# FPGA Development for the LHCb Vertex Locator Upgrade

Nicholas Mead
8064141
School of Physics and Astronomy
University of Manchester

December 15, 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

# 1 Introduction

## 1.1 The Standard Model of Particle Physics

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

$$\begin{aligned}L_{GWL} = \sum_f (\bar{\Psi}_f(i\gamma^\mu \partial\mu - m_f)\Psi_f - eQ_f\bar{\Psi}_f\gamma^\mu\Psi_f A_\mu) + \frac{g}{\sqrt{2}}\sum_i(\bar{a}^i_L\gamma^\mu b^i_L W^+_\mu + \bar{b}^i_L\gamma^\mu a^i_L W^-_\mu) \\ + \frac{g}{2x_w}\sum_f \bar{\Psi}_f\gamma^\mu(I^3_f - 2s^2_w Q_f - I6e_f\gamma_5)\Psi_f Z_\mu - \frac{1}{4}|\partial_\mu A_v - \partial_v A_\mu - ie(W^-_\mu W^+_v - W^+_\mu W^-_v)|^2 \\ - \frac{1}{2}|\partial_\mu W^+_v - \partial_v W^+_\mu - ie(W^+_\mu A_v - W^+_v A_\mu) + ig' c_w(W^+_\mu Z_v - W^+_v Z_\mu|^2 \\ - \frac{1}{4}|\partial_\mu Z_v - \partial_v Z_\mu + ig' c_w(W^-_\mu W^+_v - W^+_\mu W^-_v)|^2 - \frac{1}{2}M^2_\eta \eta^2 - \frac{gM^2_\eta}{8M_W}\eta^3 - \frac{g'^2 M^2_\eta}{32M_W}\eta^4 \\ + |M_W W^+_\mu + \frac{g}{2}\eta W^+_\mu|^2 + \frac{1}{2}|\partial_\mu\eta + iM_Z Z_\mu + \frac{ig}{2c_w}\eta Z_\mu|^2 - \sum_f \frac{gm_f}{2M_W}\bar{\Psi}_f\Psi_f\eta. \quad (1.1)\end{aligned}$$

The standard model, shown in equation 1.1, is a quantum field theory that discribes the fundermental particles and how they interact. While this essay does require, or attempt, to understand the intricate detail of the stardard 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 behond 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

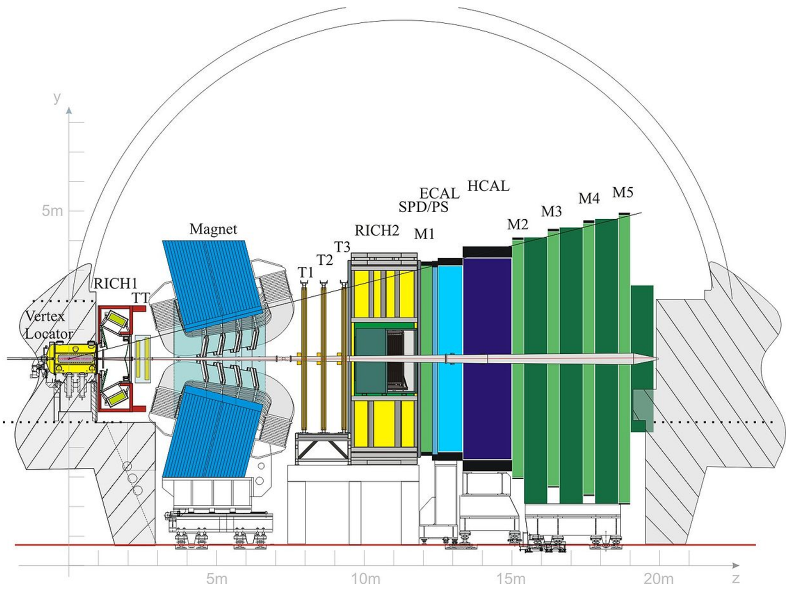


Figure 1.1: The LHCb Detector along the bending plane.

### 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 linkes, 20 per modual, are used to transfer the data from the front end VELO to the Data Aquizition FPGA (DAQ). When comunicating data digitaly, the tranfering modual (TX) and the recieving modual (RX) must have syncrinised clocks. When achieving this, there are three main approunches:

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 sycnronise the RX clock.

The two former of these options, although the most convienient, are not appropriate for the VELO as they are suseptable to unforseens delays that could cause desyncronisation. The latter, while less suseptable to delays, requires data with a high density of tranitions to reduce the likelyhood of a desyncronisation event. Becuase 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 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 SP hitmaps are empty, the data has a large bais towards 0s. This reduces the frequancy of transitions in the data - increasing the probability of a desyncronisation event. It is therefor nesseccary to scramble the data pria to transmition and descramble the data in the DAQ FPGA.

Scrambleing and later descrambleing the data is not a trivial exercise. The scrambleing (TX) modual and descrambling (RX) modual must use a sycronised *'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 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).

## 2.2 Scrambler Options

Three scrambling algorithums have been concidered:

**Additive Scrambler**
This algoritum is simple and easy to use, however has the drawback of time dependance. If a desyncronisation event occours, all subsequent data is rendered unrecoverable untill 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 sccrambler, however, will not. This will ofcourse desyncronise the *'keys'*, and as before all subsequent data is lost.

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

**VeloPix Scrambler**

Named as, at the time of the start of the project *(September 2015)*, this algorithum is the current preffered option by the VeloPix team; this too is a **multiplicative** algorithm. The *'key'* is, again like the Intermediate Scrambler, generaged from the currect and previous data frame. The VeloPix Scrambler differse from the Intermediate scrambler as it aims to more effeciently scramble the data.

## 2.3 Cross Checks

Ofcourse, the main prioritys when scrambling data is ensuring that the data in recoverable. For all three scramblers, the algorithum was sysnthesised in Quartus[2] and simulated in Modelsim[3]. The aim of sysnthesising and simulating the scramblers in these 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.

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 simulated 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 C++ code allowed for the descrambling to be delayed by several frames. The aim of this was to simulate a desycronisation event. As expected, the additive scarmbler was unable to recover any data, however the intermediate and VeloPix scarmblers 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 indistinguisable from randomly 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 data was created using the Python *'random'* library, selecting a *'0'* or *'1'* with probubility 1/2 each. While the Python *'random'* library is only sudo-random, on the scale of this example (i.e. $>$ 100,000 frames), this is by far sufficient.

More mathematicaly rigorus, however, is to evaluade 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 statisical model, it is reasonable to concider the degree of *'scrambledness'* as entropy. Therefor a scrambled system can be assumed to one of maximum entropy; and from Boltzmans law,

$$S \sim ln(\Omega) \tag{2.1}$$

where $\Omega$ is the number of microstates assosiated with the macrostate, we learn that this state of maximum entropy is a macrostate with the maximum number of assosiated microstates.

### 2.4.1 Messurements of the Algorithms

To compare the effecincy of the three algorithums in section 2.2, the algorithums where run over the same unput data and compared for the following measures:

**Number of Transitions Per Frame**

This meassure 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 20 bit frames,

a) 10101010101111111111

b) 10011001100110011001

both with 10 transitions, will be concidered equal. However, (b) is clearly a more suitable output for data transfer as (a) has a large probability of desyncronisated due to the long chains of *'1'*s. It is therefore also nessecary to evaluate the length of common bit chains within the scrambled data.

**Total Bit Frequancy**

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

### 2.4.2 Statistical Predictions

**Number of Transitions Per Frame**

Concider a particle in a symmetric, descrete time-dependent, two state system,

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

At each time itteration,

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

However, as $p_{1\to 0}(t)$ is equal in both probalility and importance to $p_{0\to 1}(t)$, the probability of a bit change shall herefore be refered to as $p_t(t)$.

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,

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

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

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

For $n = 120$, we can calulate,

$$< N_t >^{Binomial} = \sum_{N_t=0}^{n-1} N_t\ f(N_t) = n\ p_t = 60 \tag{2.6}$$

$$\sigma_{N_t}^{Binomial} = \sqrt{n\ p_t^2} = 5.48 \tag{2.7}$$

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,

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

$$< N_t >^{Entropic} = MAX[S_t] = MAX[\Omega_t] \tag{2.9}$$

This can be numerically solved,

$$< N_t >^{Entropic} = 60 \tag{2.10}$$

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.

### Common Bit Chain Length

The probability of a chain of legth $n$ is,

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

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,

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

Takeing the log of both sides,

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

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 $p_t = 0.5$,

$$log(1 - p_t) = -0.30 \; . \tag{2.14}$$

### Total Bit Frequancy

$A$, the assymetry of *'1'*s and *'0'*s is defined as,

$$A = N_1 - N_0, \tag{2.15}$$

where $N_1$ and $N_0$ are the number of *'1'*s and *'0'*s respectively. We can concider the evolution of $A$ with frame $t$ of size $n$ as a stockastic itterative map with zero deterministic growth[4],

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

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

If we concider the moments of $A$,

$$< A(t = M \; \Delta t) > = \sum_{m=0}^{M-1} \mathcal{N}(m \; n \; \Delta t), \tag{2.17}$$

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

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

$$< A(t = M\ \Delta t)^2 > = M(\Delta t)^\alpha. \tag{2.19}$$

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,

$$< A(t = M\ \Delta t)^2 > = t_f\ (\Delta t)^{\alpha-1}. \tag{2.20}$$

Conciding the three cases of $\alpha$:

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

With $\alpha = 1$,

$$< A(t = M\ \Delta t)^2 >= M(\Delta t). \tag{2.21}$$

And thus,

$$\sigma_A = \sqrt{< A^2 > - < A >^2} = \sqrt{< A^2 >} = \sqrt{\Delta t}. \tag{2.22}$$

### 2.4.3 Results of Analysis

Figure 2.1 shows histograms of the number of transition per frame, as outlined in Section 2.4.1. The qualitative results of this analysis are shown in Table 2.1.

| | Mean | $\sigma$ |
|---:|---|---|
| Unscrambled Data | 54 | 6.63 |
| Additive Scrambler | 60 | 7.35 |
| Intermediate Scrambler | 60 | 5.45 |
| VeloPix Scrambler | 50 | 5.46 |
| Random Data | 60 | 5.45 |
| Theoretical Prediction | 60 | 5.48 |

Table 2.1: The results of the *'Number of Transitions Per Frame'* analysis.

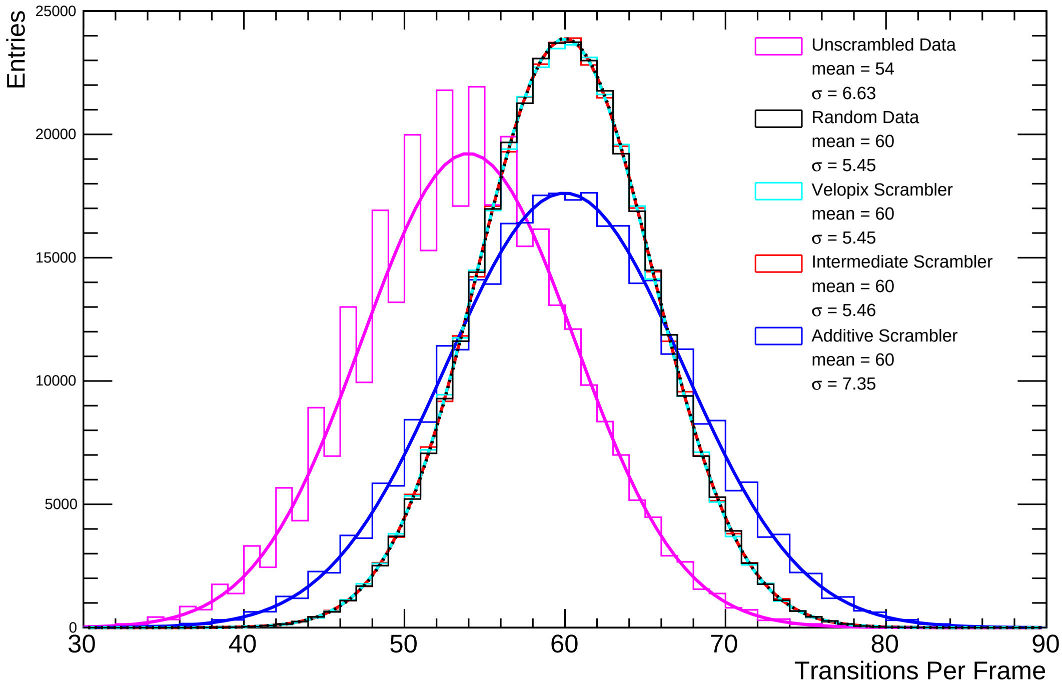


Figure 2.1: The results of the *'Number of Transitions Per Frame'* analysis. The histogram and fit for the Random Data, Intermediate Scrambler and VeloPix Scrambler approximatly overlap.

## 2.5 Conclusion

# 3 Event Isolation Flagging

## 3.1 Motivation

## 3.2 Time Sorting Data

## 3.3 Bubble Sorting

## 3.4 Isotation Checking

## 3.5 Conclusion

# 4 Future Development

## 4.1 LHCb 2020 Upgrade



## 4.2 Further Development of FPGA's in the VELO

# 5 Conclusion

# 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.