

FPGA Development for the LHCb Vertex Locator Upgrade

Nicholas Mead
8064141

School of Physics and Astronomy
University of Manchester

January 10, 2016

Abstract

This document discusses two areas of FPGA development for the LHCb VELO upgrade scheduled to coincide with LHC Long Shutdown 2 in 2019. The selection of a suitable scrambling algorithm and the continued work on the event isolation flagging.

The analysis for three scrambling algorithms, required for data transfer from the front end electronics to the Data Acquisition FPGA, was compared against the theoretical predictions of scrambled data. It was found that the currently implemented VeloPix scrambler is the optimum of the choices but also that an alternative multiplicative scrambler was suitable for computer simulations.

Event Isolations Flaging (EIF) system is intended to identify and flag the easier to re-construct events in order to reduce event pill-up in the computer network. While a bug is identified in the simulated data, the VHDL developent has coninued and is ongoing. A bubble sorting algorithm, implemented by the EIF system, is in ready for testing. Once the EIF system is complete, it will be implemented in the master code for the low level interface.

Contents

24	1 Introduction	1
	1.1 The Standard Model of Particle Physics	1
26	1.2 Field Programable Gate Arrays	1
	1.3 The LHCb Experiment	2
28	1.4 LHCb Upgrade	3
	1.4.1 VELO Upgrade	3
30	1.4.2 Data Flow and Low Level Interface	4
	2 Scrambler	5
32	2.1 Scrambler Options	6
	2.2 Cross Checks	7
34	2.3 Algorithm Analysis	7
	2.3.1 Messurements of the Algorithms	8
36	2.3.2 Statistical Predictions	9
	2.3.3 Results of Analysis	11
38	2.4 Conclusion	13
	3 Event Isolation Flagging	14
40	3.1 Time Sorting Data	14
	3.2 Bubble Sorting	15
42	3.3 Isotation Checking	16
	3.4 Data Train Overflow	16
44	3.5 Current Stage of Development	18
	References	20
46	A Appendix	21
	A.1 Event Isolation Flagging	21

1 Introduction

1.1 The Standard Model of Particle Physics

Central to the modern study of particle physics is the standard model,

$$\begin{aligned}
L_{GWL} = & \sum_f (\bar{\Psi}_f (i\gamma^\mu \partial_\mu - m_f) \Psi_f - e Q_f \bar{\Psi}_f \gamma^\mu \Psi_f A_\mu) + \frac{g}{\sqrt{2}} \sum_i (\bar{a}_L^i \gamma^\mu b_L^i W_\mu^+ + \bar{b}_L^i \gamma^\mu a_L^i W_\mu^-) \\
& + \frac{g}{2x_w} \sum_f \bar{\Psi}_f \gamma^\mu (I_f^3 - 2s_w^2 Q_f - I6e_f \gamma_5) \Psi_f Z_\mu - \frac{1}{4} |\partial_\mu A_v - \partial_v A_\mu - ie(W_\mu^- W_v^+ - W_\mu^+ W_v^-)|^2 \\
& - \frac{1}{2} |\partial_\mu W_v^+ - \partial_v W_\mu^+ - ie(W_\mu^+ A_v - W_v^+ A_\mu) + ig' c_w (W_\mu^+ Z_v - W_v^+ Z_\mu)|^2 \\
& - \frac{1}{4} |\partial_\mu Z_v - \partial_v Z_\mu + ig' c_w (W_\mu^- W_v^+ - W_\mu^+ W_v^-)|^2 - \frac{1}{2} M_\eta^2 \eta^2 - \frac{g M_\eta^2}{8 M_W} \eta^3 - \frac{g'^2 M_\eta^2}{32 M_W} \eta^4 \\
& + |M_W W_\mu^+ + \frac{g}{2} \eta W_\mu^+|^2 + \frac{1}{2} |\partial_\mu \eta + i M_Z Z_\mu + \frac{ig}{2c_w} \eta Z_\mu|^2 - \sum_f \frac{gm_f}{2M_W} \bar{\Psi}_f \Psi_f \eta. \quad (1.1)
\end{aligned}$$

The standard model shown in equation 1.1, is a quantum field theory that describes the fundamental particles and how they interact. While this document does not require or attempt a detailed understanding the intricate detail of the standard model; the aim of many particle physics experiments is to verify, measure and expand the model. Despite being the current best theory to explain particle interactions, the model is not complete. There are many undescribed phenomena, such as the matter domination in the universe, that require physics beyond the standard model in order to be described. To that end, major international efforts namely in the form of the Large Hadron Collider, aim to gain further knowledge and understanding of the underlying physics of the universe. [1]

1.2 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) are silicon based integrated circuit chips. Unlike conventional chips, where the circuit and logic gates are permanently synthesised from silicon transistors at manufacture time, the internal structure of an FPGA can be manipulated to the desired structure of the user. The advantage of FPGAs is the high data transfer rates and versatility they deliver, without the cost of manufacturing purpose built chips. As such, FPGA's are used expensively in the development of new technology and small batch size production. [2]

FPGAs are programmed in a variety of languages known as Hardware Description Languages (HDLs). One of the more common HDLs and the HDL used in this project is Very High Speed Integrated Circuit Hardware Description Languages (VHDL). Programming a circuit in VHDL requires the creation of entities, sometimes referred to as modules. These entities can be thought of as a 'black box' that compute some form of logic on their inputs and outputs. The form of this logic computed but the entity is known as the entity architecture.

When building a circuit larger than one simple function it is often necessary to use more than one entity. In VHDL, entities can contain sub-entities within their architecture; entities that do this are known as top-level entities. Top level entities are often used to compute signals between their sub-entities and/or control how the sub-entities operate.

1.3 The LHCb Experiment

One experiment at the Large Hadron Collider is Large Hadron Collider beauty (LHCb). Located at intersection point 8, LHCb is designed to study rare particle physics phenomena, such as lepton flavour violation and CP violation. LHCb studies the decay modes of the B meson. These hadrons travel in the order of μm 's in the detector before decaying. As such, B meson decays can be identified by decay products that propagate from a secondary vertex.

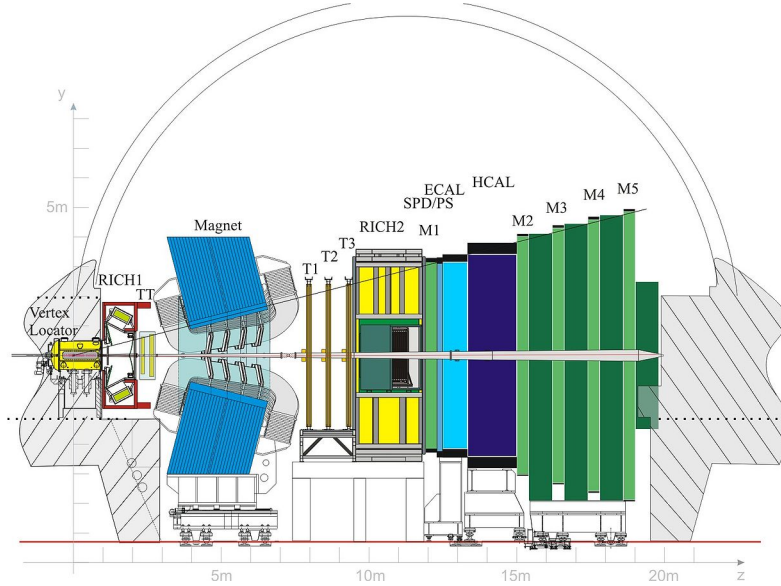


Figure 1.1: The LHCb Detector along the bending plane.

As B mesons are light (in comparison to other particles studied in the LHC), the decay products are produced at a shallow angle relative to the beam pipe; this is the driving factor in the design of the experiment. LHCb is a single arm forward spectrometer. Surrounding the point of collision is the Vertex Locator (VELO), this high precision detector uses silicon strips to detect ionising particles as they propagate from a collision and provides the coordinates of the particle in terms of R^1 and ϕ^2 . By reconstructing the paths of particles back to the intersection point, it can be identified whether or not the particular decay particles are a product of the primary vertex³, or a secondary vertex⁴.

The Rich detector, comprised of two subdetectors either side of the magnet, uses Cherenkov radiation to deduce the velocity of the particle. The silicon trackers, labeled TT

¹Radial distance from the beam pipe.

²Azimuthal angle from the beam pipe.

³The position at which the protons collided.

⁴The decay point of a short lived particle. i.e. B Meson.

and T1-3 in Figure 1.1, calculate the angle deflection by the magnet. By combining the velocity and angle of deflection, the mass, momentum and energy of the particles can be deduced from simple relativistic kinematics,

$$E^2 = M^2c^4 + p^2c^2. \quad (1.2)$$

The Muon detectors, labeled M1-5 in Figure 1.1, are important to detect muon's the detector. This is of particular importance on LHCb as muons can be easily misidentified as charged Pions, due to there similar mass. Pions are a common decay product of the interactions studied at LHCb, furthering the need to accurately identify Muon and Pions.

HCAL and ECAL, shown in Figure 1.1, are hadronic and electric calorimeters respectively. Both measure the total energy of incoming particles. As the calorimeters are absorb the particles they detect, any leptonic particle reaching the M2-5 muon detectors can be assumed to be a muon. Electrons and Photons are absorbed by the ECAL and any Tauons would have decayed long before reaching the far muon detectors.

1.4 LHCb Upgrade

With the advancements in accelerator technology, the detectors must also advance in order to make best use of the accelerators. The LHC is scheduling to increase its luminosity during Long Shutdown 2 (LS2), and as such LHCb will have to cope with this greater luminosity. The front end electronics of LHCb implement a hardware trigger and this is limited to a 1MHz maximum readout speed. Post LS2, LHCb will have to cope is a luminosity of $\mathcal{L} = 2.10^{33}cm^{-2}s^{-1}$, this is significatly greater than the current $\mathcal{L} = 4.10^{32}cm^{-2}s^{-1}$. A simple luminosity increase will not significantly increase that statistics for some statistical error dominated channels. To achieve greater statistical significance, greater resolution of the VELO and fully software triggers are required. Detailed in the '*LHCb VELO Upgrade Technical Design Report*' [3] the main goals of the 2019 upgrade are as follows:

- Increase the luminosity to $\mathcal{L} = 2.10^{33}cm^{-2}s^{-1}$.
- Read data from the detector at the bunch crossing frequency, 40 Mhz.
- Convert to a purely software based trigger.

1.4.1 VELO Upgrade

Common with its predecessor, the upgraded VELO uses thin, retractable modules. The advance of this approach is that during collisions, the modules can sit closer that otherwise possible to the beam line. The modules retract for the beam fill, avoiding the radiation damage from the wider fill beams. In order to gain greater resolution of secondary verticies, the upgraded VELO will sit at 5.1 mm from the beam at the closest pixel [3]. The current VELO achieves 8 mm [4].

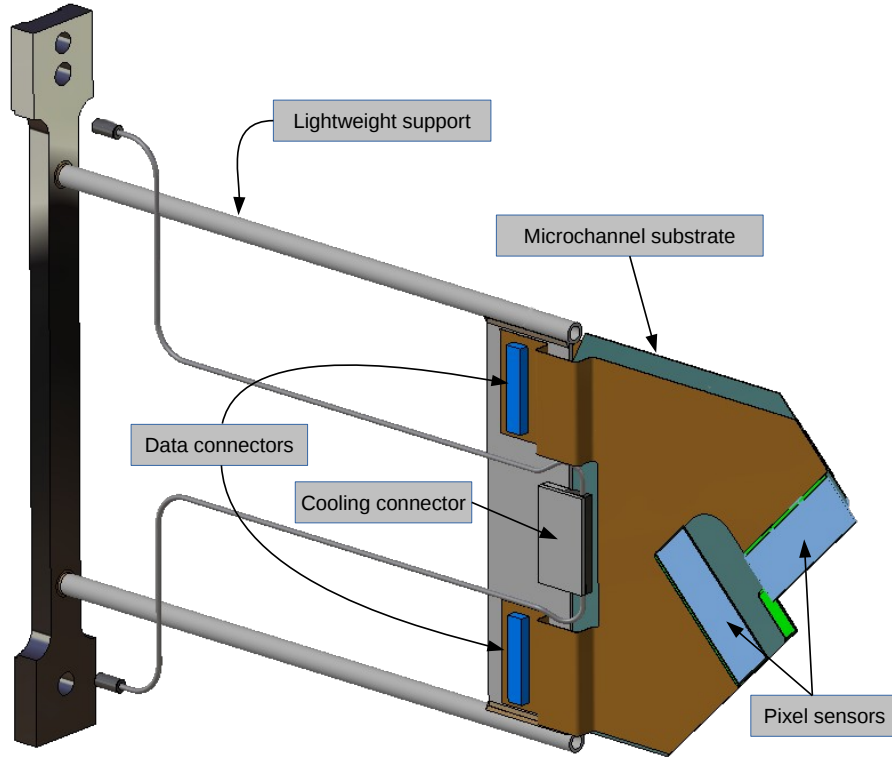


Figure 1.2: The current module design. Two sensors are shown, the remaining two are mounted to the rear face of the module to for two horizontal rows - covering the right most area of the module (as viewed in the figure).

As previously mentioned, the current VELO uses silicon strips to detect particles. The upgraded VELO, however, will use silicon pixels. These pixels, $55\mu m \times 55\mu m$ in size and $200\mu m$ thick [3], are arranged in a 256 wide square matrix on a ASIC chip. The pixels are arranged into groups of 8 to form a Super Pixel (SP). The ASIC chips are arranged in a row of 3 and bonded to a sensor. Each module has 4 sensors, 2 per side, as shown in Figure 1.2. The module is is cooled by bi-phase CO_2 in micro-channels etched into the microchannel substrate.

The VELO modules will operate in the LHC secondary vacuum. It is seperated from the primary vacuum by RF foil that is 3.5mm from the beamline, at the closest point [3]. The foil is made of $250\mu m$ thick aluminum to reduce the its interaction with the collision decay products.

1.4.2 Data Flow and Low Level Interface

FPGAs are used in the Data Aquisition (DAQ) modules for speed and paralell procesing capabilities. The DAQ, in its simplest form, is a series of optical links, a data processing FPGA and a PCIe port for data transfer to the VELO computer system.

The data from each SP is packaged in a 30 bit Super Pixel Packet (SPP). The SPP is comprised of a (from most to least significant bits) 9 bit Bunch Cross ID (BCID); 13 bit

146 SPP location information (horizontal and vertical coordinates); 8 bit SP hitmat.

148 A GWT serialiser forms a 128 bit ‘frame’ comprising of a header (1010), four single bit parity flags and four SPPs. The parity flags indicate the parity of the four SPPs as a validation check for downstream processes. The data then is transmitted via an electrical to optical converter though optical fibers to the DAQ. In the DAQ is the Low Level

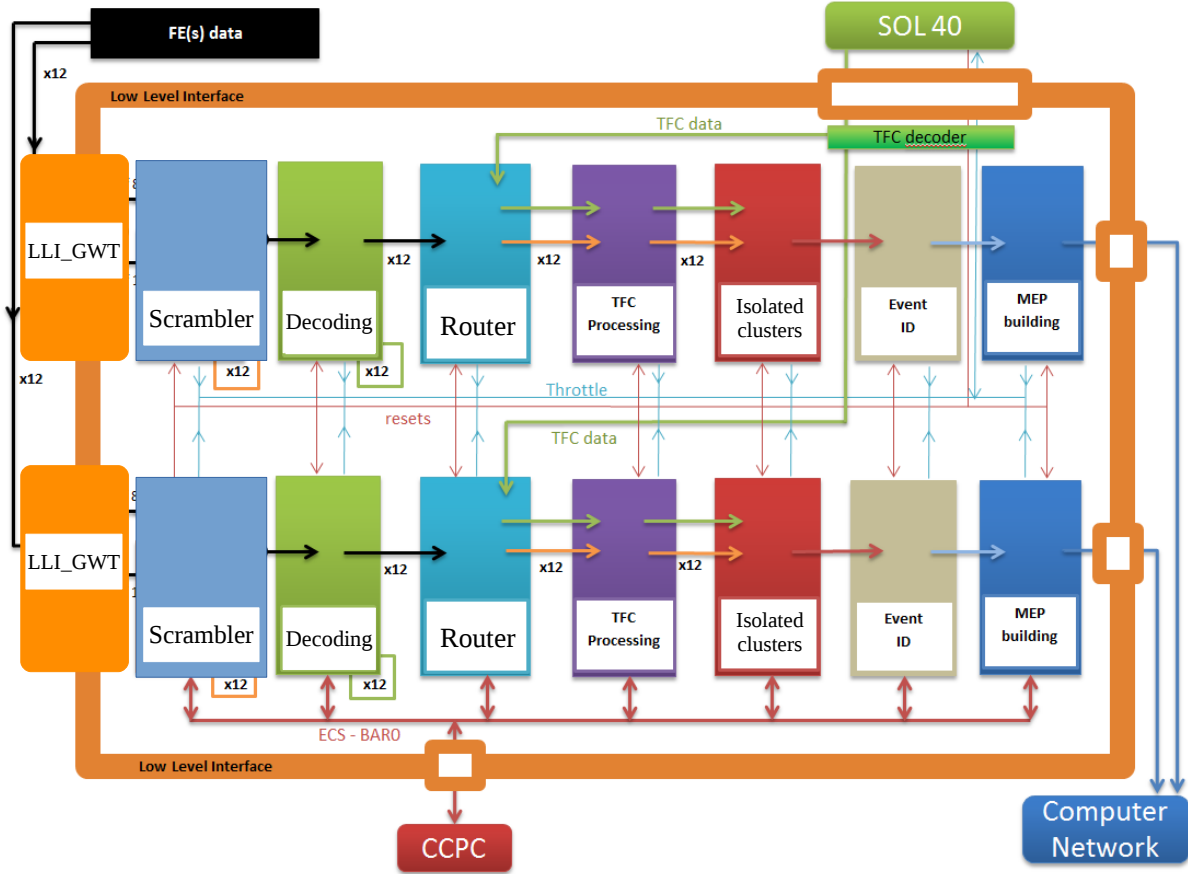


Figure 1.3: A Diagram showing the data flow in the low level interface.

150 Interface (LLI). The LLI is responsible for sorting the incomming data into time order and
 152 packaging the data in the correct form for the computer systems and to optimise output
 154 bandwidth. Other processes included in the LLI are descrambling and event isolation
 tagging. These processes will be discussed in more detail later in this document.

2 Scrambler

156 Due to radiation levels inside the detector chamber, the main data processing takes
 place in a concrete bunker away from the detector. To facilitate this, 20 optical links
 158 (per modual) are used to transfer the data from the front end VELO to the DAQ FPGA.
 When communicating data digitally, the transferring modual (TX) and the recieving modual
 160 (RX) must have synchronised clocks. In these case, the GWT serialiser is the TX, and the

DAQ is the RX. When achieving synchronised clock, there are two main approaches:

- Transmit the TX clock with the data to the RX modual - used in I²C and SPI communication.
- Use bit-changes in the data to continuously synchronise the RX clock.

The former of these options, although widely used in conventional electronics, requires a finely tuned clock accounting for all possible delays. The latter, while negating cons of the former, requires data with a high density of transitions to reduce the likelihood of a desynchronisation event. Because delays in the data are possible, the latter option has been selected.

As mentioned, it is necessary to ensure that the data has large density of transitions before being transmitted from the front-end detector to the DAQ modual. However, as the majority of super pixel hitmaps are empty, the data has a bias towards '0's. This reduces the frequency of transitions in the data - increasing the probability of a desynchronisation event. It is therefore necessary to scramble the data prior to transmission and descramble the data in the LLI of the DAQ FPGA.

Scrambling and later descrambling the data is not a trivial exercise. The scrambling (TX) modual and descrambling (RX) modual must use a synchronised 'key', that is used in both the scrambling and descrambling processes. In the FPGA, the 'key' is derived from the previous states of the data. There are two methods when generating this 'key':

Additive The 'key' is generated by evolving the previous 'key' at each iteration of data using the incoming frame.

Multiplicative The 'key' is generated from the previous n frames. (Here n is a variable specific to the algorithm).

2.1 Scrambler Options

Three scrambling algorithms have been considered:

Additive Scrambler

This scrambler is was originally implemented and used two sets of two-input XOR logic gates. As the name implies, this scrambler used additive key generation which is dependent all previous input frames since the last reset signal.

Intermediate Scrambler

Created by Karol Hennessy, and deriving its name arbitrarily from the order of consideration, this multiplicative scrambler combines the current and previous frames to generate the 'key'. Therefore, in the event of desynchronisation, only two frames are lost before the 'key' is automatically recovered. This feature alone is a significant improvement over the Additive Scrambler.

196 VeloPix Scrambler

198 This is the current implemented scramble algorithm in the DAQ and VeloPix code.
200 Like the Intermediate Scrambler, it uses multiplicative ‘*key*’ generation. However,
202 the VeloPix scrambler is compatible with further constraints enforced by the ASIC,
including the number of combinational logic operations. The Intermediate Scram-
bler was design purely for simulation purposes and as such does not meet these
constraints.

2.2 Cross Checks

204 The main priority when scrambling data, is ensuring that the data is recoverable. For
all three scramblers, the algorithm was synthesised in Quartus [5] and simulated in
206 Modelsim [6]. The aim of synthesising and simulating the scramblers in these programs
was to ensure that the design was both physical in term of on-board logic gates, and to
208 check that the scrambled data was recoverable, respectively.

Furthermore, a C++ simulation was created for the three scramblers. This simulation had
210 two main purposes: firstly to cross check the output of the C++ against the Modelsim
simulations; secondly to simulate the scrambler over a much larger sample of data as
212 Modelsim simulations are less time effecient. In attition to the cross checks, the C++
code allowed for the injection of a descronisation event, in which the ‘*key*’ is lost. As
214 expected, the Additive Scarmbler was unable to recover any data post descronisation,
however the intermediate and VeloPix scarmblers both recovered the ‘*key*’ after two
216 frames and continioud to recover data.

2.3 Algorithm Analysis

218 For analytical purposes, it is assumed that fully scrambled data is indistinguishable from
randomly generated data. For this reason, the three algorithm are not only tested against
220 eachother and the pre-scrambled QWT data but also randomly generated binary. The
randomly generated data was created using the Python ‘*random*’ library, selecting a ‘0’
222 or ‘1’ with equal probubility. While the Python ‘*random*’ library is only sudo-random,
on the scale of this example (i.e. $\gg 100,000$ frames), it is by far sufficient.

224 A more mathematically rigorous approuch, however, is to evaluate the system abstractly
in the framework of statistical physics. In this abstraction, the 120 bit frame (with the
226 header and parity removed) is considered an ensemble; microstates are the particular
form of the frames; and macroscopic quantities can be calculated by averaging a large
228 number of frames. For the analysis outlined in section 2.3.1, predictions will be made
using these principles and outlined in section 2.3.2.

230 In the context of the statisical model, it is reasonable to concider the degree of ‘*scrambled-*
ness’ analogous to entropy. This analogy is not disimilar to the common interpritation
232 of entropy as a measure of dissorder. From Bolzman’s equation for entropy,

$$S \sim \ln(\Omega) \quad (2.1)$$

where Ω is the number of microstates associated with the macrostate, we learn that this state of maximum entropy is a macrostate with the maximum number of associated microstates.

The entropic argument of Equation 2.1 is not only mathematical founded. For a scramble algorithm to hold for all possible data sets, it must also be capable of outputting all possible permutations. As such, assuming all possible output are equally likely, the count of each macroscopic output will be proportional to the number of microstates associated.

2.3.1 Measurements of the Algorithms

To compare the efficiency of the three algorithms in section 2.1, the algorithms were run over the same input data and compared for the following measures:

Number of Transitions Per Frame

This measure counts the total number of bit transitions (i.e. $bit(n) \neq bit(n-1)$) in a 120 bit frame. The header and parity information was not included as they are not scrambled. This is an important test as one of the roles of the scrambler is to maximise the number of transitions.

Common Bit Chain Length

One of the downfalls of the ‘Number of Transitions Per Frame’ analysis is that the two hypothetical 20 bit frames,

a) 10101010101111111111,

b) 10011001100110011001,

both with 10 transitions, are considered equal. However, (b) is clearly a more suitable output for data transfer as (a) has a large probability of desynchronised due to the long chains of ‘1’s in the right most bits. It is therefore also necessary to evaluate the length of common bit chains within the scrambled data as shorter chains are more suitable for data transfer.

Bit Asymetry

Pre-scramble, the data had a large bias towards ‘0’s due to the majority of the hitmaps being empty. Scrambled data, via entropic arguments, *should* show zero bias eitherway. Therefore, by investigating how the number of ‘1’s - ‘0’s evolves over many frames, any bias in the scrambler can be found.

2.3.2 Statistical Predictions

Number of Transitions Per Frame

Consider a particle in a symmetric, discrete time-dependent, two state system,

$$p_0(t) = p_1(t) = 0.5 \quad : \quad \forall t \in \mathbb{N}, \quad (2.2)$$

At each time iteration,

$$p_{i \rightarrow j}(t) = 0.5 \quad : \quad i, j = [0 \ 1], \quad \forall t \in \mathbb{N}. \quad (2.3)$$

However, assuming zero bias and detailed balance, as $p_{1 \rightarrow 0}(t)$ is equal in both probability and importance to $p_{0 \rightarrow 1}(t)$, the probability of a bit change shall hereafter be referred to as $p_\tau(t)$.

Over a n step process, analogous to a n bit frame, the probability distribution of the number of transitions N_τ is given by Binomial statistics,

$$f(N_\tau) = \frac{n!}{N_\tau!(n - N_\tau)!} p^{N_\tau} (1 - p)^{n - N_\tau} \quad (2.4)$$

Simplified for the special case $p = p_\tau = 0.5$,

$$f_\tau(N_\tau) = \frac{n!}{N_\tau!(n - N_\tau)!} (p_\tau)^n \quad (2.5)$$

For $n = 120$, we can calculate,

$$\langle N_\tau \rangle^{Binomial} = \sum_{N_\tau=0}^{n-1} N_\tau f(N_\tau) = n p_\tau = 60 \quad (2.6)$$

$$\sigma_{N_\tau}^{Binomial} = \sqrt{n p_\tau^2} = 5.48 \quad (2.7)$$

Furthermore, when considering the entropic argument of equation 2.1, the number of microstates corresponding to each macrostate N_τ can be related to equation 2.5,

$$\Omega_\tau \sim \frac{n!}{N_\tau!(n - N_\tau)!} \quad (2.8)$$

$$\langle N_\tau \rangle^{Entropic} = MAX[S_\tau] = MAX[\Omega_\tau] \quad (2.9)$$

This can be numerically solved,

$$\langle N_\tau \rangle^{Entropic} = 60 \quad (2.10)$$

While the result of equation 2.10 does not contribute anything new, it is important as a ‘*sanity check*’. Because the system can be described as in section 2.3, it would indicated a problem in the theoretical framework if the result did not match.

Common Bit Chain Length

The probability of a chain of length n is,

$$p_n = p_1(1 - p_\tau)^{n-1}, \quad : \quad n \in \mathbb{N}, \quad n > 1 \quad (2.11)$$

where p_1 is the number of chains of length 1. As $p_1 = N_0(1 - p_\tau)$, where N_0 is the total number of chains,

$$\frac{N_n}{N_0} = (1 - p_\tau)^n, \quad : \quad n \in \mathbb{N}, \quad n > 1 \quad (2.12)$$

where N_n is the number of chains of length n . Taking the log of both sides,

$$\begin{aligned} \log\left(\frac{N_n}{N_0}\right) &= n \log(1 - p_\tau), \\ \log(N_n) &= n \log(1 - p_\tau) + \log(N_0). \end{aligned} \quad (2.13)$$

Therefor, for a graph of $\log(N_n)$ against n for a large sample of data, the gradient would be $\log(1 - p_\tau)$. In this case, as $p_\tau = 0.5$,

$$\log(1 - p_\tau) = -0.30. \quad (2.14)$$

Bit Asymetry

$A_{1,0}$, the assymetry of ‘1’s and ‘0’s is defined as,

$$A_{1,0} = N_1 - N_0, \quad (2.15)$$

where N_1 and N_0 are the number of ‘1’s and ‘0’s respectively. We can consider the evolution of $A_{1,0}$ with frame t of size n as a stockastic iterative map with zero deterministic growth [7],

$$A_{1,0}(nt + n \Delta t) = A_{1,0}(nt) + \mathcal{N}(nt) \quad (2.16)$$

Where \mathcal{N} is an independant random variable picked from a gaussian distribution. While $A_{1,0}(t) \in \mathbb{Z}$, in the limit of large nt we can approximate that $A_{1,0}$ is continious.

If we consider the moments of $A_{1,0}$,

$$\langle A_{1,0}(nt = M n \Delta t) \rangle = \sum_{m=0}^{M-1} \mathcal{N}(m n \Delta t), \quad (2.17)$$

$$\begin{aligned} \langle A_{1,0}(nt = M n \Delta t)^2 \rangle &= \sum_{m=0}^{M-1} \sum_{m'=0}^{M-1} \mathcal{N}(m n \Delta t) \mathcal{N}(m' n \Delta t) \delta_{mm'} \\ &= \sum_{m=0}^{M-1} \langle \mathcal{N}(m n \Delta t)^2 \rangle. \end{aligned} \quad (2.18)$$

Clearly, in Equation 2.17, $\langle A_{1,0} \rangle = 0$. In Equation 2.18, we assume the variance is of form $(n \Delta t)^\alpha$ [7]. Then,

$$\langle A_{1,0}(nt = M n \Delta t)^2 \rangle = M(n \Delta t)^\alpha. \quad (2.19)$$

Running the analysis over the frames $t = 0$ to t_f , the number of bits sampled is $M = t_f/n \Delta t$. Substituting this into Equation 2.19,

$$\langle A_{1,0}(nt = M n \Delta t)^2 \rangle = t_f (n \Delta t)^{\alpha-1}. \quad (2.20)$$

Considering the three cases of α in the approximation of continuous $n\Delta t$:

- $\alpha > 1$: Here $A_{1,0} \rightarrow 0$ as $\Delta t \rightarrow 0$.
- $\alpha < 1$: Here $A_{1,0} \rightarrow \infty$ as $\Delta t \rightarrow 0$.
- $\alpha = 1$: This is the only sensible choice.

With $\alpha = 1$,

$$\langle A_{1,0}(nt = M n \Delta t)^2 \rangle = M(n \Delta t). \quad (2.21)$$

And thus,

$$\sigma_{A_{1,0}} = \sqrt{\langle A_{1,0}^2 \rangle - \langle A_{1,0} \rangle^2} = \sqrt{\langle A_{1,0}^2 \rangle} = \sqrt{n \Delta t}. \quad (2.22)$$

2.3.3 Results of Analysis

The results from the ‘*Number of Transitions Per Frame*’ analysis, shown in Figure 2.1, show a strong similarity between the Intermediate and VeloPix Scramblers with the randomly generated data. These results are within 1% agreement with the theoretical predictions for $\langle N_\tau \rangle = 60$ and $\sigma_{N_\tau} = 5.48$, made in Section 2.3.2. The remarkable consistency between the theoretical predictions and the randomly generated data provides confidence in both the theory, and the scrambled nature of the Intermediate and VeloPix scrambler outputs.

Figure 2.1: Results of the ‘*Number of Transitions Per Frame*’ analysis. The results for the Random Data, Intermediate Scrambler and VeloPix Scrambler overlap for the ‘*Number of Transitions Per Frame*’ analysis.

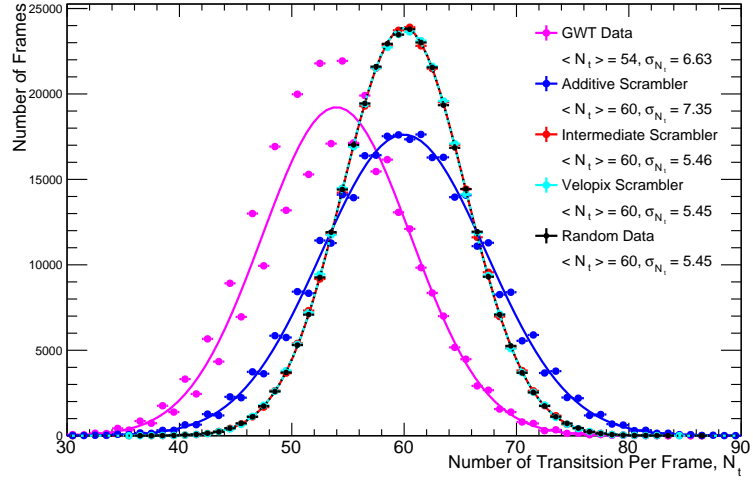
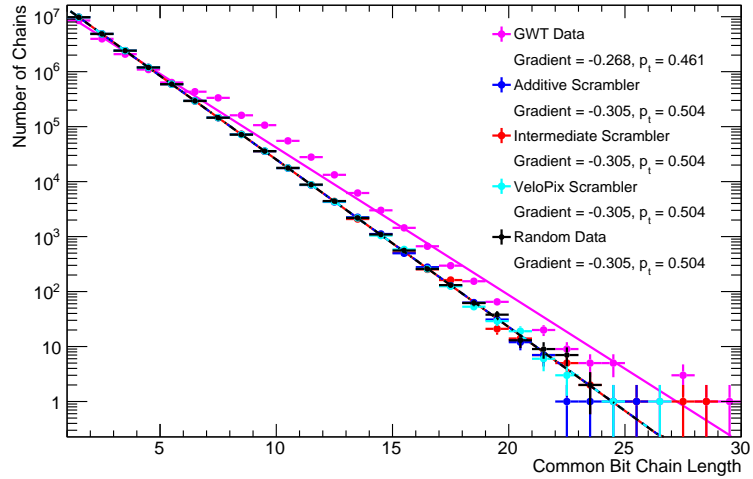


Figure 2.2: Results of the ‘*Common Bit Chain Length*’ analysis. The results for the Random Data, Additive Scrambler, Intermediate Scrambler and VeloPix Scrambler approximately overlap for the ‘*Common Bit Chain Length*’ analysis.



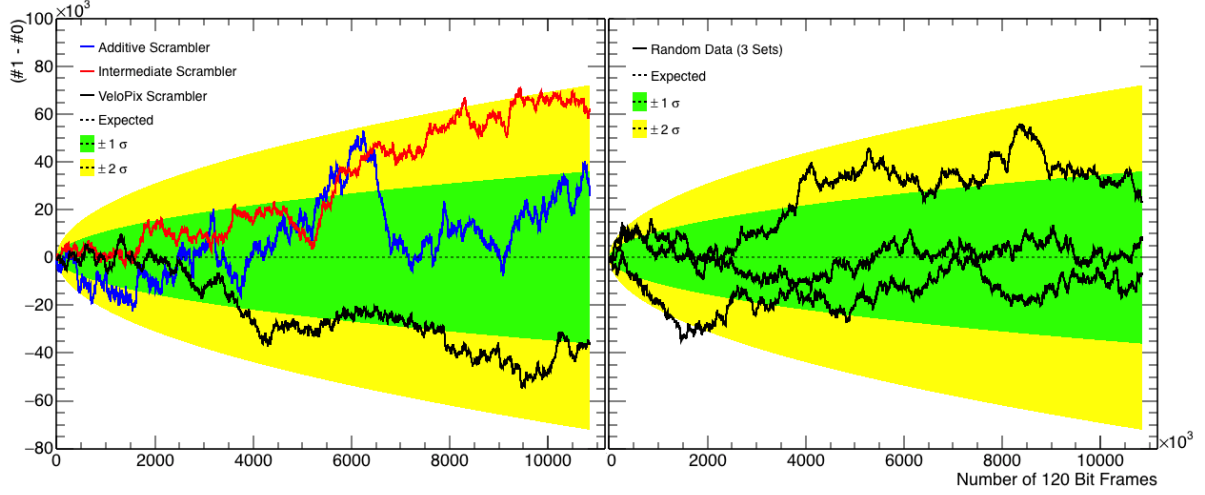


Figure 2.3: The results of the ‘*Bit Asymmetry*’ analysis.

316 All three scramblers, the random data, and the theoretical predictions are consistant to
 318 within 1%. Comparing the two results for the Additive Scrambler, however, its shown
 320 that while the frequency of longer chains is consistant with random data - the variance
 of transitions is larger than predicted, and thus the long and short trains are more locally
 clustered.

322 The ‘*Bit Asymmetry*’ of each scrambler, shown in Figure 2.3, is consistant with the the-
 324 oretical prediction. The deviation of $A_{1,0}$ for the predicted mean of 0 is fully consistant
 with stockastic noise. The random data also shows consistancy. This gives confidance in
 the assumtpions made in Section 2.3.2.

2.4 Conclusion

	$\langle N_\tau \rangle$	σ_{N_τ}	Gradient	p_τ
GQT data	54	6.63	-0.268	0.460
Additive Scrambler	60	7.35	-0.305	0.504
Intermediate Scrambler	60	5.45	-0.305	0.504
Velopix Scrambler	60	5.46	-0.305	0.504
Random Data	60	5.45	-0.305	0.504
Theoretical Prediction	60	5.48	-0.3	0.5

Table 2.1: The combined results of the algorithm analysis.

326 The consistancy of random data and the theoretical predictions justifies the assumptions
 and approximations made in Section 2.3 and Section 2.3.2. Furthermore, the conforma-

tion of the statistical model allows for accurate comparisons to be made from predicted values and their measured counterparts.

The Additive Scrambler, while consistent with the ‘*Chain Length*’ and ‘*Bit Asymmetry*’ analysis, has a variance in the transition frequency that leads to the conclusion that long and short chains are locally clustered. This is not ideal for data transfer. Many sequential long chains increase the probability of TX-RX clock desynchronisation. Furthermore, the additive scrambler will not recover from this loss of synchronisation, as the ‘*key*’ will never be recovered without a common reset signal.

The Intermediate Scrambler produced an output consistent with random data. This makes the algorithm suitable for data transfer. As already mentioned, however, the scrambler is designed for computer simulation. As such, it is not suitable for implementation as it does not meet the additional requirements of the ASIC.

The VeloPix Scrambler, like the Intermediate Scrambler, produces a statistically scrambled output. Furthermore, the algorithm is inline with the additional requirements of the ASIC. As such, it is ideal for implementation, and hence is currently the choice algorithm for use in the 2019 VELO upgrade.

steam

3 Event Isolation Flagging

One challenge of increasing the readout speed of the detector is processing the data produced. Because of this, any pre-computer data processing possible reduces the load and processing time of the computer system. One area where the DAQ’s FPGA will be used for this purpose is in Event Isolation Flagging (EIF).

Particles traversing the VELO have a probability that they will pass through the boundary of two or more Super Pixels. This will cause multiple SPPs for the same particle. As such, the reconstruction of the particle’s path is a more complicated process than a particle path that only intersects with one SP.

The aim of EIF is to identify the SPPs that completely describe the particle’s interaction with the module and flag the event as isolated. These flagged events will allow the computer systems to prioritise these easier to re-construct paths. This reduces event pile up on the computer network.

3.1 Time Sorting Data

Frames arriving at the DAQ from the GWT are not time ordered. When fully implemented, the LLI will have time sorted the data before the EIF. However, the provided simulated data of the VELO is not time ordered.

362 In order to test any EIF development, it is necessary to time order the simulation data.
This was done using a python script that sorted the SPPs into lists according to BCID.
364 The script has three main phases:

- Read in SPP and retrieve BCID.
- 366 • Add SPP to correct list according to BCID.
- Print list of opposite BCID (i.e. input SPP's BCID + 124 accounting for BCID 256
368 rolling over to 0) to file.

As not all BCID's are present, measures were put in place to ensure all BCID lists were
370 outputted in time order, preventing list containing two or more bunch crosses. The time
order of the data was tested and confirmed as correct.

372 One advantage of this process is that, regardless of the number of isolated events, the
data no longer needs to be sorted by the computer network. This further reduced the
374 computational load on the computers.

3.2 Bubble Sorting

376 The first step in EIF is to sort all the SPP's that correspond to the same bunch crossing
(Hereafter referred to as a '*data train*') by their row.

378 Bubble sorting, when implemented in series processing, is a relatively slow sorting al-
gorithm. At worst case, Bubble sorting requires n^2 iterations to complete the sort.
380 However, as FPGA's can easily parallel process. By making $\frac{n}{2}$ comparisons at a simul-
taneously (even-odd or odd-even), FPGA Bubble Sorting in the worst case scenario only
382 requires n iterations. This is made clear in Figure 3.1.

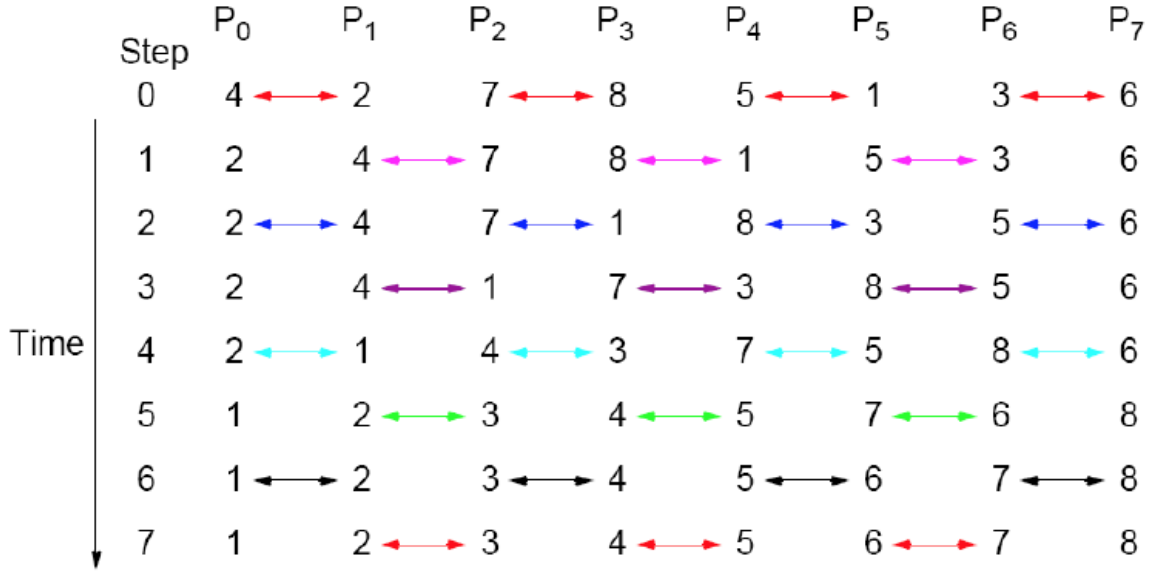


Figure 3.1: A diagram showing Bubble Sorting in an FPGA.

3.3 Isotaton Checking

Once the data train is sorted by row, each SPP in the train can be compaired against its adjesent SPP's. If the SPP is seperated by > 1 row to both adjesent SPP's, the event is isolated. The SPP is then stored as a 31 bit SPP, with the new bit added as the least significant bit (shifting the orrigional SPP 1 bit in significance), with the new bit signaling 1 for isolation and 0 for non-isolated.

3.4 Data Train Overflow

One limitation of EIF in an FPGA is the limitation on resources. The logic systems are static in design and as such there is a natural need for a cap on the size of datatrain that the EIF system can accept - specifically for the bubble sorting. Because of this limitation, the EIF system is required to implement a overflow sytem that will reget data trains above a pre-determined limit, and move them to the next step of the LLI without proccessing them. This system is also required to bypass data if a data train arrives at the EIF system before the previous train has been processed - preventing pill up.

In order to investigate the limit needed for the overflow, the distribution of data train sizes was investigated. For each ASIC, a graph similar to those in Figure 3.2 can be created.

More important, however, is the faction of data trains over the bypass limit. For four theoretical limits, the fraction of overflow datatrains was calculated from the VELO simulated data, and it shown in Figure 3.3. This analysis, however, raises questions

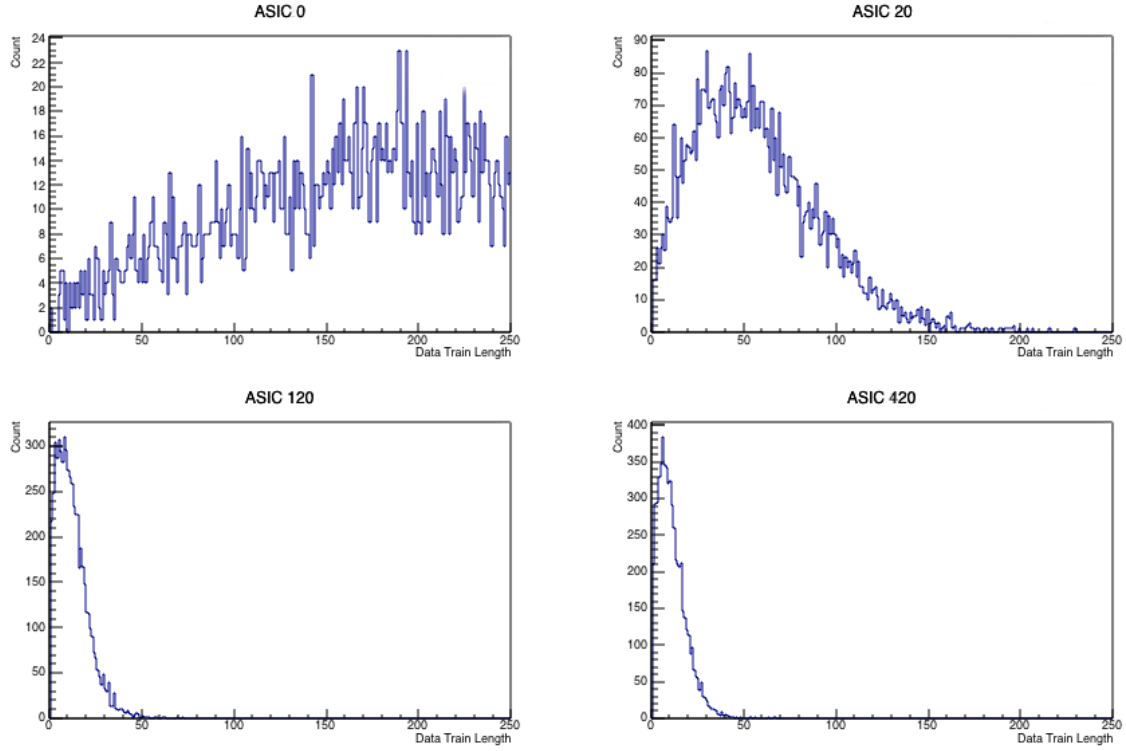


Figure 3.2: The data train length distribution of 4 ASIC chips.

behind that of the overflow limit. The ASIC's below 100 show no similarity to those above 100.

Further investigations as the structure of the simulated data is shown in Appendix A.1. From this we learn the large variance in the data (ASIC number ≥ 100) is due to the ASIC's position on the module this is expected. This structure is not consistent across the ASIC's pre and post 100. It can be concluded that the simulated data contains a 'bug' and it is now being reviewed by the creators of the simulation. No further analysis can be continued on this front until this 'bug' has been properly investigated.

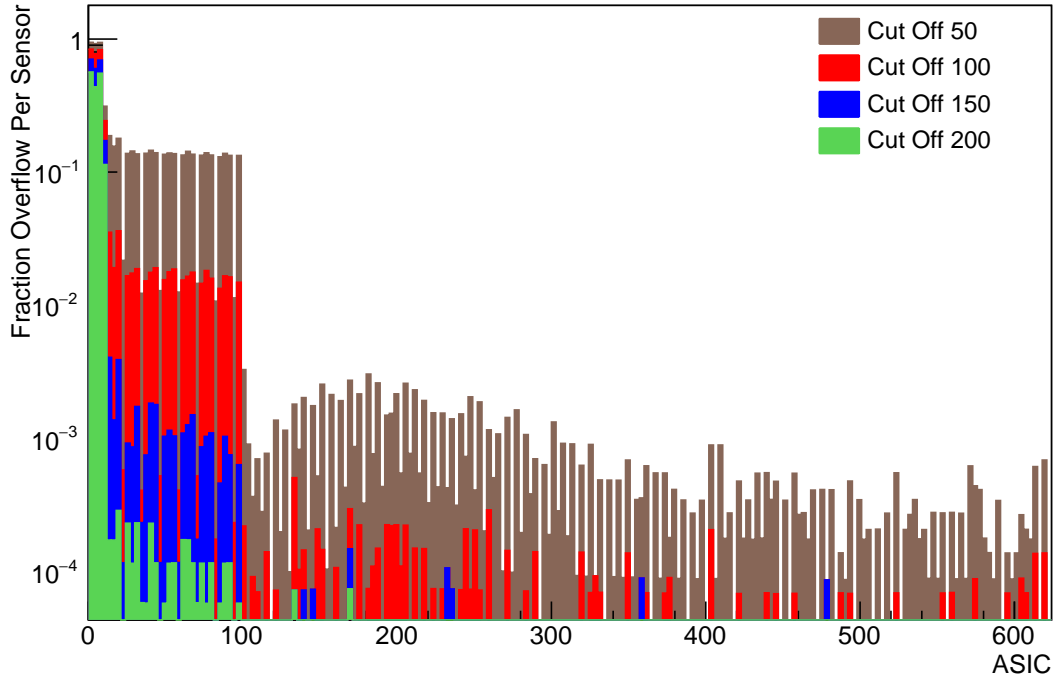


Figure 3.3: fraction of overflow data trains for four overflow limits.

3.5 Current Stage of Development

The EIF system is still currently in active VHDL development. The current development code is still in a stand alone format and not intergrated with the master LLI code. Currently created, and ready for stand alone testing, is a bubble sorting module with data in and out systems. The module consistence of a top level control entity and a comparison/swap sorting entity. The control entity forms a feedback loop passing the ouput of the sorting entity back into its input at each step. At each step, the parity of comparison is changed (i.e. odd-even to even-odd).

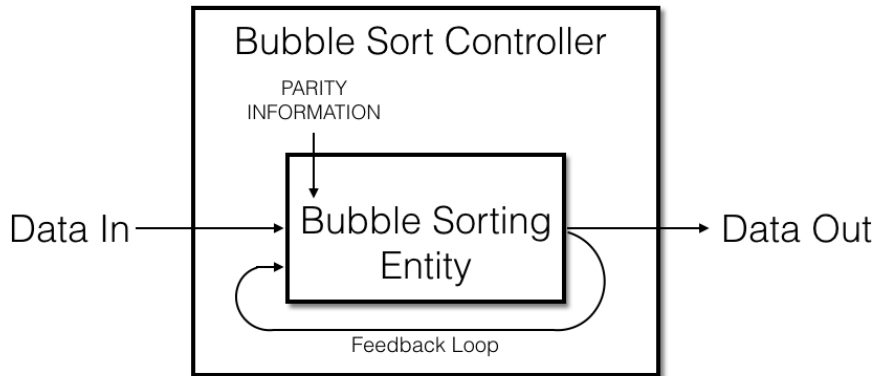


Figure 3.4: Data from for the developmental bubble sorting module.

420 This process continues intill the input and output of the sorting module is identical for
two subsequent steps. At this point the data is sorted and passed to the output. The
422 data flow is more simply demonsttraited in Figure 3.4.

Once testing a bug fixing is complete, the EIF will be expanded to include Isolation
424 Flagging and an overflow as discussed. Once the stand alone system is complete, it will
be intergrated into the LLI master code and modified to comply with theh LLI data
426 managment systems.

References

- 428 [1] Cern. *The Standard Model*. 2015. URL: <http://home.cern/about/physics/standard-model> (visited on 12/2015).
- 430 [2] “Toward FPGA-Enabled Scientific Computing”. In: *Design Test of Computers, IEEE* 28.4 (July 2011), pp. 4–4. ISSN: 0740-7475. DOI: 10.1109/MDT.2011.91.
- 432 [3] LHCb Collaboration. *LHCb VELO Upgrade Technical Design Report*. Tech. rep. CERN-LHCC-2013-021. LHCb-TDR-013. Geneva: CERN, Nov. 2013. URL: <https://cds.cern.ch/record/1624070>.
- 434 [4] CERN. *LHCb VELO Project*. 2015. URL: <http://lhcb-vd.web.cern.ch/lhcb-vd/html/project.htm>.
- 436 [5] Altera. *Quartus Prime Software*. 2015. URL: <https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html> (visited on 12/2015).
- 438 [6] Mentor Graphics. *ModelSim - Leading Simulation and Debugging*. 2015. URL: <https://www.mentor.com/products/fpga/model/> (visited on 12/2015).
- 440 [7] Kurt Jacobs. *Stochastic Processes for Physicists - Understanding Noisy Systems*. Cambridge University Press, 2010. ISBN: 9780521765428.
- 442

A.1 Event Isolation Flagging

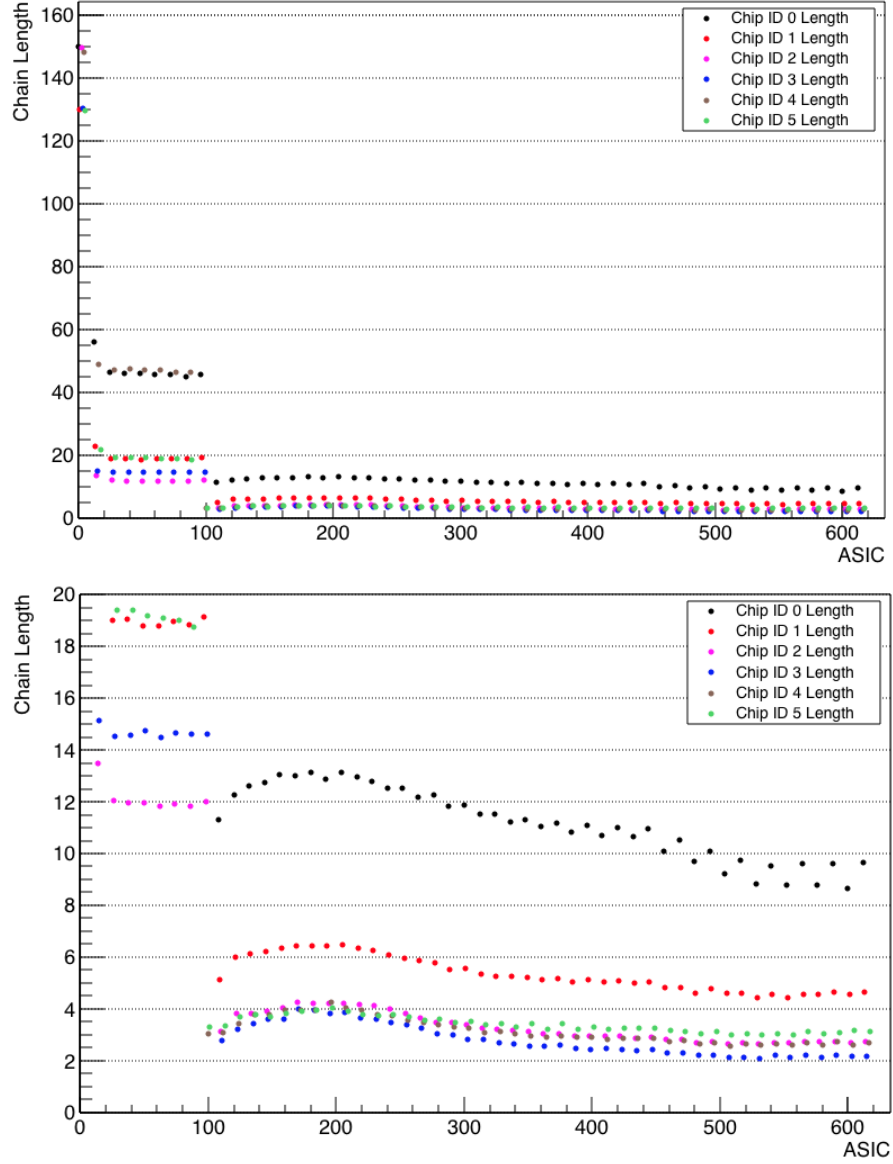


Figure A.1: The mean data train length for each ASIC, coloured by the chip number.