Regional Readout latency simulation for upgraded ATLAS Inner Detector strips

G. Crone

March 6, 2012

1 Introduction

In order for the Region of Interest (RoI) driven level 1 track trigger to work, the data from the RoI must be retrieved and processed within the maximum level 1 latency supported by the detectors. This is currently expected to be $\sim 15\mu s$ determined by the muon system. If we allow $5\mu s$ for the Level 0 decision and $5\mu s$ for the L1 track processing, that leaves $\sim 5\mu s$ to get the RoI data from the front end chips to the L1 track processor. To investigate readout schemes that allow the RoI data to be read in $\sim 5\mu s$ a simple discrete event simulation has been written.

The RoI or 'Regional Readout Request' (R3) Data latency simulation is a python script which uses the SimPy discrete event simulation package for event scheduling and ROOT for histogramming.

1.1 SimPy

The SimPy package [1] is fairly simple to use. Actions can be scheduled for a specific time or for when a 'Resource' becomes available. A 'Resource' can be blocked for a specific time by one action, making it unavailable to other actions. This makes it easy to model contention for a real resource such as the inter chip links on a hybrid.

A SimPy 'Resource' can be allocated on a first come first served basis or according to the priority of the requester. This allows us to model the situation where R3 packets are given higher priority than L1 packets.

2 The simulation script

The simulation script currently represents a 'vertical slice' from the L0/L1 trigger system to a single Si detector hybrid. Each hybrid has its own dedicated lines to the end of stave / GBT so there is no contention among the hybrids on a stave.

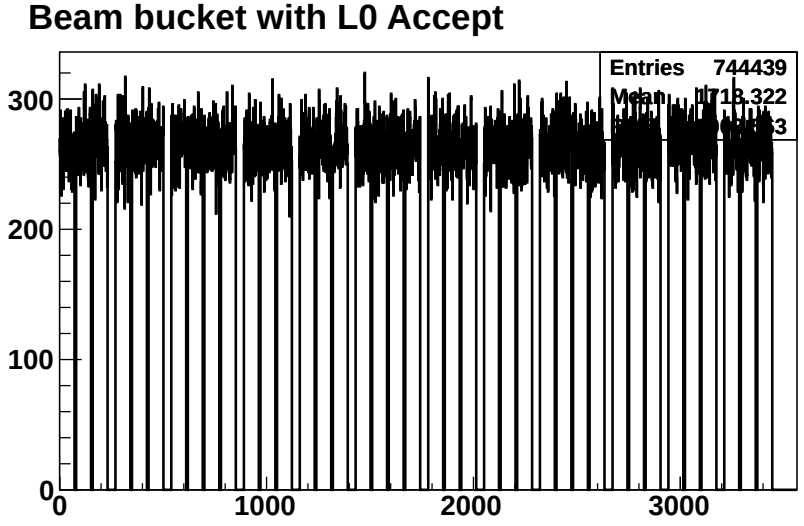

Figure 1: Distribution of events among filled beam buckets

2.1 Event generation

Level0 Accepts are generated randomly among the filled bunch crossings[1] at a rate determined by the variable option.l0Rate (the full list of script options is given in Appendix A). Figure 1 shows the distribution of events among filled beam buckets and figure 2 shows the gap between L0 accepts in cycles of the 40MHz LHC clock with a requested L0 rate of 300kHz.

A random subset of the L0 Accepts (determined by option.r3percent) are selected for having an R3 involving our modelled hybrid. An independent random subset of L0As are selected for having an Level1 Accept at a rate determined by option.l1Rate.

2.2 Event process

For each L0A generated, the following steps are taken.

- Delay for L0A latency (fixed)
- If marked for R3
  - Delay for fibre transmit latency
  - Start hybrid/stave transfer
  - Wait for it to complete
  - Delay for fibre transmit latency
- If no R3 - Delay for average R3 latency[2]

[1]The bunch crossing structure is modelled in bxgenerator.py which keeps track of which clock cycle contains the next filled bunch. This uses the LHC bunch structure described in [2]

[2]Before the 1st R3 is complete this is a fixed time based on number of chips per HCC and link speed

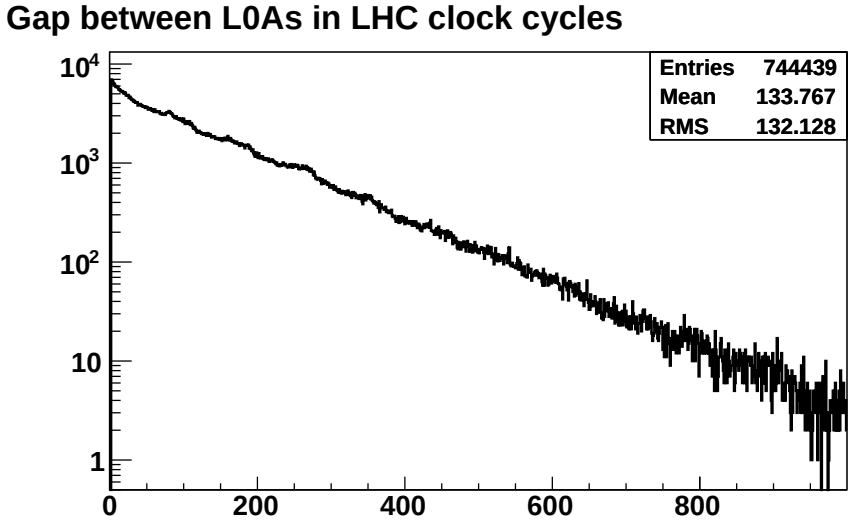

Figure 2: Number of 40MHz clock cycles between L0 Accepts

- If marked for L1A
  - Delay for fibre transmit latency
  - Start hybrid/stave transfer
  - Wait for it to complete
  - Delay for fibre transmit latency

2.3 How many data packets come from each chip

Earlier studies based on athena simulations of the upgrade geometry have produced estimates of the the number of data packets for R3 and L1 data. To simulate the correct number of data packets, for each L1A each chip uses

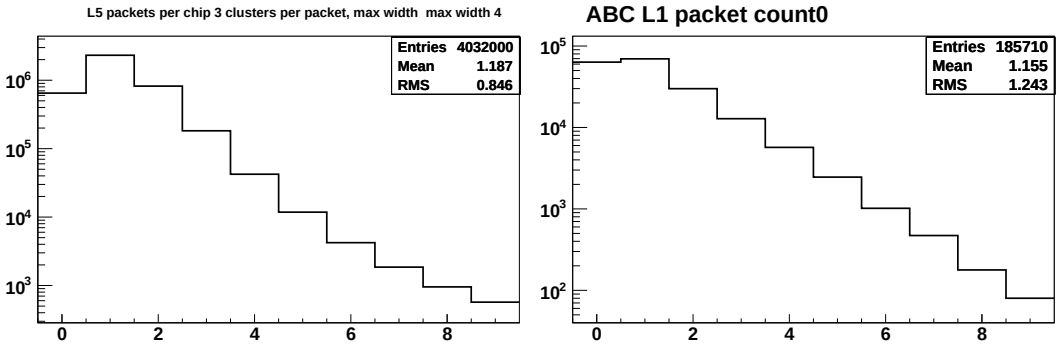

Figure 3: Number of L1 packets for 200 pileup from athena simulation (left) and discrete event simulation (right).

the python expovariate random number generator with the lambda set to 1/the mean L1 packet size.

To simulate the correct number of data packets for R3 data, each chip decides whether or not it has a packet to send based on a uniform random

number generator with the correct fraction of empty chips taken from the R3 packet size histogram.

2.4 ABC to ABC/HCC data transfer

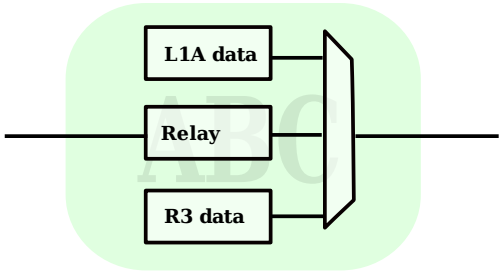

Figure 4: Data links within ABC chips.

The transfer of data between chips is complicated in that there are three sources of data, R3 data, L1 data and data from the relay buffer from the next chip in the line (as shown in Figure 4). This is modelled using priority queue Resource objects and adjusting the priority of different packet transfers dynamically.

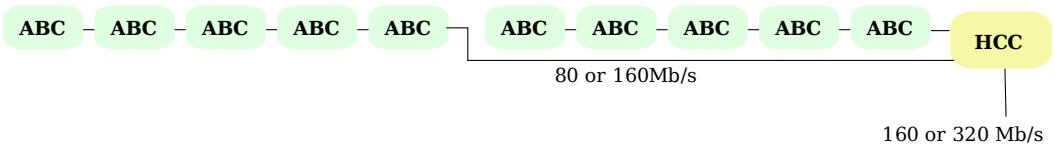

Figure 5: Data links between ABC chips, HCC and stave. ABC chips are in 2 groups of 5.

Depending on its position in the chain, each chip will prioritise data from the relay buffer over its own data for a different number of packets.

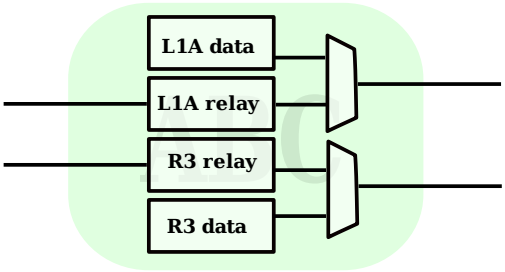

Figure 6: Data links within ABC chips.

An alternative scheme where the L1A data and the R3 data are sent on separate dedicated links is also modeled (Figure 6).

2.5 HCC to end of stave data transfer

The transfer of data along the stave from the HCC is modelled using one or two 'Resource' objects depending on whether the link is logically shared

between R3 and L1 data or split into two dedicated links. The 'Resource' objects are used with priorities with the R3 having greater priority than L1 so in the case of a shared link, the HCC will send R3 data before L1.

3 Results

Except where stated, the following results are all for runs with the basic parameters:

- Pileup 200
- 25ns bunch spacing with simple deadtime 2 (which is equivalent to 50ns with no simple deadtime)
- L0 rate 300kHz
- L1 rate 75kHz
- R3 rate 3kHz
- data packet size 60bits
- 5 ABC chips per channel to HCC

3.1 R3 data to HCC

The number of packets waiting for access to the data link out of a chip can be seen from the length of the Resource queue. An example is shown in figure 7.

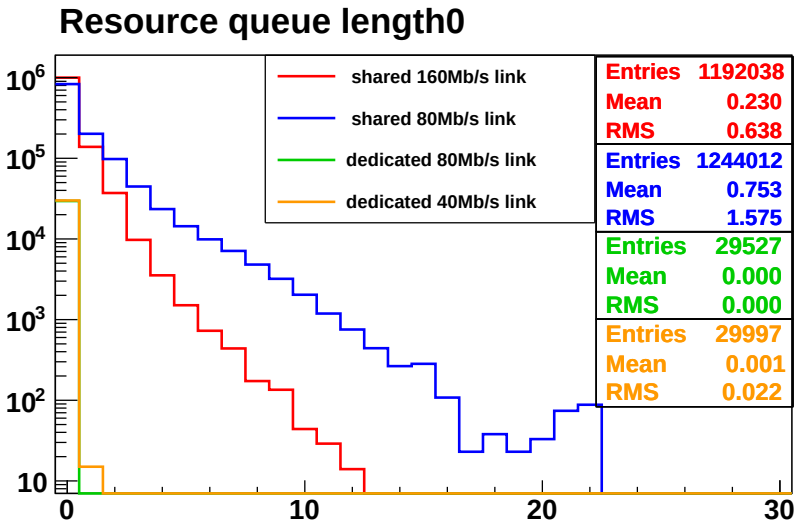

Figure 7: Resource Queue length for the ABC chip closest to the HCC

3.2 R3 data to end of stave

The time taken to transfer R3 data from chip to end of stave for 3 scenarios is shown in figure 8. A 160Mb/s link shared between L1 accepted data and R3 data is usually faster than a dedicated 80Mb/s link but has longer tails so is sometimes longer.

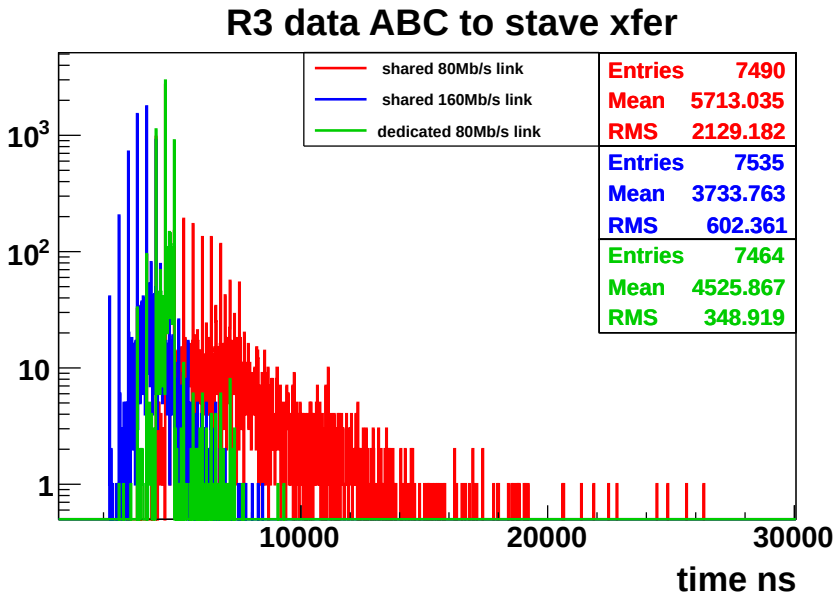

Figure 8: time taken to get R3 data from ABC chips to the end of stave

Bandwidth Mb/s		% R3 data received in			
chip	stave	$< 4\mu s$	$< 4.5\mu s$	$< 5\mu s$	$< 5.5\mu s$
40+40	80+80	0	0	0	0.0273
80+80	80+80	0.0545	0.695	0.695	4.81
80+80	160+160	2.62	98	98.3	98.6
80+80	shared 160	2.22	64.9	98.5	98.8
80+80	shared 320	26.4	98.3	98.9	99.3
shared 160	shared 160	71.7	92.7	96.6	98.5
shared 160	shared 240	94.6	97.4	98.8	99.4
shared 160	shared 320	96.7	98.5	99.1	99.5
shared 80	shared 160	1.74	51.4	56.5	62.1

Table 1: Amount of R3 data received in a given time for various bandwidth sharing scenarios

The fraction of R3 data that reaches the end of stave within a given time for various scenarios is shown in table 1. For each of the chip bandwidths given it should be remembered that there are 2 groups of 5 chips feeding in to an HCC so the aggregate bandwidth is double the number in the 1st column.

Having dedicated links for R3 and L1 data end to end allows us to use a different data format for R3 packets. If we use dedicated links and reduce the R3 data packet size to 40 bits we get the results shown in table 2.

Bandwidth Mb/s		% R3 data received in			
chip	stave	$< 4\mu s$	$< 4.5\mu s$	$< 5\mu s$	$< 5.5\mu s$
80+80	80+80	30.3	61.2	88.3	99

Table 2: Amount of R3 data received in a given time with 40 bit R3 data packets

Just for a comparison, table 3 shows the amount of data received in a given time when all 10 ABC chips are readout serially by the HCC.

Bandwidth Mb/s		% R3 data received in			
chip	stave	$< 4\mu s$	$< 6\mu s$	$< 8\mu s$	$< 10\mu s$
80+80	shared 160	0	0.0135	18.5	97.3
shared 160	shared 160	0	16.9	78.4	92.2

Table 3: Amount of R3 data received in a given time for various bandwidth sharing scenarios with 10 chips per HCC connection

3.3 What if our assumptions about input L0/RoI rates are wrong?

Just using the scenario with 2 160Mb/s shared chip links and a shared 160Mb/s stave link, table 4 shows the effect of different L0, L1 and R3 rates.

Rate kHz			% R3 data received			
L0	L1	R3	in $< 5\mu s$	in $< 5.5\mu s$	in $< 6\mu s$	in $< 6.5\mu s$
300	75	3	95.4	97.9	99	99.5
300	75	15	92.9	96.4	98.2	99.2
300	75	30	89.6	93.7	96.5	98.1
500	100	5	93.6	97	98.5	99.2
500	100	25	88.8	93.3	96.3	97.9
500	100	50	82.6	88.3	92.8	95.5

Table 4: Amount of R3 data received in a given time for various input rate scenarios with 5 chips per HCC connection using 160Mb/s shared links on hybrid and stave.

References

[1] The SimPy homepage http://simpy.sourceforge.net/index.html

[2] S.Ask et al,The ATLAS central level-1 trigger logic and TTC system

A r3sim.py command line

```
./r3sim.py --help
Usage: r3sim.py [options]

Options:
  -h, --help            show this help message and exit
  -p PILEUP, --pileup=PILEUP
                        pileup to determine event size distribution (default
                        200)
  -n SIMNAME, --simName=SIMNAME
                        simulation name, used in constructing output file name
  -t RUNTIME, --time=RUNTIME
                        amount of time to simulate in seconds (default 0.025)
  -r NREPORTS, --nReports=NREPORTS
                        Number of progress reports to issue (default 10)
  -d, --dedicatedChipLink
                        Use separate channels for R3 and L1 data between chips
  --sharedChipLink      Use single shared channel for R3 and L1 data between
                        chips (default True)
  --dedicatedStaveLink  Use separate channels for R3 and L1 data along stave
  --sharedStaveLink     Use single shared channel for R3 and L1 data along
                        stave (default True)
  --l0Rate=L0RATE       Level 0 accept rate in Hz (default 300000)
  --l1Rate=L1RATE       Level 1 accept rate in Hz (default 75000)
  -b BXSPACING, --bxSpacing=BXSPACING
                        Bunch crossing spacing in ns (default 25)
  --linkSpeed=LINKSPEED
                        Inter chip link speed in Mb/s (default 160)
  --staveLinkSpeed=STAVELINKSPEED
                        Stave link speed in Mb/s (default 320)
  --r3LinkShare=R3LINKSHARE
                        Share of inter chip link speed dedicated to R3
                        (default 0.5)
  --r3packetsize=R3PACKETSIZE
                        R3 packet size in bits (default 40)
  --l1packetsize=L1PACKETSIZE
                        L1 packet size in bits (default 60)
  --r3percent=R3PERCENT
                        percentage of time one hybrid is in RoI (default 1)
  --simpleDeadTime=SIMPLEDEADTIME
                        Minimum number of BX between L0 Accepts (default 3)
  --seed=SEED           Seed for random number generator
  --chipsPerHCC=CHIPSPERHCC
                        Number of chips per HCC connection (default 10)
  --l0Latency=L0LATENCY
                        Time taken to make the level 0 Accept decision in ns
                        (default 2400)
  --outdir=OUTDIR       output directory (default /tmp)
```