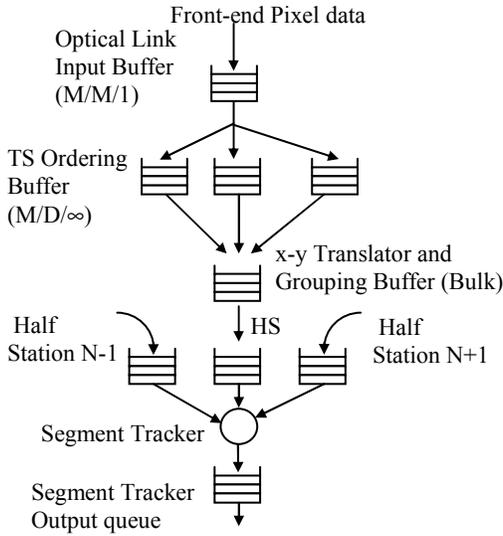


Figure 2. Pixel Preprocessor queueing model

One of the main results of the analysis of the Pixel Preprocessor and Segment Tracker has been the introduction of parallel “highways” in the L1 Trigger. Each “highway” N -folds the L1 Trigger in identical parallel branches, which process $1/N$ of the total data. The “highways” increase the event interarrival time proportional to the number of highways. This strategy allows the Pixel Preprocessors and Segment Trackers to process more hits per event and look at a bigger portion on the Pixel detector plane. The dataflow analysis also shows that $N = 8$ is a good compromise between the size of the pixel detector area that can be computed by a single Pixel Preprocessor and Segment Tracker module, the amount of electronics per board and the total number of nodes.

III. The TS-ordering process

The TS-ordering process opens a new queue when receives data with a TS different to all the ones in the existing queues. A queue dies when the data reception for that event is complete. Since the L1 Trigger has no way to tell the end of one event, we use a deterministic processing time. We have set this time equal to a complete revolution of the TS clock, that is 159 BCOs ($\sim 21\mu s$).

The analysis of the TS event ordering is fairly complex because the process must not only consider the queue birth-death distribution but, also, the size distribution of each individual queue. Each individual queues represents a non-stationary process. However, some simplifications can be made. We can define a new process that only considers the number of queues in the TS event ordering system, regardless of their size. This new process is a well-defined birth-death Markov chain. Each state represents the number

of existing queues in the system (Figure 3). The process can be modeled as a $M/D/\infty$ process. The birth time of the queues are generated by random queue arrivals. The interarrival times can be considered exponentially distributed. Queue deaths are caused by complete events leaving the system at deterministic interdeparture times of 159 BCOs.

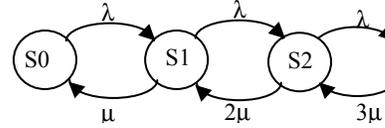


Figure 3. TS-ordering queues state transition diagram

Let λ represent the rate at which new queues are generated. From simulations the total Pixel Detector Half Station data rate is shown to be 0.9 events/BCO for 4int/BCO. This rate is reduced by the the number of highways $N=8$:

$\lambda=0.1125$ events/BCO into the Data Preprocessors.

μ , the service rate, is deterministic and equal to the time we want to wait before considering that the event is complete. In this example we set μ to $1/(159 \text{ BCOs})$ or $0.006289 \text{ BCO}^{-1}$. The $M/D/\infty$ process is always stable. The probability distribution function of this system is given by

$$p_k = \frac{(\lambda/\mu)^k}{k!} e^{-\lambda/\mu} \quad k = 0,1,2,\dots$$

The average number of queues in the system is given by:

$$E(N_q) = \frac{\lambda}{\mu} = \frac{0.1125}{0.006289} = 17.89 \text{ queues}$$

The simulations of 4410 BCOs show similar results (Figure 4). The number of TS queues open increases linearly at the beginning and stabilizes at around 18 queues. If we discard the transitory (the first 200 BCOs) the average number of queues from simulation is 18.13 respectively.

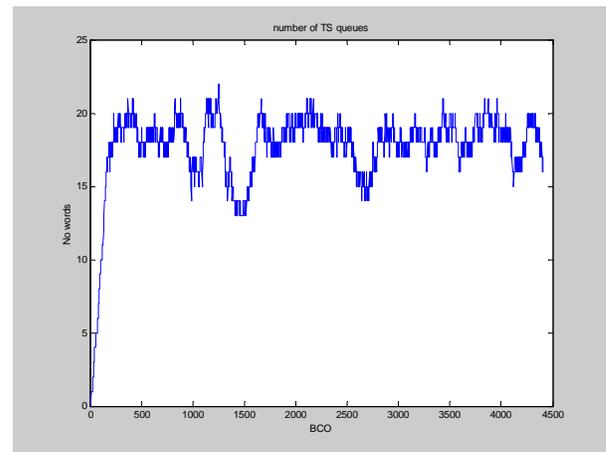


Figure 4. Simulation of the TS-ordering queues behavior.

The analysis of individual queue size can be performed as follow: We can calculate the conditional probability

distribution function of queue occupancy given that there are n queues and the total sum of data words in the queues is m . The selection of data in the queues can be modeled as a generalized binomial distribution:

$$P(M_1(t) = m_1, M_2(t) = m_2, \dots, M_n(t) = m_n | M(t) = m, N(q) = n) = \frac{m!}{m_1! m_2! \dots m_n!} p_1^{m_1} p_2^{m_2} \dots p_n^{m_n}$$

where: $\sum_{i=1}^n p_i = 1$ and $\sum_{i=1}^n m_i = m$

Since the input data-stream which generates the queues with individual TS is a Poisson process,

$$P(M(t) = m) = \frac{e^{-\lambda t} (\lambda t)^m}{m!}$$

Then, we can take away the conditionality on the total number of words m

$$P(M_1(t) = m_1, M_2(t) = m_2, \dots, M_n(t) = m_n | N(q) = n) = \frac{m!}{m_1! m_2! \dots m_n!} p_1^{m_1} p_2^{m_2} \dots p_n^{m_n} \cdot \frac{e^{-\lambda t} (\lambda t)^m}{m!}$$

the last equation can be written as

$$P(M_1(t) = m_1, M_2(t) = m_2, \dots, M_n(t) = m_n | N(q) = n) = \prod_{i=1}^n \frac{e^{-p_i \lambda t} (p_i \lambda t)^{m_i}}{m_i!} \quad (1)$$

since all the TS are equally probable

$$p_1 = p_2 = \dots = p_n = 1/n$$

$$P(M_1(t) = m_1, M_2(t) = m_2, \dots, M_n(t) = m_n | N(q) = n) = e^{-(\lambda/n)t} \prod_{i=1}^n \frac{(\lambda/n)^{m_i}}{m_i!} \quad (2)$$

Equation (2) is still conditioned by a fixed number of queues in the system. However, it let us study the distribution of data in the queues for a certain number of key values. For instance we can let n be the average number of queues or some upper bound.

What equation (2) shows is that for a given n the distribution of $M_1(t) \dots M_n(t)$ are independent Poisson processes with data rate λ/n . It is also known that as well as the interarrival times in a Poisson Process are exponentially distributed, the k -iterated interarrival of an event in (1) follows a k -stage Erlang distribution. In our case the distribution is conditioned for n fixed.

The average number of hits in the TS-ordering queues can be calculated using the average number of TS-queues and the average number of hits per event.

$$E(N_q) = \frac{\rho}{1-\rho} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}$$

$$\lambda = \text{hit_input_rate} = 0.241124$$

$$\mu = \frac{\text{Avg_NoTS_ordering_queues}}{\text{Deterministic_queuing_time} \times 14 \text{clk/BCO}} = 0.242587$$

$$\rho = 0.993966$$

$$E(N_q) = \frac{\rho}{1-\rho} = 165 \text{hits}$$

The simulations of about 4410 BCOs show an average number of words of 243.03, after the initial transitory dies out (Figure 5).

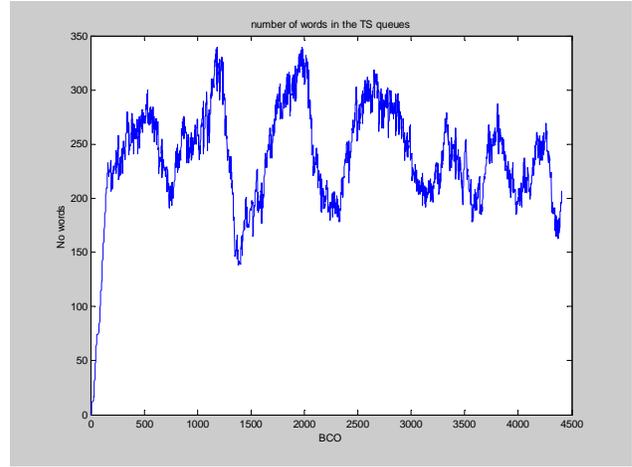


Figure 5. Simulation of number of hits in the TS-ordering queues.

IV. The x-y pixel cluster (XYPC) queue

The x-y pixel cluster (XYPC) queue can be modeled as a “bulk” M/M/1 process. In such a process the data arrives at the input queue in “bulks”. Every time the TS ordering process closes a queue, that entire queue is placed in the x-y translator buffer. The bulks are variable in size and equal to the size of the event that generates it. In other words, the x-y translator’s queue is composed by a number of queued customers, which are in turn of variable length. This problem is a generalization of the system with an r -stage Erlangian service, in this case using variable r . The bulk arrival state-transition diagram can be represented as in Figure 6.

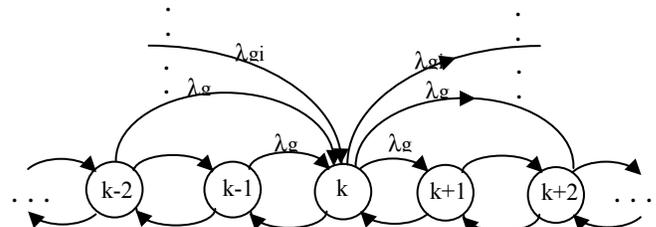


Figure 6. State transition diagram of the XYPC model.

Let $g_i = \text{Prob}[\text{bulk size is } i]$, then $\sum_{i=1}^{\infty} g_i = 1$

The equilibrium equations for the bulk arrival system can be written by:

$$(\lambda + \mu) p_k = \mu p_{k+1} + \lambda \sum_{i=1}^{k-1} p_i g_{k-i} \quad k > 1 \quad (1)$$

$$\lambda p_0 = \mu p_1$$

The numbers we are looking for are the mean size of the x-y translator queue and the average service time. The solution of the equilibrium equations involves z-transform methods. The bulk M/M/1 queue size in equilibrium suffers a “modulation” effect caused by the size of the events (bulks). The modulation is reflected in the discrete convolution shown by the summation in equation (1). Convolutions show in the z-transformed plane as the product of the z-transforms. The z-transform of the probability distribution of the x-y transform queue size $P(z)$ is:

$$P(z) = \frac{\mu(1-\rho)(1-z)}{\mu(1-z) - \lambda z[1-G(z)]} \quad (2)$$

where $G(z)$ is the z-transform of the probability distribution of the bulk size. The utilization factor ρ is defined, as usual, $\rho=1-p_0$. The value of ρ can, also, be obtained from (2) taking into account that $P(1)=1$. Then,

$$\rho = \frac{\lambda G'(1)}{\mu} \quad (3)$$

This result is not surprising because $G'(1)$ is the average bulk size, hence $\lambda G'(1)$ is the average arrival rate and $1/\mu$ is the average service rate.

Simulations of the L1 Trigger input data show that the probability distribution of the bulk size can be approximated by a Rayleigh distribution $f_X(x) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$ whose z-transform is given by

$$G(z) = \sigma \sqrt{\frac{\pi}{2}} \ln(z) z^{-\frac{\sigma^2}{2}} \quad (4)$$

Using (4) into (2), the expected number of queues in the bulk M/M/1 process is

$$E(N) = \frac{2\sqrt{\frac{\pi}{2}} \frac{\lambda\sigma}{\mu} - \sqrt{\frac{\pi}{2}} \frac{\lambda\sigma}{\mu} (1+\sigma^2)}{2\left(\mu + \lambda\sigma\sqrt{\frac{\pi}{2}}\right)} \quad (5)$$

Using, equation (3) and simplifying (5) can be written as

$$E(N) = \frac{\rho(\sigma^2 - 1)}{2(1 - \rho)} \quad (6)$$

The unknown parameter σ of equation (5) can be calculated using Maximum Likelihood Estimation (MLE) over the data sample. The estimated σ values for the PP&ST that process data coming from the central region of the Pixel Detector are in the range (31.15, 31.87). Applying these values to (6), the mean XYPC queue size is in the interval (4.02, 4.21) hits. Figure 7 shows a 4410 BCO simulation of the XYPC queue. The mean size of the XYPC input queue is $E(N)=4.32$.

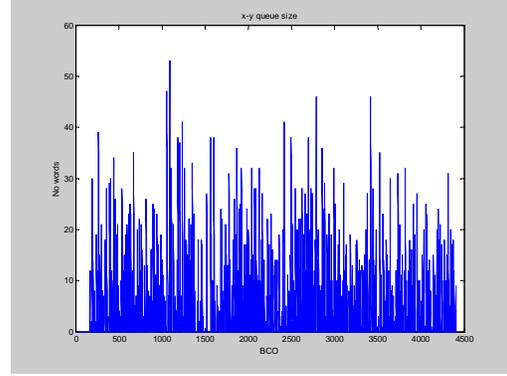


Figure 7. Simulation of the queue size in the XYPC module.

V. The Segment Tracker Architecture

The Segment Tracker finds 3-station long track segments called inner and outer triplets. A detailed description of the L1 Trigger Track and Vertex algorithm can be found in [4]. The Segment Tracker receives input from 6 Half Planes corresponding to both sides of three consecutive stations in the Pixel Detector. The Long Doublet module finds pairs of points using 2 neighbor stations. The Triplets module uses that result and adds the 3rd hit in the triplet. The Short Doublet modules validate the triplets using hits measured with lower precision. There is a queue associated to each input to store the incoming data. We have, also, defined other 7 internal queues for temporary data storage, which allows pipelining through the processing modules (Figure 8).

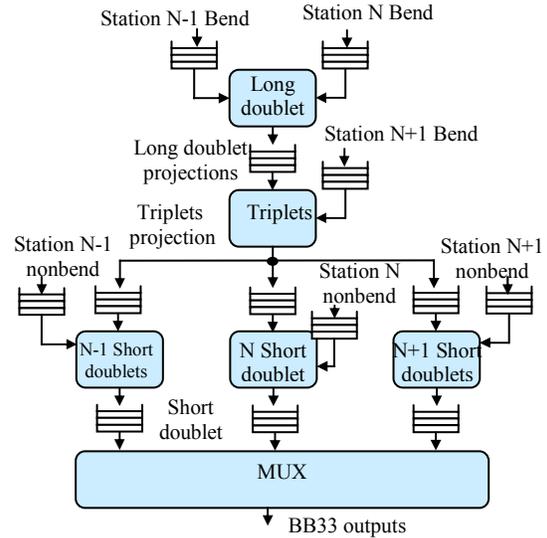


Figure 8. Segment Tracker algorithm queuing model.

As it was done for the TS-ordering process, if we only look at the stochastic process generated by the events and disregard the event sizes, the whole Segment Tracker can be model by a network of M/M/1 queues. This implies that the queues are independent and their input interarrival times are distributed exponentially with parameter λ , which can be

easily estimated from the data sample. The service time distribution for each module can, also, be estimated from the simulations. The simulations show that they are exponentially distributed as well. Hence, the mean event queue sizes are obtained by

$$E(R_q) = \frac{\rho}{1-\rho} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}$$

A more detailed analysis can model the Segment Tracker as a “bulk” service process. The data arrives hit by hit but is serviced event by event, where every event represents a “bulk”. The “bulks” are of variable size and modulate the queue sizes. The equilibrium equations for this process can be derived from the state transition model shown in Figure 9.

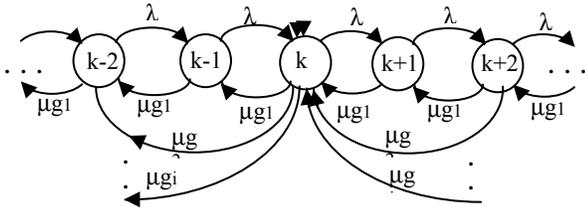


Figure 9. Segment Tracker state transition model

$$(\lambda + \mu) p_k = \lambda p_{k-1} + \mu \sum_{i=k+1}^{\infty} p_i g_{i-k} \quad k > 1 \quad (*)$$

$$\lambda p_0 = \mu \sum_{i=1}^{\infty} p_i g_i, \quad \text{where } g_i = \text{Prob}[\text{bulk size is } i], \text{ then}$$

$$\sum_{i=1}^{\infty} g_i = 1$$

Applying the z-transform, the mean number of hits in the queues responds to

$$E(N) = z_o \left(\frac{1}{1 - \frac{1}{z_o}} \right) = \frac{z_o^2}{z_o - 1} \quad \text{where } z_o \text{ can be calculated for}$$

every queue based on the queue's initial conditions.

Figure 10 shows a simulation run of 4 queues of the Segment Tracker model.

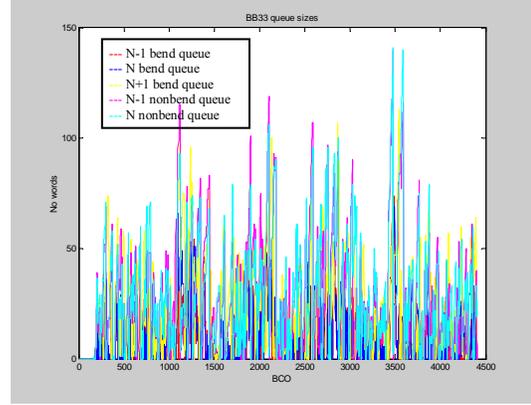


Figure 10. Queue size simulation in the Segment Tracker.

VI. Conclusions

The dataflow analysis of the L1 Pixel trigger has feedback valuable information into the system. An analysis of data bandwidth and latency has been very useful to balance the workload across the system such as bandwidth, latency, queue sizes and system service times. Those numbers are available at [2]. The latency in the L1 Trigger system is dominated by the TS-ordering function and by the Track and Vertex algorithm. The first one is constrained by the data generation process in the Pixel Detector, hence harder to modify, the last one can be reduced by speeding up the algorithms and taking advantage of FPGAs for part of their implementation. The dataflow analysis has also benefit the design of a fault tolerance trigger. Since stages cannot provide infinite data queuing or infinite processing bandwidth, they must deal with occasional buffering and processing overflows. The way we deal with this problem is by throttling the data stream, purging events to reduce queue sizes and processing load. A well-implemented throttle must handle data inefficiency gracefully. The overflows worsen in the event of a failure in the system. This problem is the main subject of the analysis reported in paper NS-xx of the current conference.

References

- [1] E. Gottschalk, “BTeV detached vertex trigger”, Nucl. Instrum. Meth. A 473 (2001) 167.
- [2] G. Cancelo, “Level 1 Pixel Trigger Data Flow Analysis”, BTeV-doc-1177-v1, Sept. 2002.
- [3] G. Cancelo, “Dataflow analysis in the L1 Pixel Trigger Processor Farm”, BTeV-doc-1178-v1, Sept. 2002.
- [4] M. Wang, “BTeV L1 Trigger Vertex Algorithm”, BTeV-doc-1179-v2, Sept 2002.