

# FPGA Development for the LHCb Vertex Locator Upgrade

Nicholas Mead

8064141

School of Physics and Astronomy  
University of Manchester

December 10, 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

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	The Standard Model of Particle Physics . . . . .	1
1.2	The LHCb Experiment . . . . .	1
1.2.1	The Detector . . . . .	2
1.2.2	Physics Studied at LHCb . . . . .	2
1.2.3	VELO Upgrade . . . . .	2
1.3	FPGAs in Particle Detectors . . . . .	2
1.3.1	Field Programable Gate Arrays . . . . .	2
1.3.2	The Role of FPGA's in the VELO Upgrade . . . . .	2
<b>2</b>	<b>Scrambling Algorithms</b>	<b>3</b>
2.1	The Role of Scrambling Data in the VELO . . . . .	3
2.2	Scrambler Options . . . . .	4
2.3	Algorithm Analysis . . . . .	4
2.3.1	Messurements of the Algorithms . . . . .	5
2.3.2	Statistical Predictions . . . . .	5
2.3.3	Results of Analysis . . . . .	5
2.4	Conclusion . . . . .	5
<b>3</b>	<b>Event Isolation Flagging</b>	<b>6</b>
3.1	Motivation . . . . .	6
3.2	Time Sorting Data . . . . .	6
3.3	Bubble Sorting . . . . .	6
3.4	Isotaton Checking . . . . .	6
3.5	Conclusion . . . . .	6

<b>4</b>	<b>Future Development</b>	<b>7</b>
4.1	LHCb 2020 Upgrade . . . . .	7
4.2	Further Development of FPGA's in the VELO . . . . .	7
<b>5</b>	<b>Conclusion</b>	<b>8</b>
<b>6</b>	<b>Acknowledgments</b>	<b>8</b>

# 1 Introduction

## 1.1 The Standard Model of Particle Physics

Central to the modern age particle physics is the standard model,

The standard model, shown in equation ??, 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 verify 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. To that end, major international efforts, namely in the form of the Large Hadron Collider, aim to further knowledge and understanding of the underlying physics of the universe. [?]

## 1.2 The LHCb Experiment

One such Experiment and the Large Hadron Collider is Large Hadron Collider beauty (LHCb).

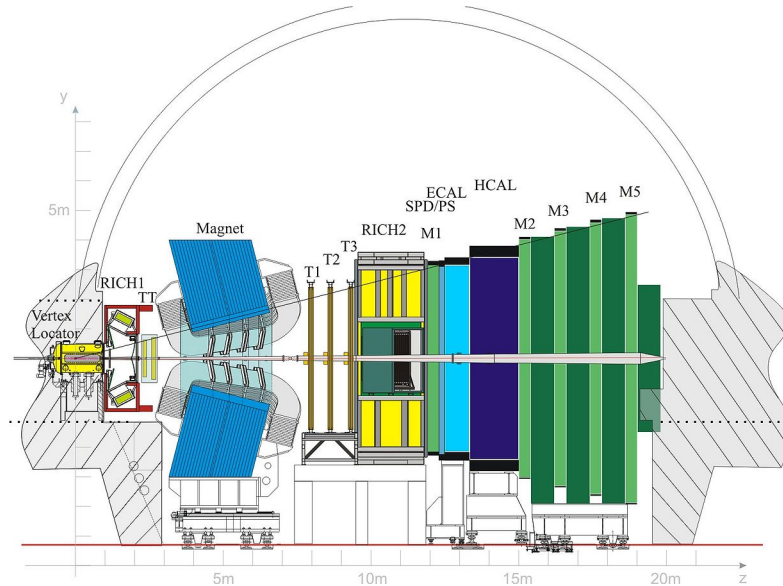


Figure 1: The LCHb Detector along the bending plane.

### **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 Acquisition FPGA (DAQ). When communicating data digitally, the transferring modular (TX) and the receiving modular (RX) must have synchronised clocks. When achieving this, there are three main approaches:

- I. Synchronise both the TX and RX from a single central clock.
- II. Transmit the TX clock to the RX modular.
- III. Use bit-changes in the data to continuously 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 modular. 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) modular and descrambling (RX) modular 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 ofcourse 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 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 2.3.1, prediction will be made using these principles and outlined in section 2.3.2.

### **2.3.1 Measurements of the Algorithms**

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:

**Number of Transitions Per Frame**

**Common Bit Chain Length**

**Total Bit Frequency**

### **2.3.2 Statistical Predictions**

### **2.3.3 Results of Analysis**

## **2.4 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

## 6 Acknowledgments

I would like to Acknowledge Pablo Rodriguez and Marco Gersabeck for their continued support and supervision.