

FPGA Development for the LHCb Vertex Locator Upgrade

Nicholas Mead
8064141

School of Physics and Astronomy
University of Manchester

December 18, 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 Introduction	1
	1.1 The Standard Model of Particle Physics	1
26	1.2 The LHCb Experiment	1
	1.2.1 The Detector	2
28	1.2.2 Physics Studied at LHCb	2
	1.2.3 VELO Upgrade	2
30	1.3 FPGAs in Particle Detectors	2
	1.3.1 Field Programable Gate Arrays	2
32	1.3.2 The Role of FPGA's in the VELO Upgrade	2
	2 Scrambling Algorithms	2
34	2.1 The Role of Scrambling Data in the VELO	3
	2.2 Scrambler Options	3
36	2.3 Cross Checks	4
	2.4 Algorithm Analysis	4
38	2.4.1 Measurements of the Algorithms	5
	2.4.2 Statistical Predictions	5
40	2.4.3 Results of Analysis	9
	2.5 Conclusion	10
42	3 Event Isolation Flagging	10
	3.1 Motivation	10
44	3.2 Time Sorting Data	10
	3.3 Bubble Sorting	10
46	3.4 Isotaton Checking	10
	3.5 Conclusion	10

48	4 Future Development	10
	4.1 LHCb 2020 Upgrade	10
50	4.2 Further Development of FPGA's in the VELO	10
	5 Conclusion	10
52	6 Acknowledgments	10
	References	11

1 Introduction

1.1 The Standard Model of Particle Physics

Central to the modern age of particle physics in the standard model,

$$\begin{aligned}
L_{GWL} = & \sum_f (\bar{\Psi}_f (i\gamma^\mu \partial_\mu - m_f) \Psi_f - e Q_f \bar{\Psi}_f \gamma^\mu \Psi_f A_\mu) + \frac{g}{\sqrt{2}} \sum_i (\bar{a}_L^i \gamma^\mu b_L^i W_\mu^+ + \bar{b}_L^i \gamma^\mu a_L^i W_\mu^-) \\
& + \frac{g}{2x_w} \sum_f \bar{\Psi}_f \gamma^\mu (I_f^3 - 2s_w^2 Q_f - I_6 e_f \gamma_5) \Psi_f Z_\mu - \frac{1}{4} |\partial_\mu A_v - \partial_v A_\mu - ie(W_\mu^- W_v^+ - W_\mu^+ W_v^-)|^2 \\
& - \frac{1}{2} |\partial_\mu W_v^+ - \partial_v W_\mu^+ - ie(W_\mu^+ A_v - W_v^+ A_\mu) + ig' c_w (W_\mu^+ Z_v - W_v^+ Z_\mu)|^2 \\
& - \frac{1}{4} |\partial_\mu Z_v - \partial_v Z_\mu + ig' c_w (W_\mu^- W_v^+ - W_\mu^+ W_v^-)|^2 - \frac{1}{2} M_\eta^2 \eta^2 - \frac{g M_\eta^2}{8 M_W} \eta^3 - \frac{g'^2 M_\eta^2}{32 M_W} \eta^4 \\
& + |M_W W_\mu^+ + \frac{g}{2} \eta W_\mu^+|^2 + \frac{1}{2} |\partial_\mu \eta + i M_Z Z_\mu + \frac{ig}{2c_w} \eta Z_\mu|^2 - \sum_f \frac{gm_f}{2M_W} \bar{\Psi}_f \Psi_f \eta. \quad (1.1)
\end{aligned}$$

The standard model, shown in equation 1.1, is a quantum field theory that describes the fundamental particles and how they interact. While this essay does require, or attempt, to understand the intricate detail of the standard model; the aim of many particle physics experiments is to Test, measure and varify the model. Dispite being the current best theory to explain particle interactions, the model is not complete. There are many undescribed phemimina, such as the matter domination in the universe, that require physics behind the standard model. To that end, major international efforts, namely in the form of the Large Hardrom Collider, aim to further knowledge and understanding of the underlying physics of the universe. [1]

1.2 The LHCb Experiment

One such Experiment and the Large Hadrom Colider is Large Hadron Colider beauty (LHCb). Located at intersection point

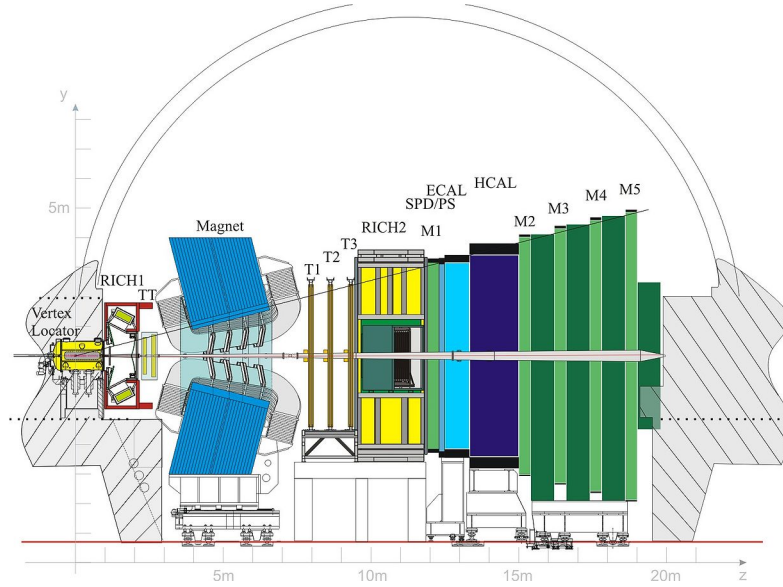


Figure 1.1: The LHCb Detector along the bending plane.

1.2.1 The Detector

1.2.2 Physics Studied at LHCb

1.2.3 VELO Upgrade

1.3 FPGAs in Particle Detectors

1.3.1 Field Programable Gate Arrays

1.3.2 The Role of FPGA's in the VELO Upgrade

2 Scrambling Algorithms

Due to radiation levels inside the detector chamber, the main data processing takes place in a concrete bunker away from the detector, minimising radiation damage to the hardware. To facilitate this, optical links, 20 per modular, are used to transfer the data from the front end VELO to the Data Aquizition FPGA (DAQ). When communicating data digitaly, the tranfering modual (TX) and the recieving modual (RX) must have synchronised clocks. When achieving this, there are three main approuches:

- I. Syncinize both the TX and RX from a single central clock.
- II. Transmit the TX clock to the RX modual.
- III. Use bit-changes in the data to coninuesly synchronise the RX clock.

The two former of these options, although the most convenient, are not appropriate for the VELO as they are susceptible to unforeseen delays that could cause desynchronisation. The latter, while less susceptible to delays, 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.

2.1 The Role of Scrambling Data in the VELO

For the reasons described in Section 2, it is necessary to ensure that the data has large density of transitions before being transmitted from the front-end detector to the DAQ module. However, as the majority of SP hitmaps are empty, the data has a large bias towards 0s. 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 DAQ FPGA.

Scrambling and later descrambling the data is not a trivial exercise. The scrambling (TX) module and descrambling (RX) module must use a synchronised 'key', derived from the previous states of the data. There are two options 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).

2.2 Scrambler Options

Three scrambling algorithms have been considered:

Additive Scrambler

This algorithm is simple and easy to use, however has the drawback of time dependence. If a desynchronisation event occurs, all subsequent data is rendered unrecoverable until such time as a global reset signal is sent. Further adding to the drawbacks, if a data packet is not sent from the TX during any clock cycle the RX descrambler will still evolve its descramble key - the TX scrambler, however, will not. This will of course desynchronise the 'keys', and as before all subsequent data is lost.

Intermediate Scrambler

Deriving its name from being the second algorithm under consideration, the Intermediate Scrambler is a **multiplicative** algorithm. The 'key' is generated from the current incoming frame and the previous frame. Therefore, in the event of desynchronisation, only two frames are lost before the 'key' is automatically recovered. This is a significant improvement over the Additive Scrambler.

VeloPix Scrambler

Named as, at the time of the start of the project (*September 2015*), this algorithm is the current preferred option by the VeloPix team; this too is a **multiplicative** algorithm. The ‘*key*’ is, again like the Intermediate Scrambler, generated from the current and previous data frame. The VeloPix Scrambler differs from the Intermediate scrambler as it aims to more efficiently scramble the data.

2.3 Cross Checks

Ofcourse, the main priorities when scrambling data is ensuring that the data is recoverable. For all three scramblers, the algorithm was synthesised in Quartus² and simulated in Modelsim³. The aim of synthesising and simulating the scramblers in these programs was to ensure that the design was physical in terms of on-board logic gates, and to check that the scrambled data was recoverable.

Furthermore, a C++ simulation was created for the three scramblers. This simulation had two purposes: firstly the output of the C++ can be checked against the Modelsim simulated check consistency; secondly to simulate the scrambler over a much larger sample of data as Modelsim simulations are less time efficient. In addition to the cross checks, the C++ code allowed for the descrambling to be delayed by several frames. The aim of this was to simulate a desynchronisation event. As expected, the additive scrambler was unable to recover any data, however the intermediate and VeloPix scramblers both recovered the ‘*key*’ after two frames and start descrambling data.

2.4 Algorithm Analysis

Intuitively, one can assume that fully scrambled data will be indistinguishable from randomly generated data. For this reason, the three algorithms are not only tested against each other and the pre-scrambled data but also randomly generated binary. The randomly generated data was created using the Python ‘*random*’ library, selecting a ‘0’ or ‘1’ with probability 1/2 each. While the Python ‘*random*’ library is only pseudo-random, on the scale of this example (i.e. > 100,000 frames), this is by far sufficient.

More mathematically rigorous, however, is to evaluate the system abstractly in the framework of statistical physics. In this abstraction, the ensemble is the 120 bit frame (with the header and parity removed); microstates are the particular form 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 2.4.1, predictions will be made using these principles and outlined in section 2.4.2.

In the context of the statistical model, it is reasonable to consider the degree of ‘*scrambledness*’ as entropy. Therefore a scrambled system can be assumed to be one of maximum entropy; and from Boltzmann’s law,

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

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

2.4.1 Measurements of the Algorithms

160 To compare the efficiency of the three algorithms in section 2.2, the algorithms were
 run over the same input data and compared for the following measures:

162 Number of Transitions Per Frame

164 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
 166 maximise the number of transitions.

Common Bit Chain Length

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

- 170 a) 10101010101111111111
- b) 10011001100110011001

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

Total Bit Frequency

178 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
 180 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.

182 2.4.2 Statistical Predictions

Number of Transitions Per Frame

184

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

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

186

At each time itteration,

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

188

However, 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 herefore be refered to as $p_t(t)$.

190

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

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

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

$$f_t(N_t) = \frac{n!}{N_t!(n - N_t)!} (p_t)^n \quad (2.5)$$

192

For $n = 120$, we can calulate,

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

$$\sigma_{N_t}^{Binomial} = \sqrt{n p_t^2} = 5.48 \quad (2.7)$$

194

Furthermore, when concidering the entropic argument in section 2.4 equation 2.1, the number of microstates corespoding to each macrostate N_t can be related to equation 2.5,

$$\Omega_t \sim \frac{n!}{N_t!(n - N_t)!} \quad (2.8)$$

$$\langle N_t \rangle^{Entropic} = MAX[S_t] = MAX[\Omega_t] \quad (2.9)$$

196

This can be numerically solved,

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

198

While the result of equation 2.10 does not contibute anything new, it is important as a ‘*sanity check*’. Because the system can be described as in section 2.4, it would indicated a problem in the theoretical framework if the result did not match.

200

Common Bit Chain Length

202

The probability of a chain of legth n is,

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

204

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

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

Takeing the log of both sides,

$$\log\left(\frac{N_n}{N_0}\right) = n \log(1 - p_t). \quad (2.13)$$

206

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

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

208

Total Bit Frequency

210

A , the assymetry of ‘1’s and ‘0’s is defined as,

$$A = N_1 - N_0, \quad (2.15)$$

212

where N_1 and N_0 are the number of ‘1’s and ‘0’s respectively. We can consider the evolution of A with frame t of size n as a stockastic iterative map with zero deterministic growth⁴,

$$A(nt + \Delta t) = A(nt) + \mathcal{N}(nt) \quad (2.16)$$

214

Where $\Delta t = \Delta frame \times n$ and \mathcal{N} is an independant random variable picked from a gaussian distribution. While $A(t) \in \mathbb{Z}$, in the limit of large nt we can approximate that A is continious.

216

If we consider the moments of A ,

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

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

218

Clearly, in Equation 2.17, $\langle A \rangle = 0$. In Equation 2.18, we assume the variance is of form $(\Delta t)^\alpha$ [4]. Then,

$$\langle A(t = M \Delta t)^2 \rangle = M(\Delta t)^\alpha. \quad (2.19)$$

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

$$\langle A(t = M \Delta t)^2 \rangle = t_f (\Delta t)^{\alpha-1}. \quad (2.20)$$

222 Considering the three cases of α :

- $\alpha > 1$: Here $A \rightarrow 0$ as $\Delta t \rightarrow 0$.
- 224 • $\alpha < 1$: Here $A \rightarrow \infty$ as $\Delta t \rightarrow 0$.
- $\alpha = 1$: This is the only sensible choice.

226 With $\alpha = 1$,

$$\langle A(t = M \Delta t)^2 \rangle = M(\Delta t). \quad (2.21)$$

And thus,

$$\sigma_A = \sqrt{\langle A^2 \rangle - \langle A \rangle^2} = \sqrt{\langle A^2 \rangle} = \sqrt{\Delta t}. \quad (2.22)$$

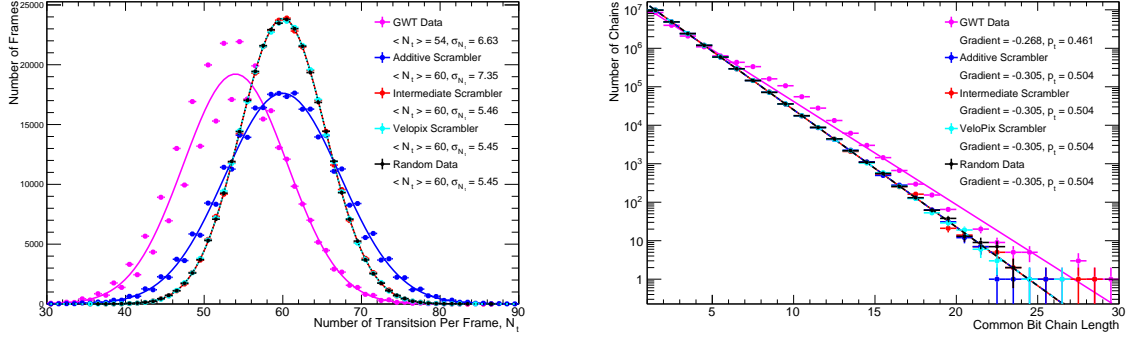


Figure 2.1: Results of the ‘*Number of Transitions Per Frame*’ analysis (Left) and the ‘*Common Bit Chain Length*’ analysis (right). The results for the Random Data, Intermediate Scrambler and VeloPix Scrambler overlap for the ‘*Number of Transitions Per Frame*’ analysis. The results for the Random Data, 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 2.1, 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 2.4.2. 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.

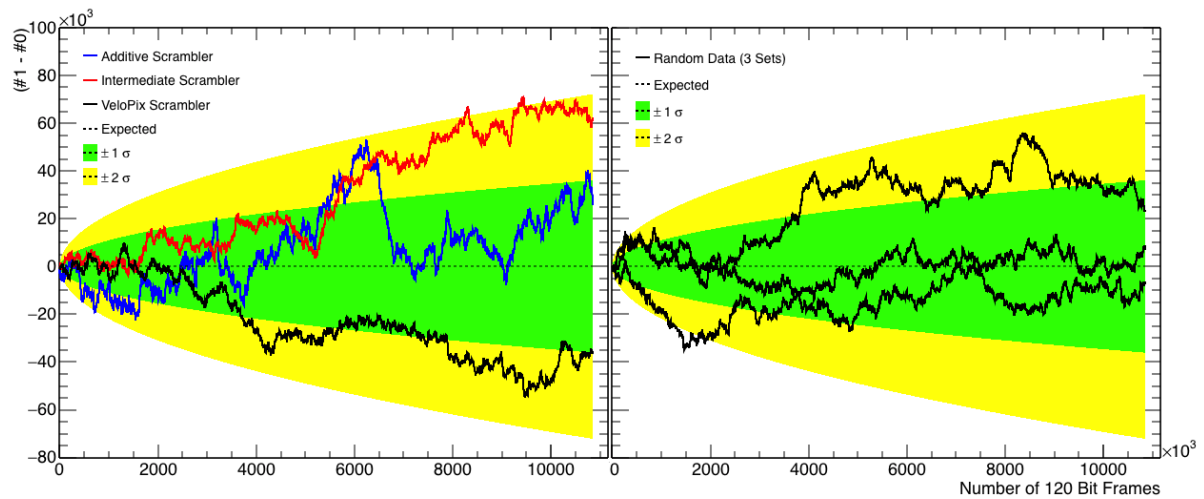


Figure 2.2: The results of the ‘Total Bit Frequency’ analysis.

2.5 Conclusion

3 Event Isolation Flagging

3.1 Motivation

3.2 Time Sorting Data

3.3 Bubble Sorting

3.4 Isotaton Checking

3.5 Conclusion

4 Future Development

4.1 LHCb 2020 Upgrade

4.2 Further Development of FPGA’s in the VELO

5 Conclusion

6 Acknoledgments

I would like the Acknowledge Pablo Rodriguez and Marco Gersabeck for there continued support and supervision.

References

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