

FPGA Development for the LHCb Vertex Locator Upgrade

Nicholas Mead
8064141

School of Physics and Astronomy
University of Manchester

December 19, 2015

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur blandit purus ut lacus aliquam, a sodales ante sodales. Etiam a elit nunc. Mauris ipsum tellus, ullamcorper et arcu at, cursus malesuada elit. In tempus pellentesque nisi, vel egestas enim cursus tempus. Sed velit urna, luctus sed efficitur sed, laoreet vitae magna. Mauris elementum dignissim lacus vitae tempus. Curabitur laoreet molestie dictum. Donec sit amet auctor nisl.

Duis pellentesque euismod pellentesque. Praesent volutpat tincidunt eros, at faucibus tellus eleifend a. Quisque molestie sed ante sit amet sodales. Duis sed justo quam. Curabitur tellus felis, laoreet et bibendum a, posuere eget nisi. Donec suscipit lacinia porttitor. Aenean posuere sem nibh, et iaculis nisl faucibus eu. Donec ac posuere sapien. Aenean suscipit, nisi eget porttitor viverra, dui sapien vulputate lectus, ut dapibus purus orci nec arcu. Etiam placerat sapien non massa fringilla, et malesuada nibh hendrerit. Vestibulum et porttitor mi. Aliquam turpis velit, rutrum vitae erat at, scelerisque cursus lacus. Praesent libero urna, sodales efficitur eros id, sodales lacinia sem. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Contents

24	1 Scrambler	1
	1.1 The Role of Scrambling Data in the VELO	1
26	1.2 Scrambler Options	1
	1.3 Cross Checks	2
28	1.4 Algorithm Analysis	2
	1.4.1 Measurements of the Algorithms	3
30	1.4.2 Statistical Predictions	3
	1.4.3 Results of Analysis	7
32	1.5 Conclusion	8
	References	10

1 Scrambler

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 (per modual) are used to transfer the data from the front end VELO to the Data Aquizition FPGA (DAQ). When communicating data digitaly, the transferring modual (TX) and the recieving modual (RX) must have syncrinised clocks. In these case, the (name form dataflow) is the TX, and the DAQ is teh RX. When achieving synchronised close, there are two main approuches:

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

The former of these options, although the more convienient, is not appropriate for the VELO as it is suseptable to unforseens delays that could cause desynchronisation of the clocks to the data. The latter, while more invariant during delays, requires data with a high density of tranitions to reduce the likelyhood of a desynchronisation event. Becuase delays in the data are possible, the latter option has been selected.

1.1 The Role of Scrambling Data in the VELO

For the reasons described in Section 1, it is nessesary 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 bais towards '0's. This reduces the frequancy of transitions in the data - increasing the probability of a desynchronisation event. It is therefor nessecary to scramble the data prior to transmtion and descramble the data in 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 sycronised '*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 itteration of data using the incoming frame.

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

1.2 Scrambler Options

Three scrambling algorithms have been concidered:

66 this section is out of date, but I dont have access to the uptoday section untill
after I sent this. I have just not included it for now.

68 1.3 Cross Checks

Ofcourse, the main prioritys when scrambling data is ensuring that the data is recover-
70 able. For all three scramblers, the algorithm was synthesised in Quartus[1] and sim-
ulated in Modelsim[2]. The aim of synthesising and simulating the scramblers in these
72 programs was to ensure that the design was physical in term of on-board logic gates, and
to check that the scrambled data was recoverable.

74 Furthermore, a C++ simulation was created for the three scramblers. This simulated
had two purposes: firstly the output of the C++ can checked against the Modelsim sim-
76 ulated cehck consistancy; secondly to simulate the scrambler over a much larger simple of
data as Modelsim simulations are less time effecient. In attition to the cross checks, the
78 C++ code allowed for the injection of a desynchronisation event. As expected, the additive
scarmbler was unable to recover any data post-desynchronisation, however the Intermedi-
80 ate and VeloPix scramblers both recovered the ‘key’ after two frames and returned to
descrambling data.

82 1.4 Algorithm Analysis

One assumption made is that fully scrambled data will be indistinguishable from randomly
84 generated data. For this reason, the three algorithm are not only tested against eachother
and the pre-scrambled data but also randomly generated binary. The randomly generated
86 data was created using the Python ‘random’ library, selecting a ‘0’ or ‘1’ with equal
probubility. While the Python ‘random’ library is only sudo-random, on the scale of this
88 example (i.e. > 100,000 frames), it is by far sufficient.

A more mathematically rigorous aprouch, however, is to evaluate the system abstractly
90 in the framework of statistical physics. In this abstraction, the 120 bit frame (with the
header and parity removed) is considered a ensemble; microstates are the particular form
92 of the frames; and macroscopic quantities can be calculated by averaging a large number
of frames (i.e. the desync data). For the analysis outlined in section 1.4.1, predictions
94 will be made using these principles and outlined in section 1.4.2.

In the context of the statisical model, it is reasonable to concider the degree of ‘scrambled-
96 ness’ analogous to entropy. This analogy is not disimilar to the common interpretation
of entropy as a measure od disorder. Therefor a scrambled system can be assumed to
98 one of maximum entropy; and from Boltzmans law,

$$S \sim \ln(\Omega) \tag{1.1}$$

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 1.1 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.

1.4.1 Measurements of the Algorithms

To compare the efficiency of the three algorithms in section 1.2, 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 Asymmetry

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 imbalance of ‘1’s and ‘0’s evolves over many frames, any bias in the scrambler can be found.

1.4.2 Statistical Predictions

Number of Transitions Per Frame

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

$$p_0(t) = p_1(t) = 0.5, \quad : \quad \forall t \in \mathbb{N} \quad (1.2)$$

at each time iteration,

$$p_{i \rightarrow j}(t) = 0.5. \quad : \quad i, j = [0 \ 1], \quad \forall t \in \mathbb{N} \quad (1.3)$$

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

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

$$f(N_t) = \frac{n!}{N_t!(n - N_t)!} p^{N_t} (1 - p)^{n - N_t} \quad (1.4)$$

Simplified for the special case $p = p_t = 0.5$,

$$f_t(N_t) = \frac{n!}{N_t!(n - N_t)!} (p_t)^n \quad (1.5)$$

140 For $n = 120$, we can calulate,

$$\langle N_t \rangle^{Binomial} = \sum_{N_t=0}^{n-1} N_t f(N_t) = n p_t = 60 \quad (1.6)$$

$$\sigma_{N_t}^{Binomial} = \sqrt{n p_t^2} = 5.48 \quad (1.7)$$

142 Furthermore, when considering the entropic argument in section 1.4 equation 1.1,
 the number of microstates corespoding to each macrostate N_t can be related to
 equation 1.5,

$$\Omega_t = \binom{n}{N_t} = \frac{n!}{N_t!(n - N_t)!} \quad (1.8)$$

$$\langle N_t \rangle^{Entropic} = MAX[S_t] = MAX[\Omega_t] \quad (1.9)$$

144 This can be numerically solved,

$$\langle N_t \rangle^{Entropic} = 60 \quad (1.10)$$

146 The result of equation 1.10 is consistant with Equation ???. This is an important
 as a ‘*sanity check*’ as the descrepincy would indicate an issue in the theory.

Common Bit Chain Length

148

The probability of a chain of length n is,

$$p_n = p_1(1 - p_t)^{n-1}, \quad : \quad n \in \mathbb{N}, \quad n > 1 \quad (1.11)$$

150

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

$$\frac{N_n}{N_0} = (1 - p_t)^n, \quad : \quad n \in \mathbb{N}, \quad n > 1 \quad (1.12)$$

152

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

$$\begin{aligned} \log\left(\frac{N_n}{N_0}\right) &= n \log(1 - p_t), \\ \log(N_n) &= n \log(1 - p_t) + \log(N_0). \end{aligned} \quad (1.13)$$

154

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

$$\log(1 - p_t) = -0.30. \quad (1.14)$$

Bit Asymetry

156

$A_{1,0}$, the measure of assymetry of '1's and '0's is defined as,

$$A_{1,0} = N_1 - N_0, \quad (1.15)$$

158

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 [3],

160

$$A_{1,0}(nt + n \Delta t) = A_{1,0}(nt) + \mathcal{N}(nt) \quad (1.16)$$

162

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 $A_{1,0}$,

$$\langle A_{1,0}(nt = M n \Delta t) \rangle = \sum_{m=0}^{M-1} \mathcal{N}(m n \Delta t), \quad (1.17)$$

$$\begin{aligned} \langle A_{1,0}(nt = M n \Delta t)^2 \rangle &= \sum_{m=0}^{M-1} \sum_{m'=0}^{M-1} \mathcal{N}(m n \Delta t) \mathcal{N}(m' n \Delta t) \delta_{mm'} \\ &= \sum_{m=0}^{M-1} \langle \mathcal{N}(m n \Delta t)^2 \rangle. \end{aligned} \quad (1.18)$$

164 Clearly, in Equation 1.17, $\langle A \rangle = 0$. In Equation 1.18, we assume the variance is of form $(\Delta t)^\alpha$ [3]. Then,

$$\langle A_{1,0}(nt = M \Delta t)^2 \rangle = M(n \Delta t)^\alpha. \quad (1.19)$$

166 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 1.19,

$$\langle A_{1,0}(nt = M n \Delta t)^2 \rangle = t_f (n \Delta t)^{\alpha-1}. \quad (1.20)$$

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

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

172 With $\alpha = 1$,

$$\langle A_{1,0}(nt = M n \Delta t)^2 \rangle = M(n \Delta t). \quad (1.21)$$

And thus,

$$\sigma_{A_{1,0}} = \sqrt{\langle A_{1,0}^2 \rangle - \langle A_{1,0} \rangle^2} = \sqrt{\langle A_{1,0}^2 \rangle} = \sqrt{n \Delta t}. \quad (1.22)$$

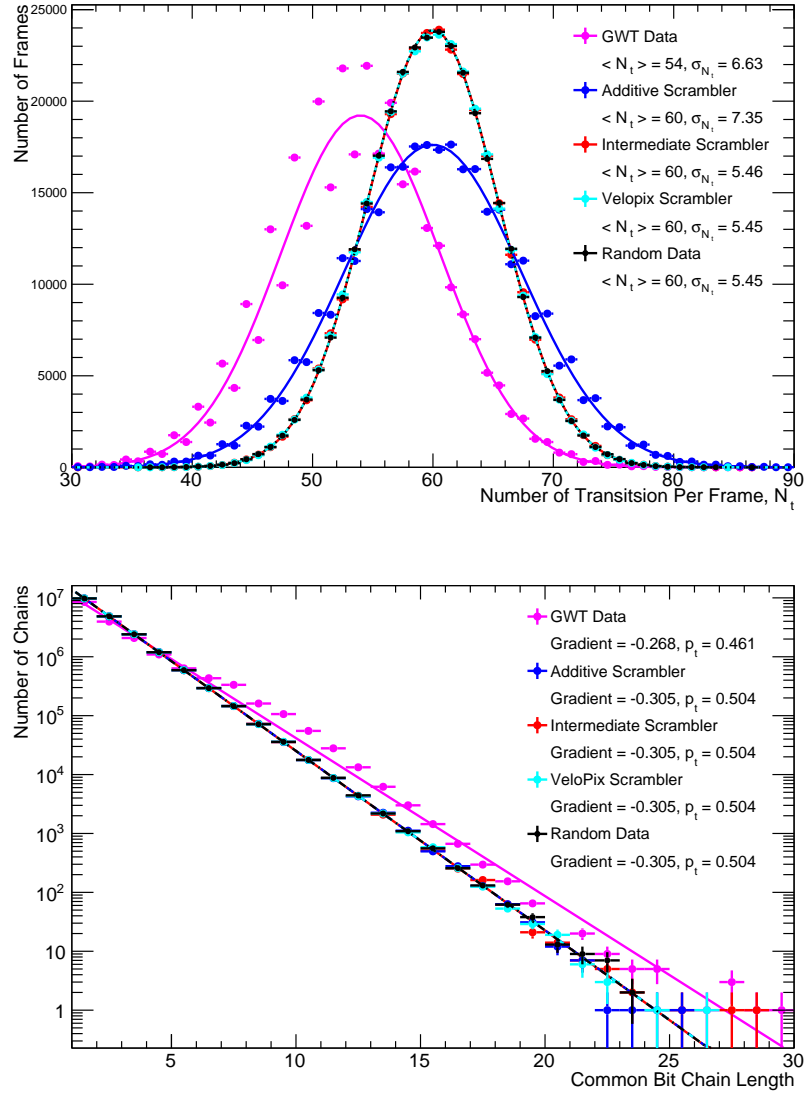


Figure 1.1: Results of the ‘*Number of Transitions Per Frame*’ analysis (Top) and the ‘*Common Bit Chain Length*’ analysis (Bottom). The results for the Random Data, Intermediate Scrambler and VeloPix Scrambler overlap for the ‘*Number of Transitions Per Frame*’ analysis. The results for the Random Data, Additive Scrambler, Intermediate Scrambler and VeloPix Scrambler approximately overlap for the ‘*Common Bit Chain Length*’ analysis.

The results from the ‘*Number of Transitions Per Frame*’ analysis, shown in Figure 1.1, show a strong correlation between the Intermediate and VeloPix Scramblers with the randomly generated data. These results are within 1% agreement with the theoretical predictions for $\langle N_t \rangle = 60$ and $\sigma_{N_t} = 5.48$, made in Section 1.4.2. 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.

All three scramblers, the random data, and the theoretical predictions are all consistent to within 1%. Comparing the two results for the Additive Scrambler, it is shown that

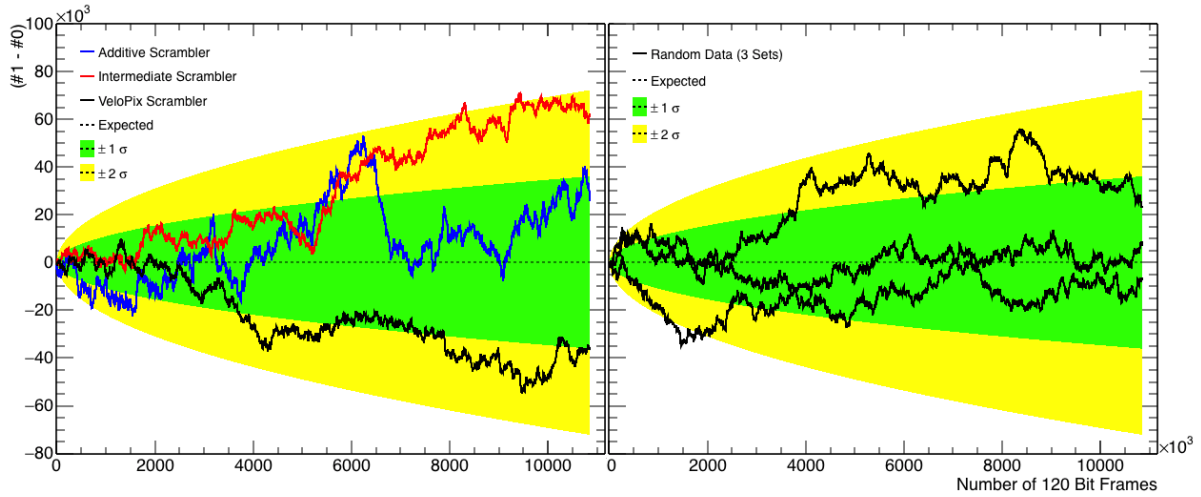


Figure 1.2: The results of the ‘*Bit Asymmetry*’ analysis.

while the frequency of longer chains is consistent with random data; but as the variance of transitions is larger than predicted, the long and short trains are more locally clustered.

The ‘*Bit Asymmetry*’ of each scrambler, shown in Figure 1.2, is consistent with the theoretical prediction. The deviation of $A_{1,0}$ for the predicted mean of 0 is fully consistent with stochastic noise. The random data also shows consistency. This gives confidence in the assumptions made in Section 1.4.2.

One notable feature of Figure 1.2 is the steep gradient of the additive scrambler at $t \sim 6 \cdot 10^6$. However, as the data stays within the theoretical limits and the ‘*drop*’ is of approximately $\Delta A_{1,0} \sim 60 \cdot 10^3$ over the range $n \Delta t \sim 1.2 \cdot 10^8$ it would be difficult to construct any argument claiming that this feature is of statistical significance.

(I am tempted to run χ^2 analysis for a fit of $y=0$ so show that the data is consistent with the model, but am not sure this will actually add to the argument?)

1.5 Conclusion

The consistency of random data and the theoretical predictions justifies the assumptions and approximations made in Section 1.4 and Section 1.4.2. Furthermore, the conformation 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.

	$\langle N_t \rangle$	σ_{N_t}	Gradient	p_t
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 1.1: The combined results of the algorithm analysis.

208 The Intermediate Scrambler produced an output consistant with random data. This
makes the algorithm suitable of data transfer. As already mentioned¹, however, the
210 scrambler is designed for computer simulated. As such, it is not suitable for implemen-
tation as it does not meet the additions requirments of the ASIC.

212 The VeloPix Scrambler, like the Intermediate Scrambler, produces a statistically scram-
bled output. Furthermore, the algorithm in inline with the additional requirments of the
214 ASIC. As such, it ideal for implementation, and hense is currently the choice algorithm
for use in the 2019 VELO upgrade.

¹Note to Marco: this is in the scrabler options section

216 References

- 218 [1] Altera. *Quartus Prime Software*. 2015. URL: <https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html> (visited on 12/2015).
- 220 [2] Mentor Graphics. *ModelSim - Leading Simulation and Debugging*. 2015. URL: <https://www.mentor.com/products/fpga/model/> (visited on 12/2015).
- 222 [3] Kurt Jacobs. *Stochastic Processes for Physicists - Understanding Noisy Systems*. Cambridge University Press, 2010. ISBN: 9780521765428.