

Data Flow Analysis and Simulation of the Pixel Processor and Segment Tracker

Modeling and simulation procedure

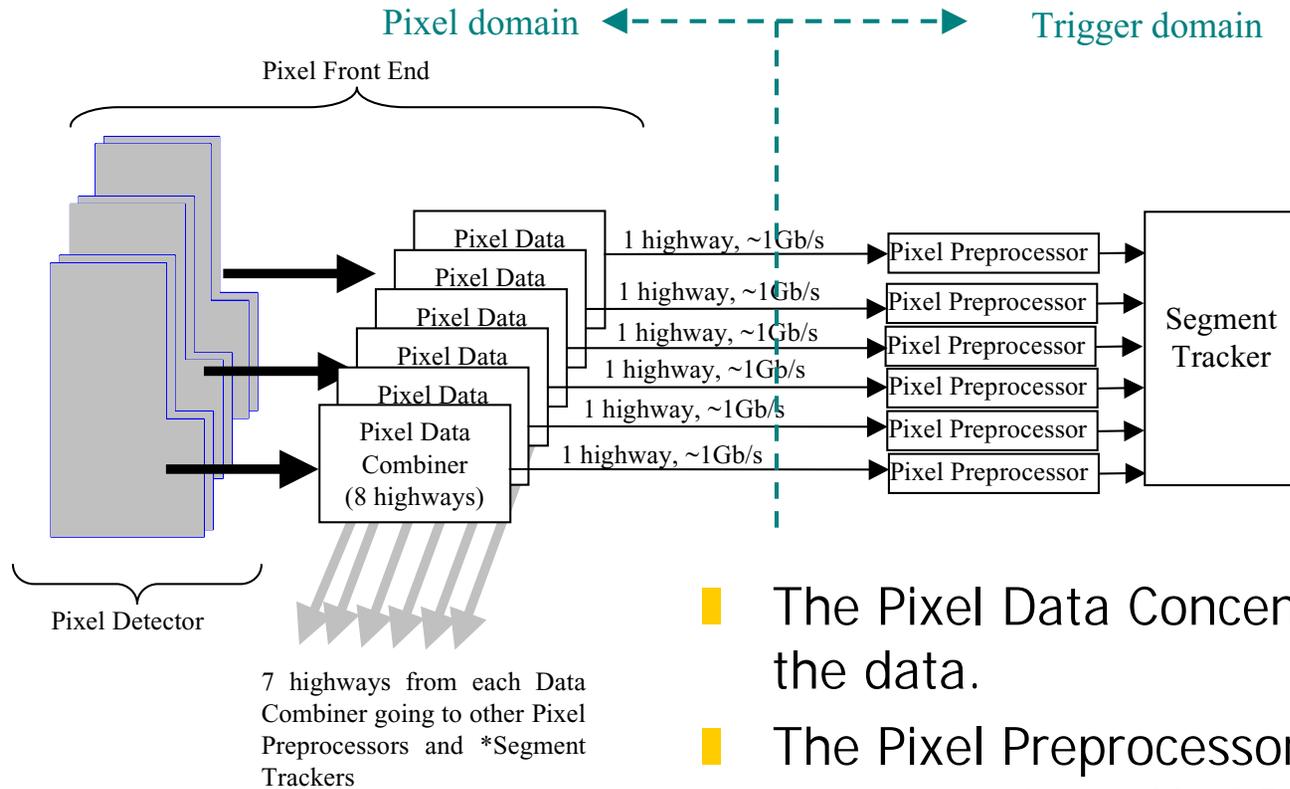
■ Modeling:

- Queuing Models are stochastic processes.
- Model inputs are probability distribution functions (pdf).
- Queues are modeled by one or more differential-difference equations which describe the queue's dynamic behavior.
 - The differential-difference equations describe a continuous time, discrete state process.
 - The state of the process concentrate all we need to know from it to predict its future dynamics. All queuing models in the current analysis are Markov.
- The dynamic system is solved for the equilibrium point and parameters of interest obtained from there.

■ Simulation:

- System modules are really implemented in code.
- Function processing and timing are as close as possible to the final implementation.
- Simulation inputs come from detector simulations.
- Parameters of interest are observed and compared to the ones obtained from models.

Pixel Front End Highways

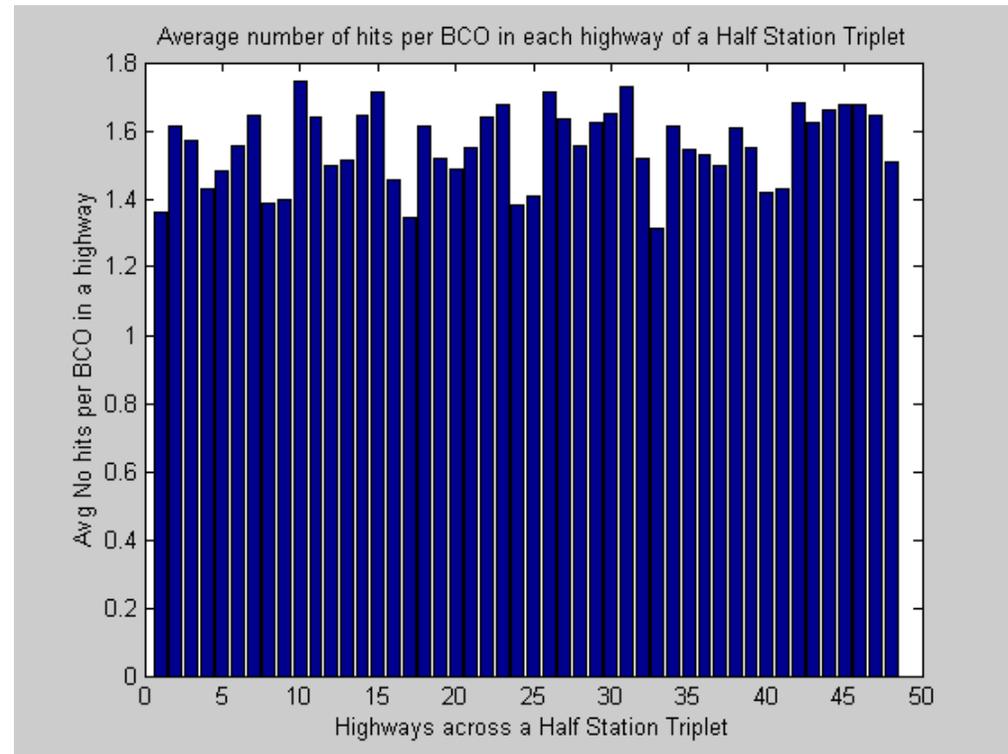


- The Pixel Data Concentrators "highway" the data.
- The Pixel Preprocessors process one highway of one Half Pixel Plane.
- The Segment Tracker process the output of six Pixel Processors (i.e. a Half Station Triplet).

Pixel Front End Highways (2)

■ Benefits of highways

- Make the highway average distribution fairly uniform
 - All Pixel Preprocessors process an equal amount of data on average
- The Pixel Preprocessors and Segment Trackers can process bigger portions of the Pixel Detector
 - The average inter-arrival time grows by a factor equal to the number of highways



- We have one more degree of freedom in the equation of bandwidth and computational power required in the Pixel Preprocessors and Segment Trackers

The Simulation Input data file (1)

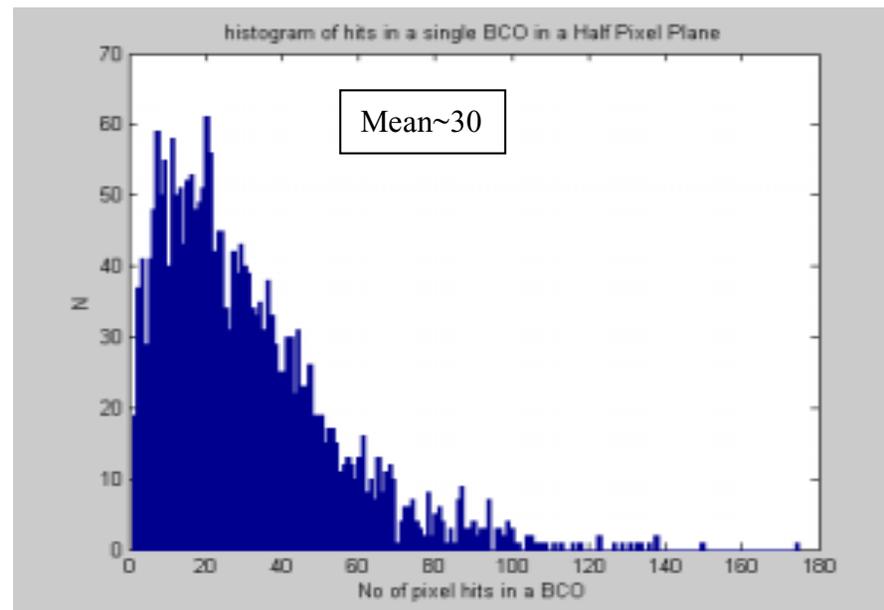
- The Pixel data was generated with BTeVgeant using the following parameters:
 - Pixel size: 50 x 400 microns,
 - Chip size: 22 columns, 128 rows
 - Magnetic field: 1.6T
 - Threshold: 2000 e-
 - Total No of Bunch Crossings (BCO): 745
 - Luminosity: 2^* (4 interactions per BCO on average)
 - No of stations simulated: 3, corresponding to the central Triplet of the Pixel Detector (i.e. stations 15, 16, and 17). Each Station is double-sided with one bend-view plane and one non-bend-view plane.

The Simulation Input data file (2)

■ *Some file statistics*

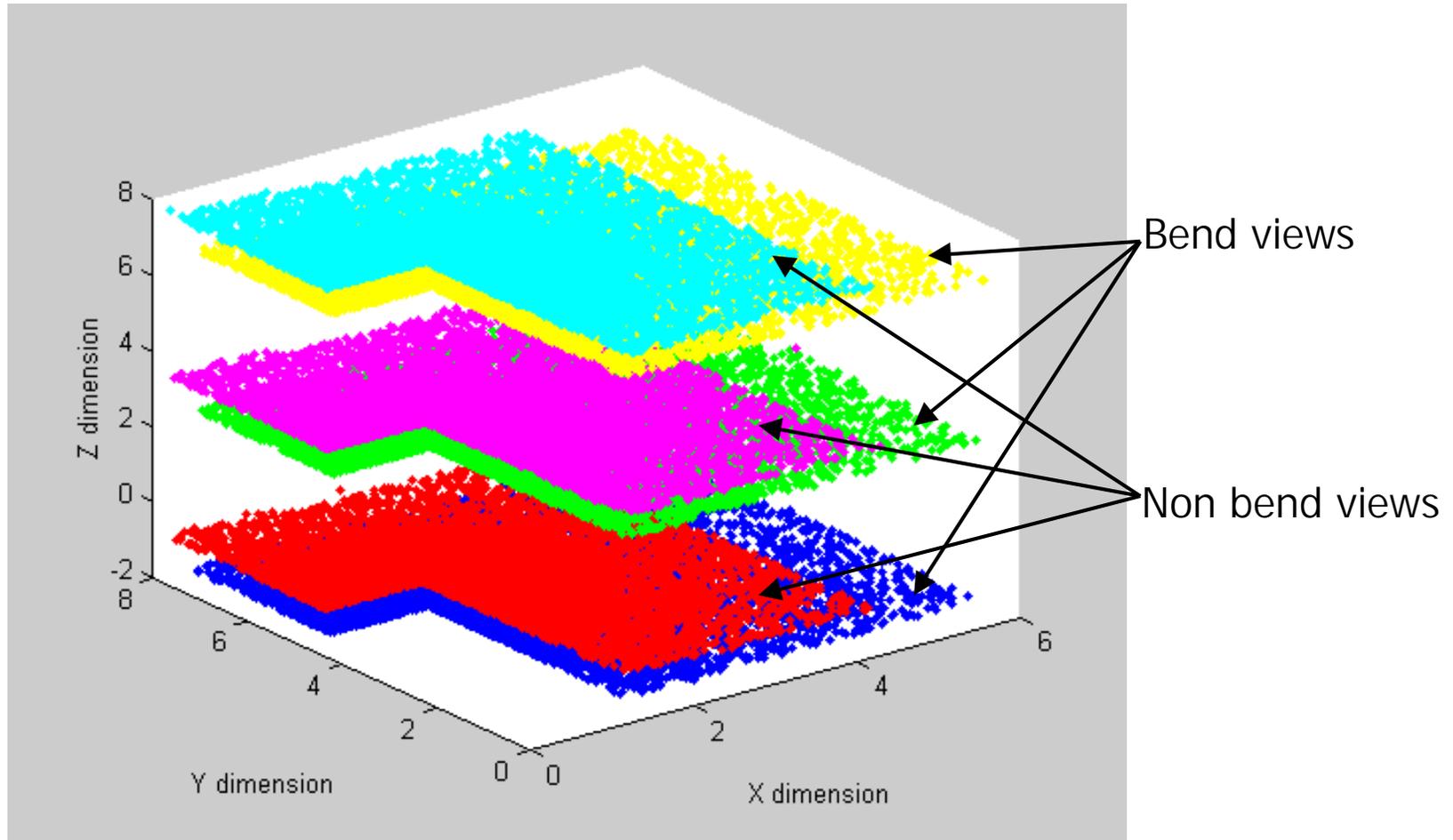
- Total N° of Hits (6 planes): 260,855
- Avg. No of tracks per BCO (1 plane): 25.14
- Avg. No hits per BCO (1 plane): 58.35
- Avg. No of hits generated by a track crossing a single sided plane: 2.32

The input bandwidth into the L1 Trigger is about 1Gb/s per Highway per Half Plane.

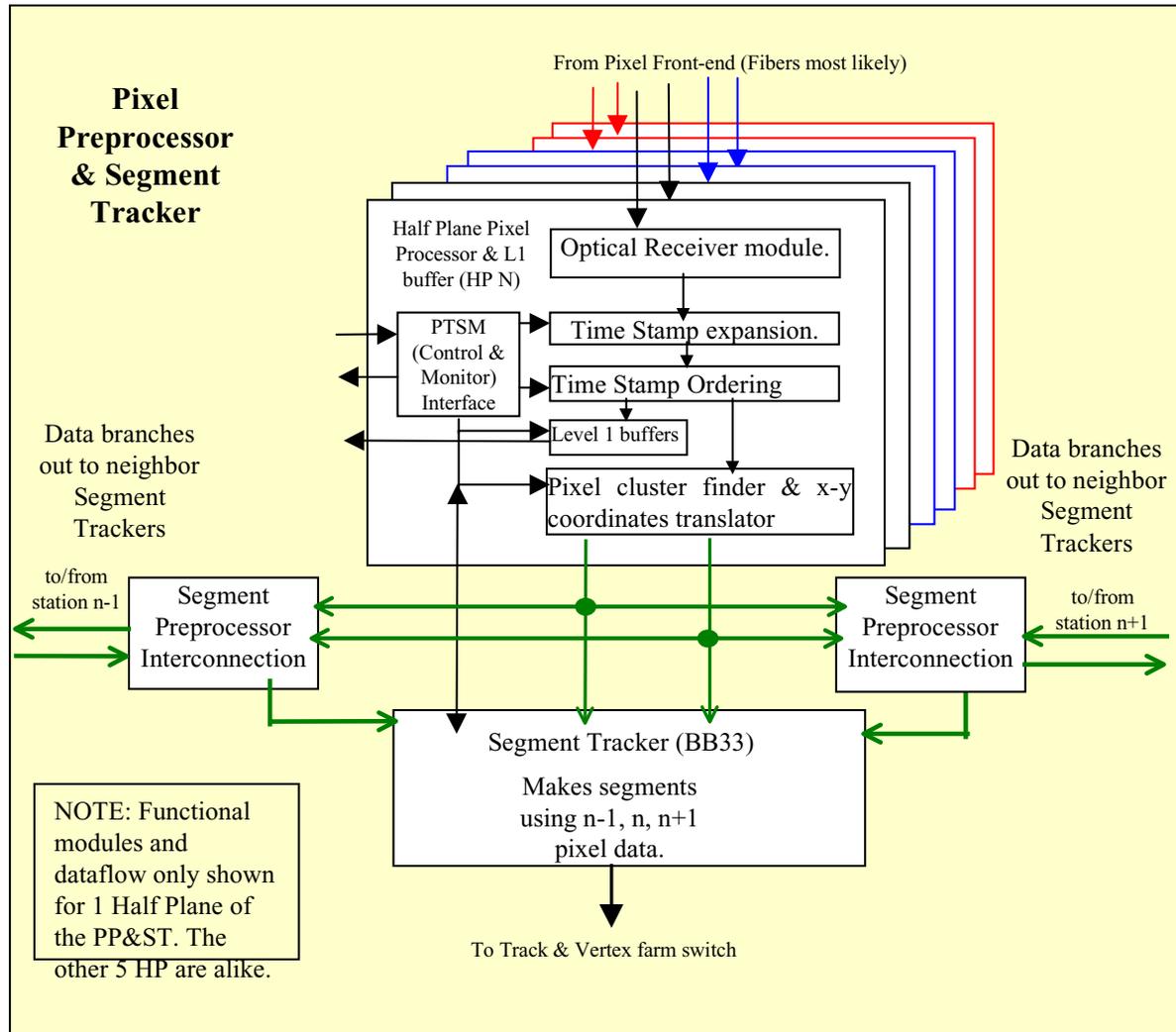


The Simulation Input data file (3)

Bend view half planes are larger than non bend view half planes

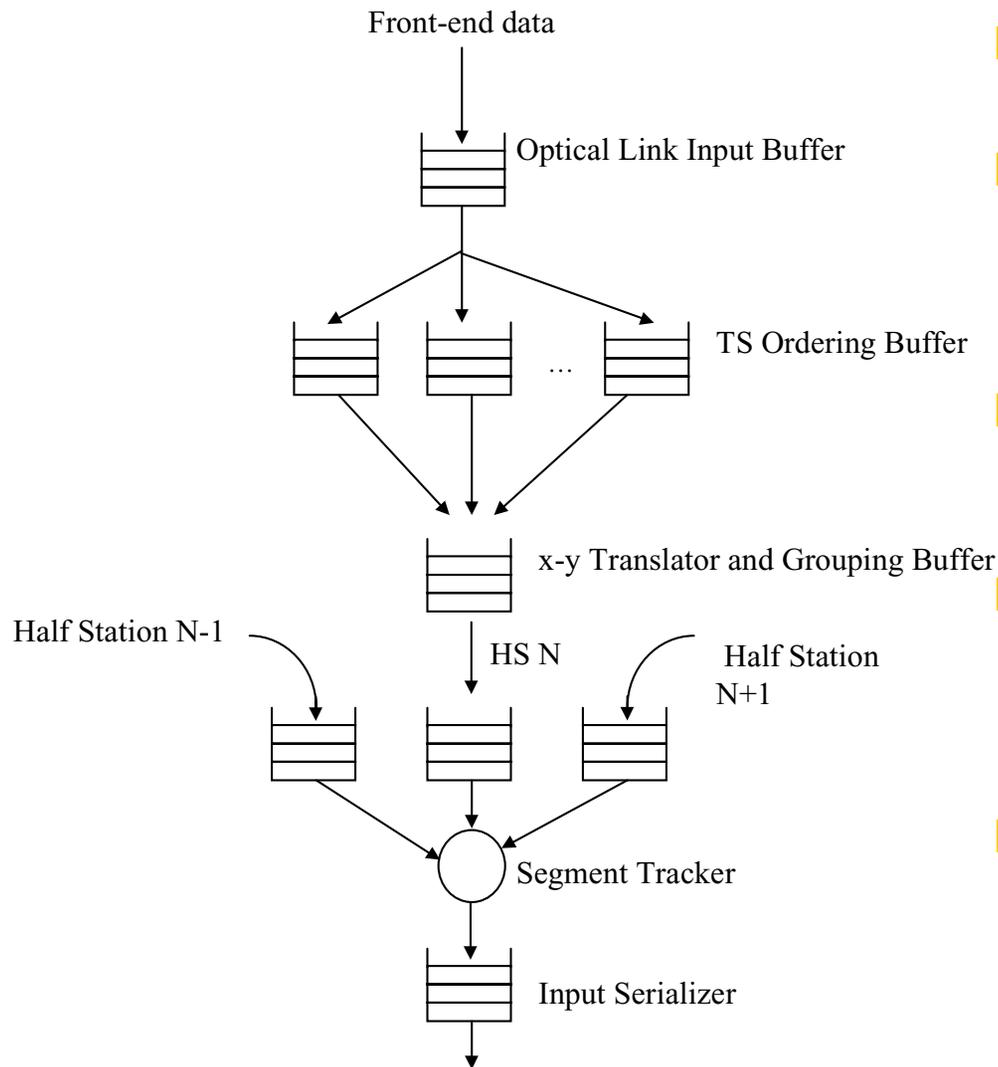


Pixel Preprocessor and Segment Tracker (PP&ST)



- The PP&ST block diagram is only for data flow analysis purpose and may not be exhaustive.
- Some Pixel Processors and a Segment Tracker can, probably, be designed on the same board.
- Pixel Preprocessors must branch their outputs to neighbor Segment Trackers.

Queuing model of the Pixel Preprocessor



- The PP&ST must be as close as possible to 100% efficient.
- The PP&ST is very time constrained. Buffers are used to equalize data rate fluctuations and diminish processor dead times.
- The data analysis is done by modeling the PP&ST processes and buffers as stochastic processes.
- Probability distributions are obtained whenever possible otherwise the 1st and 2nd moments are calculated.
- The variables to estimate are:
 - Queue sizes.
 - Data queuing and service times.
 - Channel utilization factors.

The Optical Link Input queue (1)

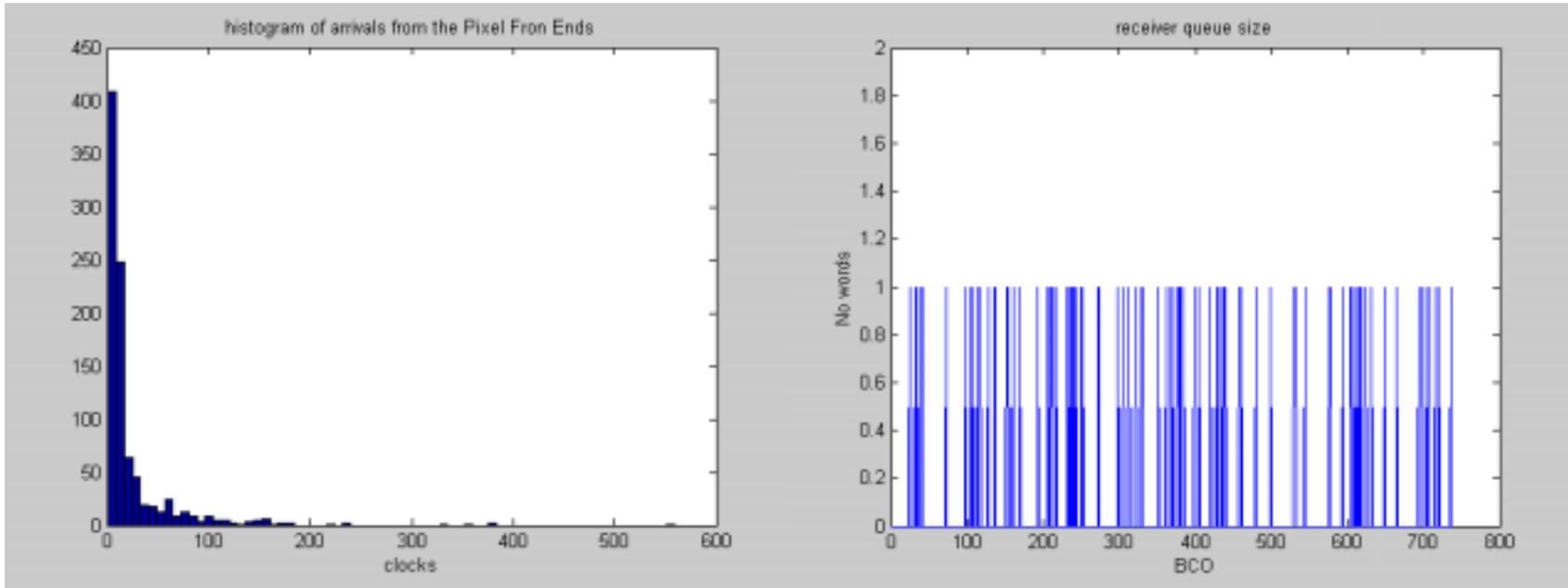
- The Input Link Buffer queue is fed by the Optical Receiver electronics. A single optical channel has a maximum bandwidth of about 2 Gb/s. This is equivalent to 250Mby/s or 125 Mega-16bit words/s, which is close to the maximum frequency that a current FPGA can handle.
- Since the processing time is deterministic, the mean Input Link Buffer output rate, μ , is constant and its variance is 0. If μ is greater than the maximum input bandwidth of the optical channel (125 Mw/s), the Input Link Buffer size needed is just 1 word deep.
- If μ is not greater than the maximum input bandwidth, it must be greater than the average input bandwidth of the optical channel λ to avoid instability. In this case the buffer behaves as a M/D/1 queue. The value of λ is directly proportional to the clock frequency of the input receiver and the Input Link Buffer's utilization factor.
- In the later case, the average queue size can be calculated by:

$$E(N_q) = \frac{\rho}{1-\rho} - \frac{\rho^2}{2(1-\rho)} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}$$

The arrivals at the receiver queue are exponentially distributed with $\lambda=0.26$ hits/clock and $\mu=0.5$ hits/clock.

$$\rho = \frac{\lambda}{\mu} = \frac{0.26}{0.5} = 0.53 \quad E(N_q) = 0.82$$

The Optical Link Input queue (2)



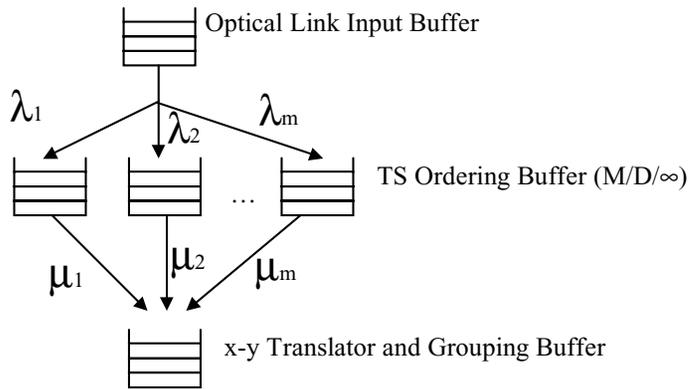
Exponential Pixel Hit arrival distribution at the receiver's queue.

Receiver queue size

Note: Pixel data is already scrambled and delayed by the Pixel Detector readout

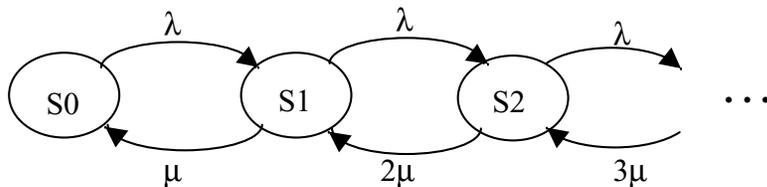
The TS-ordering queues (1)

TS- ordering queue model:



- The number of queues in the TS ordering process follows a $M/D/\infty$
- The service time μ is constant and programmable (e.g. $\mu=159$ BCOs)
- λ : queue birth rate
- $\mu_k = k\mu$: queue death rate
- $\lambda=0.1125$ events/BCO for 4int/BCO.
- $T_\lambda = 1/\lambda = 8.88$ BCOs for 4int/BCO.

State diagram



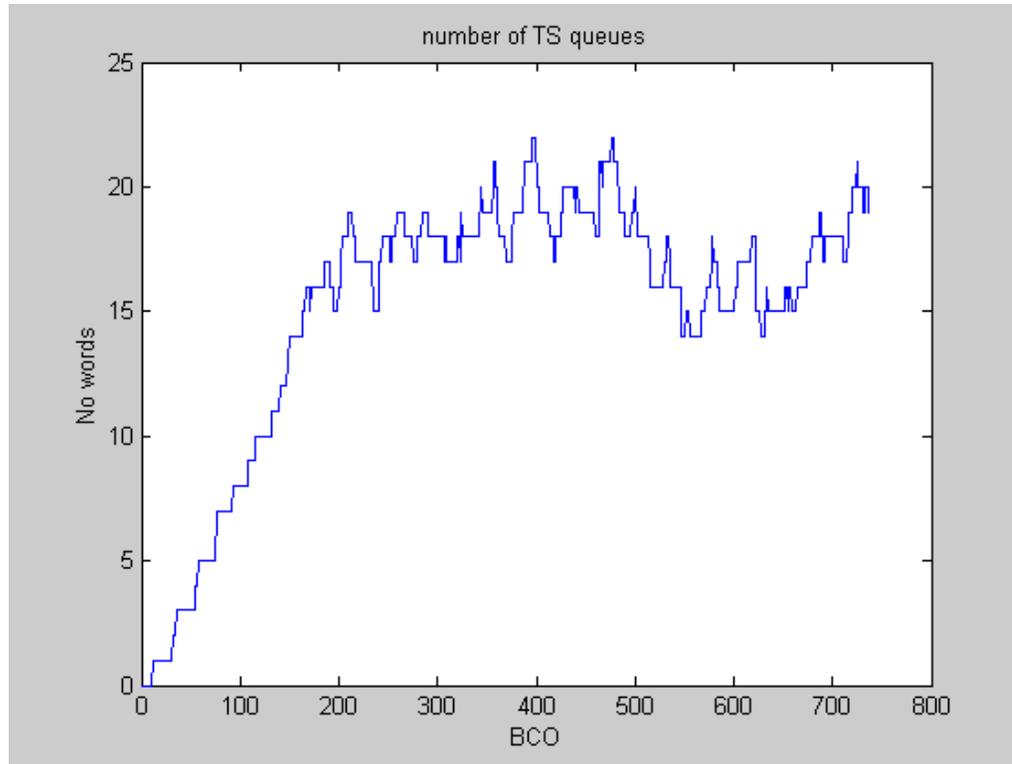
$$p_k = \frac{(\lambda/\mu)^k}{k!} e^{-\lambda/\mu} \quad k = 0,1,2,\dots$$

$$E(N_q) = \frac{\lambda}{\mu} = \frac{0.118}{0.006289} = 17.88 \text{ queues}$$

Using Little's formula:

$$T = \frac{E(N_q)}{\lambda} = \frac{1}{\mu} = 159 \text{ BCOs}$$

The TS-ordering queues (2)



- The simulation of about 750 BCOs shows a result near to predicted. 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 18.38.

The TS-ordering queues (3)

We can calculate the conditional probability distribution function of queue occupancies 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 multinomial distribution:

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

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,

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

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

$$P(M_1(t) = m_1, M_2(t) = m_2, \dots, M_n(t) = m_n | N(q) = n) = \frac{m!}{m_1! m_2! \dots m_n!} p_1^{m_1} p_1^{m_2} \dots p_1^{m_n} \cdot e^{-\lambda t} \frac{(\lambda t)^m}{m!} \quad (1)$$

since all the TS are equally probable $p_1 = p_2 = \dots = p_n = 1/n$

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

The TS-ordering queues (4)

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

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 $\lambda t/n$. 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 TS-ordering queues (5)

We can further simplify the job if we are only interested in the average total number of words in all the TS-ordering queues. 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.

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

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

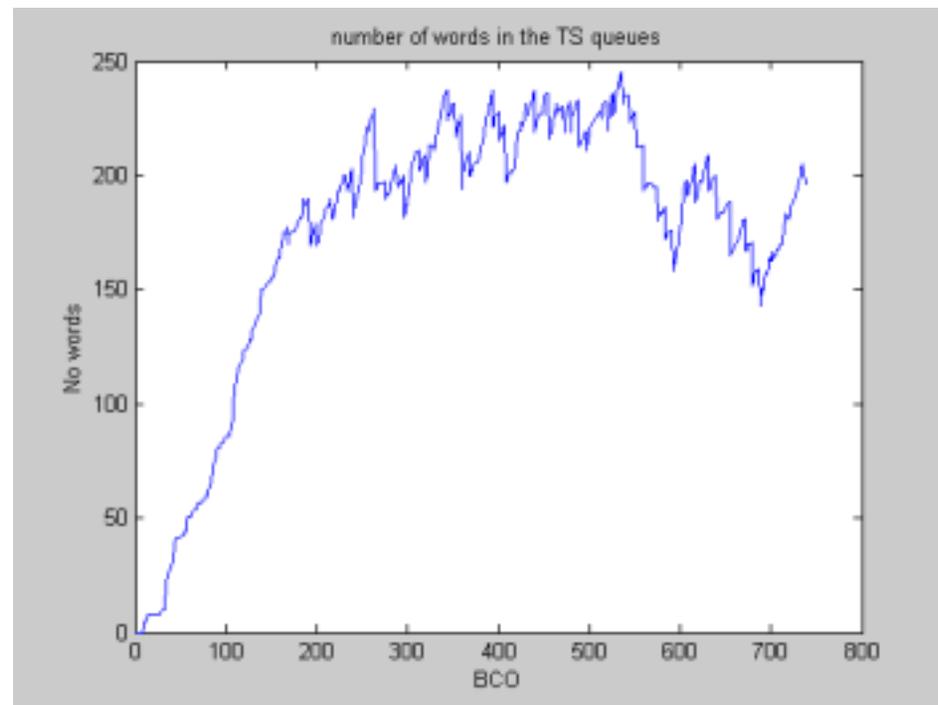
$$\mu = \frac{\text{Avg_NoTS_ordering_queues}}{\text{Deterministic_queuing_time} \times 14\text{clk/BCO}}$$

$$\mu = 0.242587$$

$$\rho = 0.993966$$

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

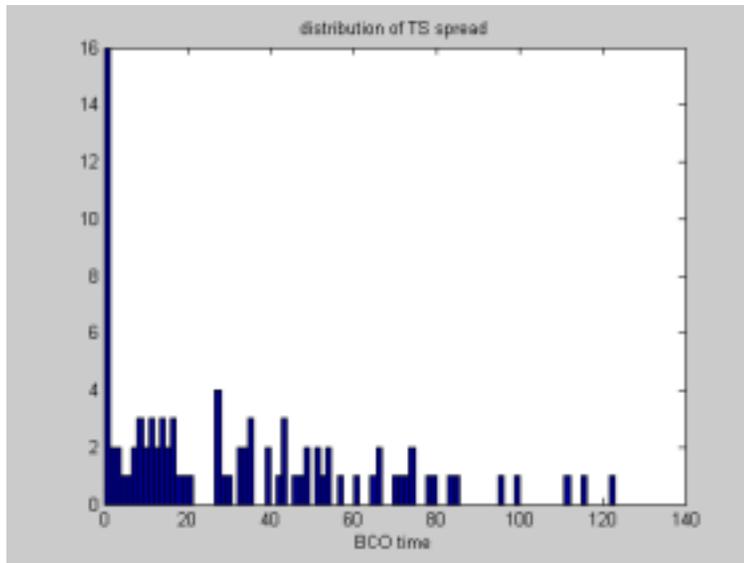
The simulation of about 750 BCOs shows an average number of words of 202.8 after the transitory



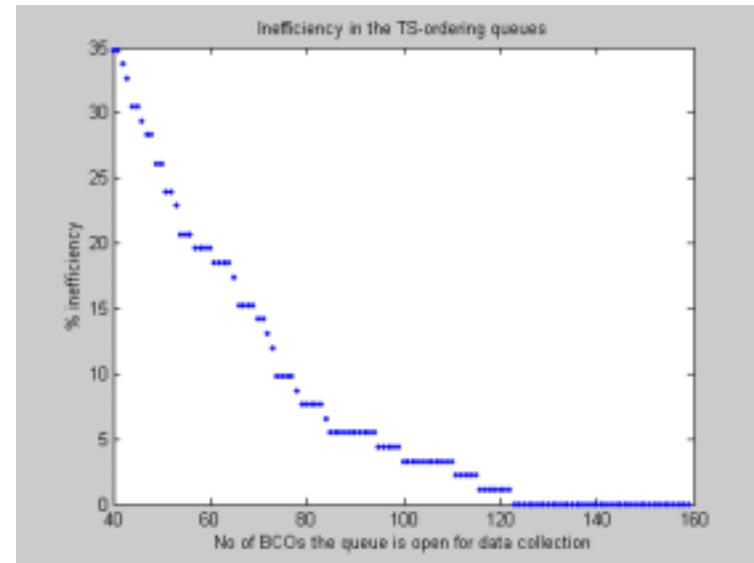
The TS-ordering queues (6)

In the previous model the TS-ordering time-out was fixed at 159 BCOs. We can let the time-out be a parameter and study the data inefficiency based on the time-out value.

Distribution of TS spread in the data



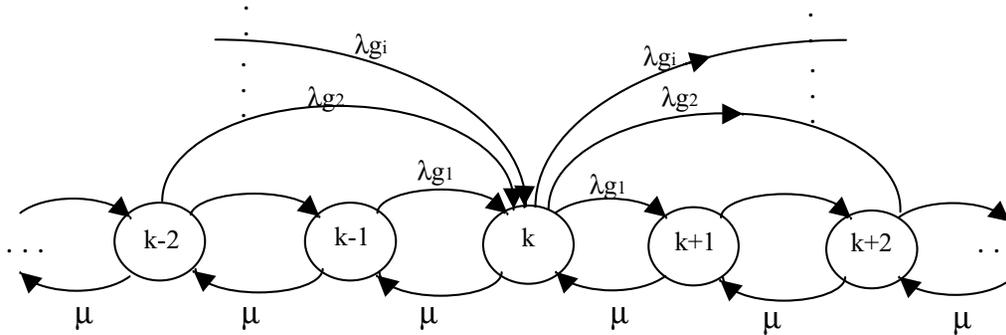
Inefficiency in the TS-ordering queues



Inefficiency is due to data arriving to the TS-ordering queues after the queue has been closed and dispatched to the XYPC input queue

The x-y pixel cluster (XYPC) queue

- Queuing model : M/M/1 with “bulk” arrivals
 - The x-y buffer customers are data bulks of variable size.
 - These queues are stored in the x-y buffer in one clock cycle.



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 are:

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

$$\lambda p_0 = \mu p_1$$

The bulk M/M/1 queue size in equilibrium suffers a “modulation” effect caused by the changing size of the events (bulks). This modulation is reflected in the discrete convolution shown in equation (*). As we know, discrete convolutions are much easily handled in the z-transformed plane because they turn into the product of the z-transforms. The z-transform of the probability distribution is:

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

The x-y pixel cluster (XYPC) queue (2)

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

The utilization factor $\rho=1-\rho_0$ can be obtained from $P(1)=1$: $\rho = \frac{\lambda G'(1)}{\mu}$

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.

The average queue size can be directly calculated from (**) using the method of moments.

$$E(N) = \left. \frac{dP(z)}{dz} \right|_{z=1}$$

After some algebra, $E(N) = \frac{2\lambda G'(1) + \lambda G''(1)}{2(\mu - \lambda G'(1))}$ Of course, this equation depends on the gk distribution.

If we assume that gk follows a Poisson distribution:

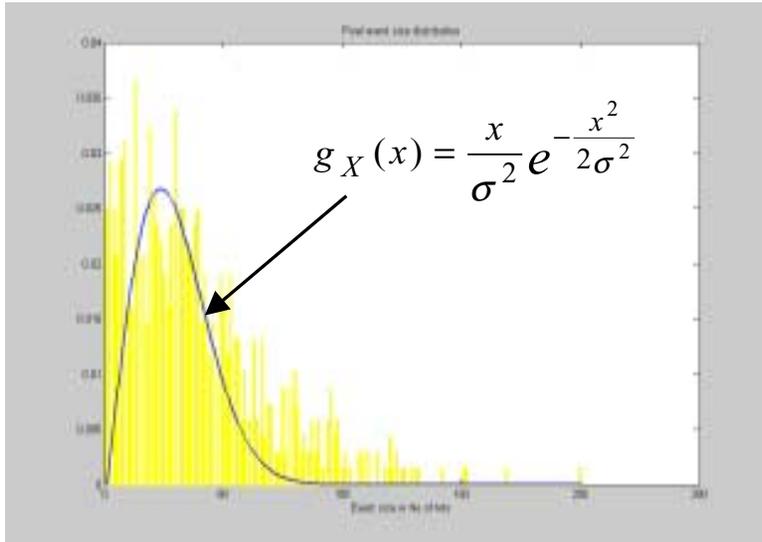
$$E(N) = \frac{2\lambda\alpha + \lambda\alpha^2}{2(\mu - \lambda\alpha)} \quad \text{It can also be expressed in terms of } \rho, \quad E(N) = \frac{2\rho + \rho\alpha}{2(1-\rho)}$$

using $\lambda=0.0083$, $\mu=0.1072$, $\alpha=25$, $E(N)=0.103$

However, the hit distribution does not look Poisson.

The x-y pixel cluster (XYPC) queue (3)

Modeling g_i using a Rayleigh distribution



The Rayleigh distribution is a continuous pdf. Its Fourier transform can be calculated as

$$G(w) = \int_{-\infty}^{\infty} g(x) e^{-jwx} dx = \int_{-\infty}^{\infty} \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} e^{-jwx} dx$$

$$G(w) = jw\sigma\sqrt{\frac{\pi}{2}} e^{-\frac{w^2\sigma^2}{2}}$$

And the z-transform is: $G(z) = \sigma\sqrt{\frac{\pi}{2}} \ln(z) z^{-\frac{\sigma^2}{2}}$

Then expected number of queues in the bulk M/M/1 process is:

$$E(N) = \frac{2\lambda G'(1) + \lambda G''(1)}{2(\mu - \lambda G'(1))} (*)$$

Substituting the derivatives, $E(N) = \frac{2\sqrt{\frac{\pi}{2}} \frac{\lambda\sigma}{\mu} - \sqrt{\frac{\pi}{2}} \frac{\lambda\sigma}{\mu} (1 + \sigma^2)}{2\left(\mu + \lambda\sigma\sqrt{\frac{\pi}{2}}\right)} (**)$

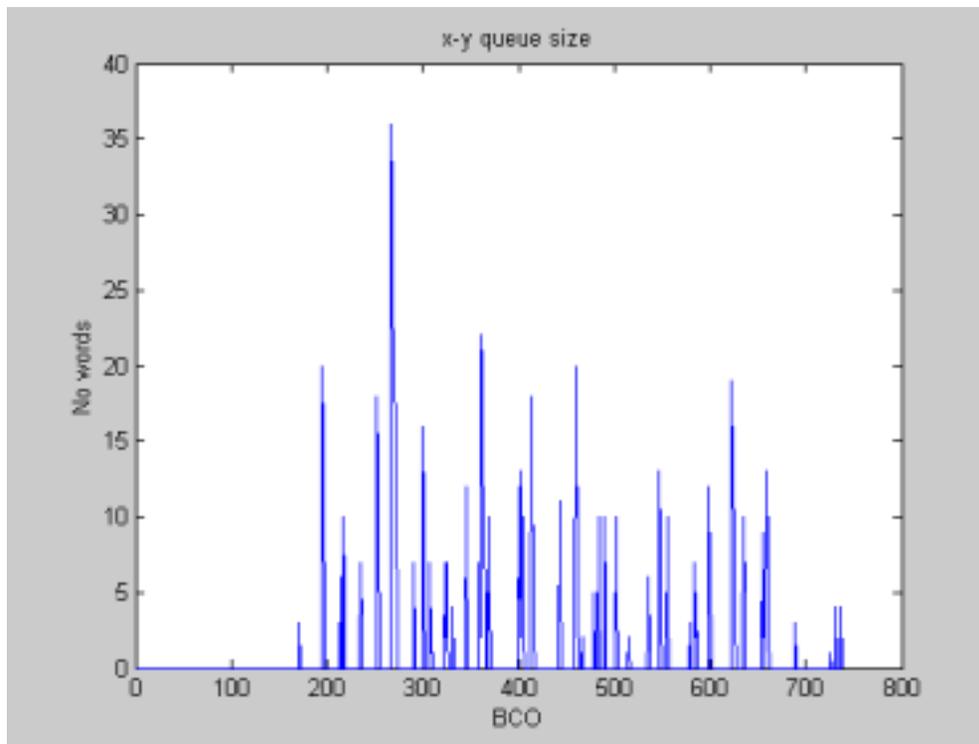
Also, using $\rho = \frac{\lambda G'(1)}{\mu}$

equation (**) can be written as

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

The x-y pixel cluster (XYPC) queue (4)

A 750 BCO simulation shows that the XYPC queue is empty most of the time and peeks suddenly every time a bulk fills it up. Since the bulk interdeparture time is fairly small compared to the bulk interarrival time, the queue returns to 0 quickly most of the time



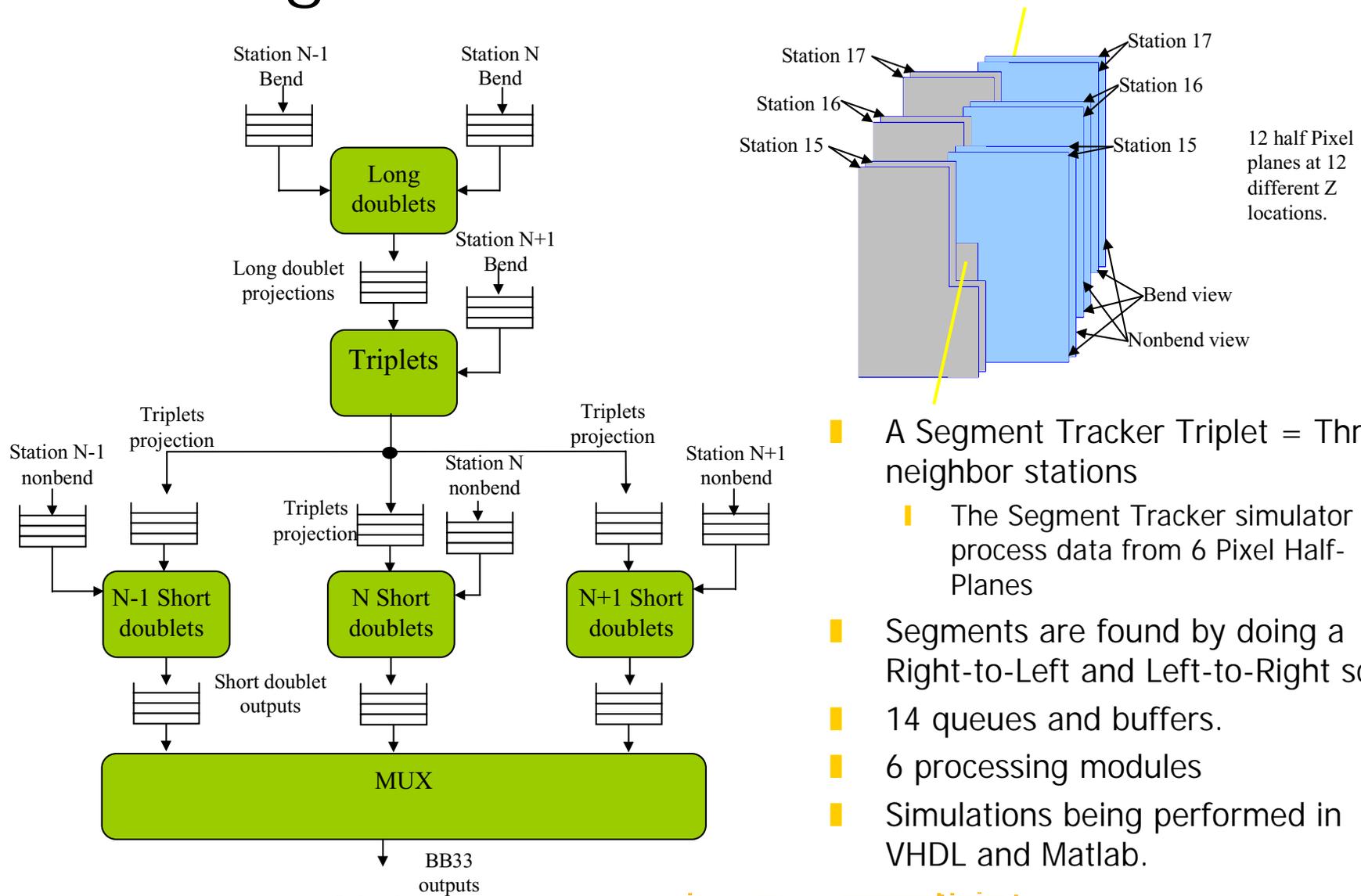
Parameter fit and mean queue size table:

The second column of the table shows the estimated parameter $\hat{\sigma}$ for the input data distribution of each half plane. The third column shows the average queue size.

Half Pixel Plane	$\hat{\sigma}$	E(N)
N-1 bend	31.15	4.02
N-1 non bend	21.64	1.94
N bend	31.74	4.18
N non bend	23.50	2.29
N+1 bend	31.87	4.21
N+1 non bend	21.97	2.0

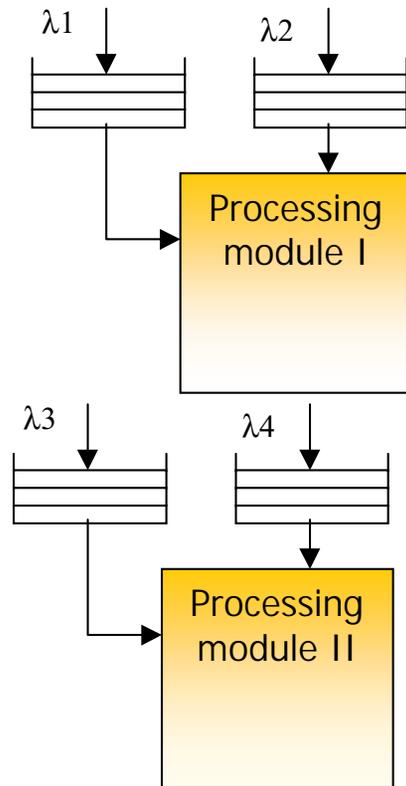
Note: N is plane 16. One of the central planes of the Pixel Detector

The Segment Tracker Architecture

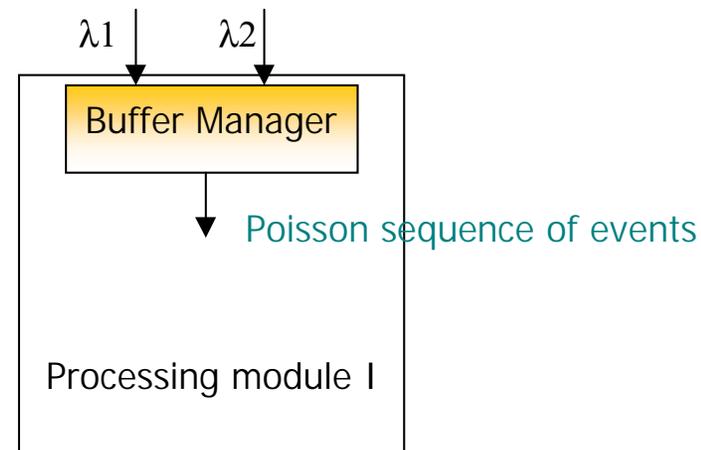


- A Segment Tracker Triplet = Three neighbor stations
 - The Segment Tracker simulator process data from 6 Pixel Half-Planes
- Segments are found by doing a Right-to-Left and Left-to-Right scan
- 14 queues and buffers.
- 6 processing modules
- Simulations being performed in VHDL and Matlab.

Segment Tracker Processing Modules

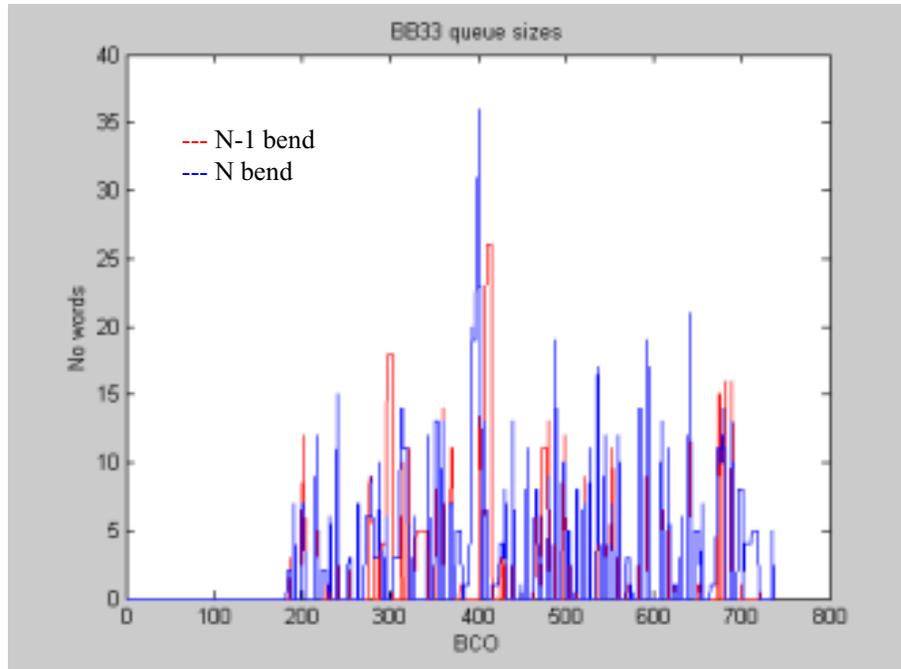


- A Processing module receives two asynchronous data streams.
- The processing module's first stage is a buffer manager which synchronizes the data inputs by BCO.
- The BB33 pipelining is organized by BCO:
 - Two processing modules always work on two different BCOs.
- The sequence of synchronized events out of the buffer manager is itself a Poisson process.



Segment Tracker Processing Modules (2)

N-1 bend and N bend plane queue sizes



The average queue sizes calculated by the simulations are:

N-1 bend: 13.57

N bend: 11.64

The mean of queued events in the Long Doublet process is:

$$E(R_q) = \frac{\rho}{1-\rho} = 0.538 \quad \text{where} \quad \rho = \frac{\lambda}{\mu} = 0.35 \quad (1)$$

We can estimate the average number of hits in the N-1 bend and N bend queues by multiplying the result in (1) by the average event size

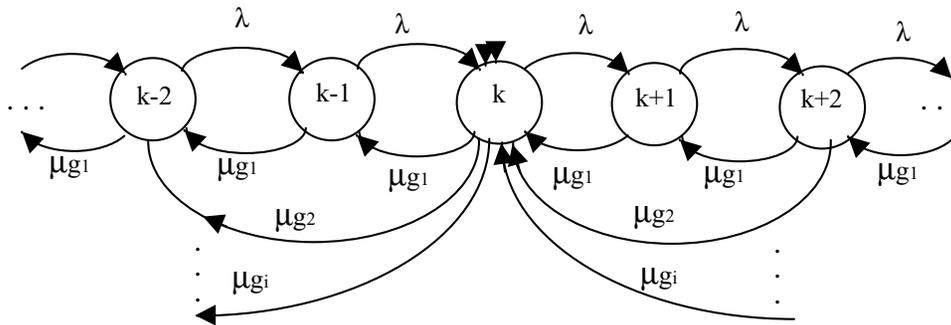
$$E(N_q) = E(E_v) * E(R_q) = 20.82 * 0.538 = 11.02$$

Queue size Mean and σ	Calculated		Simulated	
	Mean	σ	Mean	σ
N-1 bend	11.02	?	13.5796	8.8559
N-1 non bend	?	?	11.6466	9.0579
N bend	11.75	?	15.9897	8.0108
N non bend	?	?	12.1828	9.2420
N+1 bend	?	?	14.8655	9.6172
N+1 non bend	?	?	12.9621	9.9339
N triplet projection bend	?	?	2.4667	5.0369
N-1 projection non bend	?	?	0.1315	0.6177
N projection non bend	?	?	0.0259	0.2845
N+1 projection non bend	?	?	0.0259	0.2845

Note: N is plane 16. One of the central planes of the Pixel Detector

Segment Tracker Processing Modules (3)

Analysis of the BB33 queues as a "bulk" service process:



$g_i = \text{Prob}[\text{bulk size is } i]$, then

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

g_i has a "modulation" effect over the distribution of the queue size p_i

The equilibrium equations for the bulk arrival system are:

$$\left. \begin{aligned} (\lambda + \mu) p_k &= \lambda p_{k-1} + \mu \sum_{i=k+1}^{\infty} p_i g_{i-k} & k > 1 \\ \lambda p_0 &= \mu \sum_{i=1}^{\infty} p_i g_i \end{aligned} \right\} \text{Solve using the Z Transform } P(Z) = \sum_{i=1}^{\infty} p_i Z^k$$

Solution for fixed bulk size is

$$p_k = \left(1 - \frac{1}{z_o}\right) \left(\frac{1}{z_o}\right)^k$$

The mean queue size is

$$E(N) = z_o \left(\frac{1}{1 - \frac{1}{z_o}} \right) = \frac{z_o^2}{z_o - 1}$$

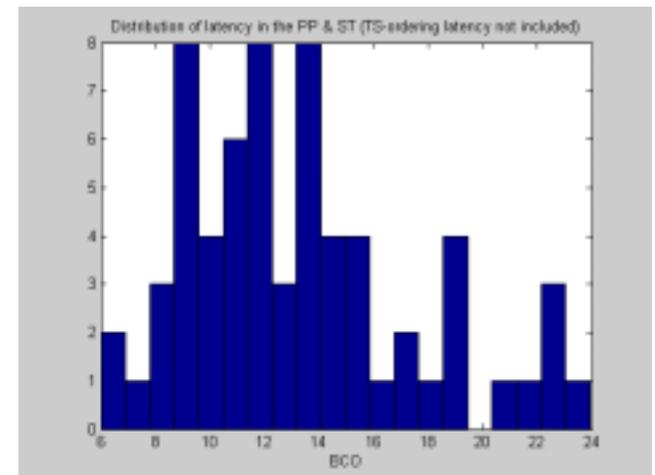
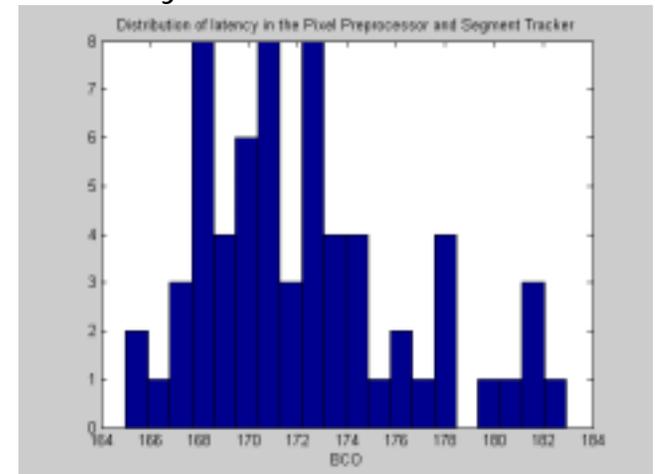
Segment Tracker Processing Modules (4)

Latency and processing times:

We define latency in a specific stage of the Pixel Preprocessor and Segment Tracker modules as the time between the arrival of the first hit of an event and the time when the processed event departs from that stage

Pixel Preprocessor	Processing Distribution	Avg. Processing Time
Receiver interface	deterministic	2 clock/hit
TS-ordering (queue)	deterministic	159 BCOs (4452 clocks)
TS-ordering (hits)	exponential	18.8 clocks/hit
X-Y translation & grouping	Rayleigh	1+Nohits/group=3.2clk/hit
Segment Tracker	Processing Distribution	Avg. Processing Time
Long Doublet	Rayleigh: $1\text{clk} + \text{AvgNoQuerys}/\text{BCO} * (2 + \text{AvgNomatches}/\text{query})$	
Triplet	Rayleigh: $1\text{clk} + \text{AvgNoQuerys}/\text{BCO} * (2 + \text{AvgNomatches}/\text{query})$	
Short Doublets	Rayleigh: $1\text{clk} + \text{AvgNoQuerys}/\text{BCO} * (2 + \text{AvgNomatches}/\text{query})$	

Latency:



Summary

■ What have we learned?

- The models and simulations show good consistency
- The queuing models reveal strategies to improve the system architecture, for instance
 - | Adding highways to the Pixel Front-ends
 - Make the highway average distribution fairly uniform
 - The Pixel Preprocessors and Segment Trackers can process bigger portions of the Pixel Detector
 - We have one more degree of freedom in the bandwidth and computational power required in the Pixel Preprocessors and Segment Trackers
 - | Some processing modules in the BB33 algorithm are very similar
 - | Some processing modules in the BB33 algorithm are underutilized
 - A hardware/firmware realization of the BB33 could implement several underutilized processing modules into a more general one, saving silicon
 - | A buffer manager simplifies the processing functions in the BB33
 - | An BCO-based event stream out of the BB33 processing modules helps to keep queue sizes low.

Summary (2)

■ What have we learned?

■ More on modeling benefits

- | The queuing models allow us to predict new output parameters based on changes in the input parameters whenever the model distributions remain the same.

■ Simulation benefits

- | Models are hardly perfect, simulations provide a closer approach to reality
- | Simulations are based on models. The input data can be adapted to different configuration

■ Analysis results

- Queue sizes and communication channel bandwidths are within reasonable margins. Once we approach the design stage, and a full VHDL code has been generated, we can refine some of the parameters shown in the modeling and simulation to optimize the design.

Summary (3)

■ Any problems?

- A significant point shown by the data analysis is the latency created by the need to reorder the Pixel data because the asynchronous nature of the Pixel Detector readout scrambles it.

■ Future work

- Finish the last part of the BB33 algorithm (with help from Erik and Mike W.)
- Complete VHDL code design and simulation of the BB33 algorithm
- Complete VHDL code design and simulation of the Pixel Preprocessor algorithm
- Study other algorithms?
- Large scale simulations (PC farm)?
- Study pixel clustering across the columns of the Pixel chip
- Study hardware/software implications of using Pixel hit analog information