Analysis of scrambling algorithms and implementation of isolated event flagging for the LHCb Vertex Locator upgrade

Benjamin Jeffrey
8393011
School of Physics and Astronomy
University of Manchester

May 16, 2016

Abstract

The LHCb VErtex LOcator (VELO) will undergo a significant upgrade in 2018. The data processing of the signal involves stages of scrambling the signal and flagging isolated events.

Analysis was performed on three competing scrambling algorithms, an additive scrambler, the intermediate scrambler, and the VELO scrambler. The number of transitions the scramblers introduced to a signal was investigated, as was the chain length of 1s and 0s, as well as an analysis of the balance of 1s and 0s in the signal. The intermediate scrambler and the VELO scrambler performed identically to a set of random data, and both performed better than the additive scrambler in terms of number of transitions. There was no significant difference between the scramblers in terms of balance of 1s and 0s and in the chain length analysis.

The full method for the isolated event flagging has been coded in VHDL. The details of the method are presented in this document. Analyses and discussion of the timing constraints and flagging acceptance are also presented. It was found if a cut off of 64 was applied to the number of signals accepted, in the hottest ASICs the acceptance does not drop below 0.95. The next step for the isolated event flagging unit is testing for resources, and if successful, implementation into the full FPGA code.

Contents

1 Introduction to the Large Hadron Collider and its goals

1.1 The Standard Model of Particle Physics

The Standard Model of Particle Physics is the theoretical model used by particle physicists that attempts to describe fundamental particles and their interactions, and how these lead to the phenomena observed in the Universe. It incorporates electromagnetism, the weak nuclear force and the strong nuclear force, however the force of gravity is not included. It is a highly successful theory, and is being probed at the Large Hadron Collider (LHC) and other experiments around the world. The theory is also known to be incomplete, as there are observations it cannot describe, such as the magnitude of CP violation and neutrino oscillations.

1.2 The Large Hadron Collider

The LHC [1] is the largest hadron collider in the world with the highest energy, performing proton-proton collisions and lead ion collisions. It is located at the European Organization for Nuclear Research (CERN) near Geneva in Switzerland. The layout is shown in Figure 1.

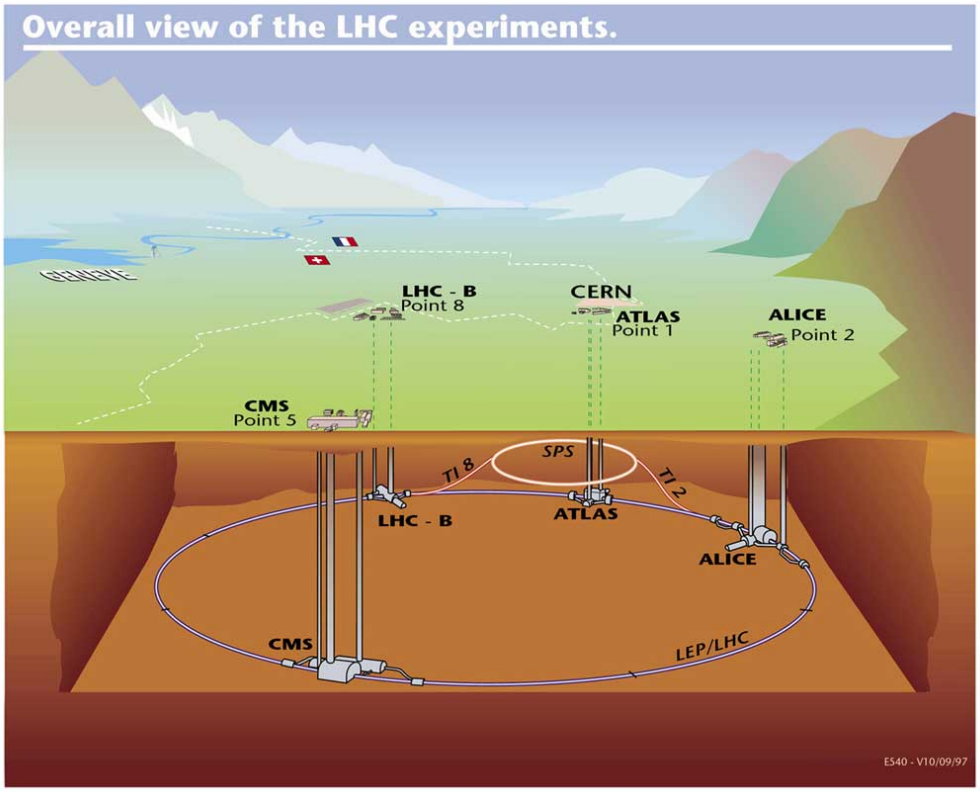

Figure 1: The LHC at CERN [1].

The LHC is the latest of a series of colliders at CERN, and occupies the same underground tunnel that was used for the Large Electron Positron collider (LEP), which was operational until 2000. The Super Proton Synchotron (SPS), originally a hadron collider experiment in its own right, is now used for the final injection into the main LHC beamline. The LHC is 27 km long, and at present operates at an energy of 13 TeV, the highest energy of a collider in history.

There are four main experiments located around the LHC, one at each of the four points where the counter rotating proton beams collide. A Toroidal LHC ApparatuS (ATLAS) and Compact Muon Solenoid (CMS) are two general purpose detectors, looking at a broad range of the spectrum of particle physics, and are the largest detectors at the LHC. The Large Hadron Collider beauty (LHCb) detector searches for new physics by studying CP violation and rare decays. The final main experiment is A Large Ion Collider Experiment (ALICE), which investigates the strong nuclear interaction by looking at lead ion collisions.

2 The LHCb detector

2.1 Aims

The LHCb experiment is one of four main experiments located at the LHC. Its aim is searching for new physics (NP) beyond the Standard Model (BSM). Key to this is understanding and quantifying asymmetries between matter and antimatter by studying B mesons, which are particles which contain b-quarks (bottom/beauty). The LHCb experiment also probes the interactions and decays of particles containing charm quarks.

Matter-antimatter asymmetry is an essential condition for the existence of the matter dominated universe that we inhabit. Charge parity (CP) violation is a measure of this asymmetry, and is required as per the Sakharov conditions for an abundance of baryons over antibaryons [2]. Charge parity is a proposed discrete symmetry of the laws of physics under the combination of charge conjugation and parity transformation. Charge conjugation turns a particle into its antiparticle, and parity transformation is an inversion of spatial coordinates. This symmetry is observed to be broken in weak decays [3], and the LHCb experiment investigates the extent and causes of this violation. The LHCb experiment also searches for rare decays, which are decays that are heavily suppressed by the Standard Model, but could occur via new virtual particles that exist in BSM physics.

2.2 Structure of the Detector

The LHCb detector is a single arm forward spectrometer, shown in Figure 2. The VErtex LOcator (VELO) sits closest to the collision point, and aims to provide detailed information on the location of the decay vertices of the collisions. VELO is the most precise vertex location system at the LHC, and is required due to the short lifetimes of B mesons,

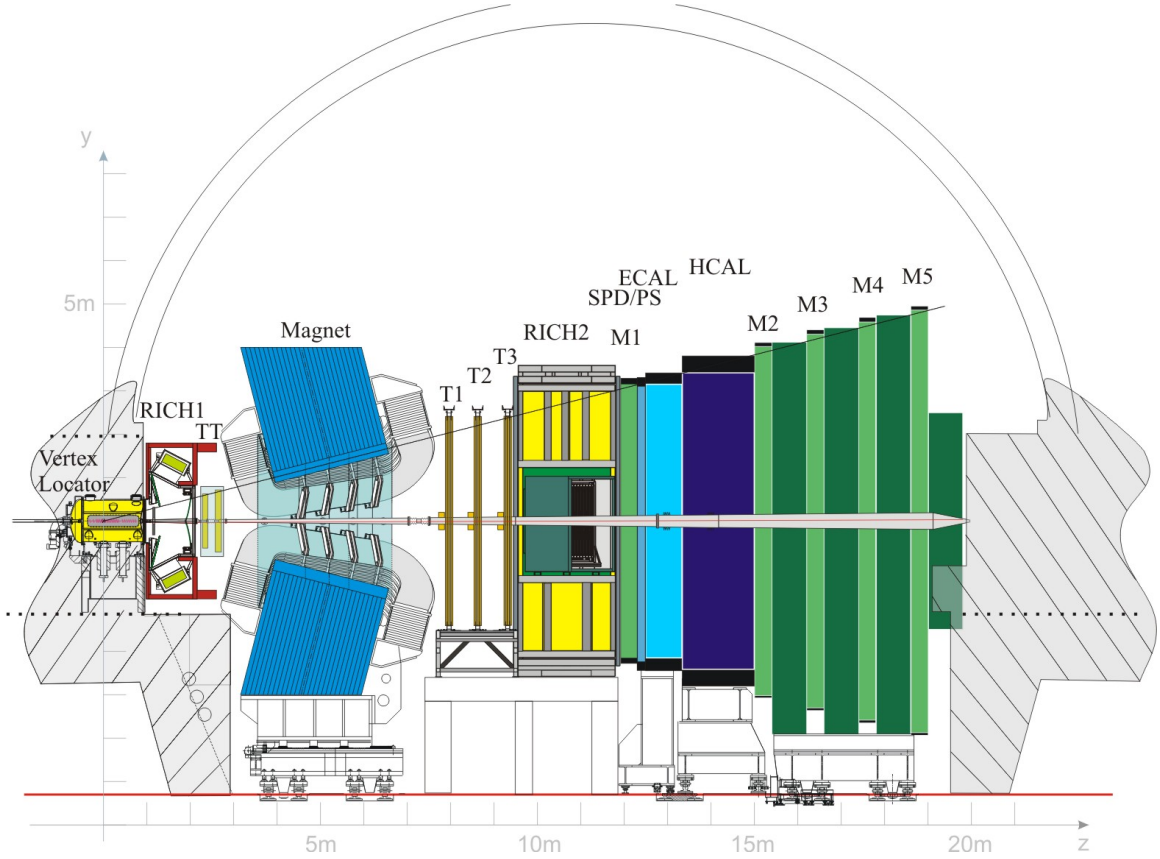

Figure 2: Schematic of the LHCb detector [4]. The beam travels in the z direction in this representation

meaning that the B meson decay vertex occurs extremely close to the collision point.

RICH1 and RICH2 are two ring imaging Cherenkov detectors. Cherenkov radiation is produced when a particle moves through a medium faster than light through the same medium. It is used to ascertain the velocity of particles.

Placing magnets around the detector bends the tracks of the particles. Trackers can then measure the curvature of these tracks, which is related to the momentum of the particles. From this momentum information and the velocity information obtained from the RICH detectors, particle identification is possible.

The Electromagnetic CALorimeter (ECAL) aims to measure the energy of particles that interact via the electromagnetic force, i.e. photons and all electrically charged particles. The ECAL consists of alternating scintillating tiles and lead tiles.

The Hadronic CALorimeter (HCAL) measures the energy of hadrons using interactions with the strong nuclear force. The HCAL consists of alternating scintillating tiles and thin iron plates.

M1 to M6 are Muon detectors, which aim to identify the tracks of muons passing through the detector. The Muon Detectors consists of chambers full of carbon dioxide, argon, and tetrafluoromethane. Any muons produced in the collision react with the gases and allow

their detection.

2.3 Upgrade Schedule

The LHC is in a continuous cycle of data collecting and upgrading [5]. The projected timeline is shown in Figure 1. Initially the LHC operated at a centre of mass energy of 7 TeV, whereas currently in Run 2 it is at 13 TeV. Part of any of the upgrade phases is maintenance, since the parts of detector closest the beam are subject to heavy radiation damage, affecting the performance of the detector. The detectors are also upgraded with new technologies that have become available, increasing performance compared to the initial design.

The LHCb detector is being significantly upgraded in the Long Shutdown 2, beginning 2018. The main aims of the upgrade are:

- Increasing the luminosity to $2\text{x}10^{33}$ $\text{cm}^{-2}\text{s}^{-1}$. This is 5 times greater than the current luminosity, and 10 times greater than design luminosity.
- Increasing front end electronics readout to 40MHz, the same as the bunch crossing frequency. Presently the front end electronics are limited to 1 MHz readout.
- Implementing event selection in the software, as opposed to the hardware as it is now. This change is due to the efficiency limitations of the hardware.

Time period (approx)	Energy/TeV	Luminosity/fb^{-1}	Upgrades
2011, Run 1	7	up to 30	Increase luminosity to 30fb^{-1}
2012, Run 1	8	30	Energy increased to 8TeV
2013-2014, LS1	n/a	n/a	Energy increased to 13∼14 TeV
2014-2018, Run 2	13∼14	up to 150	Increase luminosity to 150fb^{-1}
2018-2019, LS2	n/a	n/a	Energy increased to 14 TeV, Luminosity increased to 300fb^{-1}. Phase one of experiment upgrades.
2019-2022, Run 3	14	300	n/a
2022-2025, LS3	n/a	n/a	Phase two of experiment upgrades. HL-LHC installed

Table 1: Approximate upgrade schedule of the LHC up to the High Luminosity LHC (HL-LHC) installation. LS stands for Long Shutdown.

3 VELO Upgrade

3.1 Current VELO

The current VELO [6] is comprised of 84 single sided measuring strip sensors, mounted in pairs back to back on 42 modules. The modules are split into two groups that are centred around the beam. The active part of the VELO operates 8.2 mm from the beam axis, placing the modules inside the LHC secondary vacuum. In order to protect the modules a thin aluminium radio frequency (RF) foil is placed between the modules and the beam vacuum. To prevent excessive radiation damage during beam injection, the VELO modules are retracted by 3 cm radially during injection.

3.2 Planned Upgrade

The VELO upgrade aims to improve vertex precision, readout speed, and radiation hardness. The distance to beamline will be reduced, as will the amount of material in the modules. The data bandwidth will be increased by a factor 40, radiation load a factor 10. The current VELO modules are unable to cope with this extra load, so the modules and sensors have been redesigned. The modules in the upgraded VELO will be based on pixel sensors instead of microstrips. These have higher radiation tolerance. The increased bandwidth means that new data acquisition (DAQ) boards have been developed, which will be cooled using biphase CO_2, pumped through microchannels in the substrate of the module.

3.2.1 Modules

In the upgraded VELO a total of 52 modules will be used, 26 on each side of the beamline. Each module will contain 4 sensors, two on each side. Each sensor has 3 Application Specific Integrated Circuits (ASICs). The ASICs that are going to be used are VeloPix ASICs [7]. These are based on the Timepix3 ASICs, but adapted for the use in the VELO. The Timepix3 can produce information of Time-Over-Threshold (proportional to the charge collected) and Time-Of-Arrival (time-stamp of the hit) simultaneously, as well as the hit information.

The VeloPix will only read binary information on the hit, in order that it will be able to cope with the 900 MHits/s at the upgrade luminosity. The VeloPix ASIC has a 256x256 pixel matrix, each pixel has size 55μm x 55μm. Each of the ASICs has a 2x4 pixel structure in a SuperPixel Packet (SPP). This SPP structure has the advantage of allowing data to share address and time-stamp, reducing the amount of data transmitted. The 4 ASICs are bonded onto a silicon substrate which has microchannels etched into it. The microchannels are for pumping through biphase CO_2 at -30°C, cooling the module when in use.

3.2.2 Data aquisition board

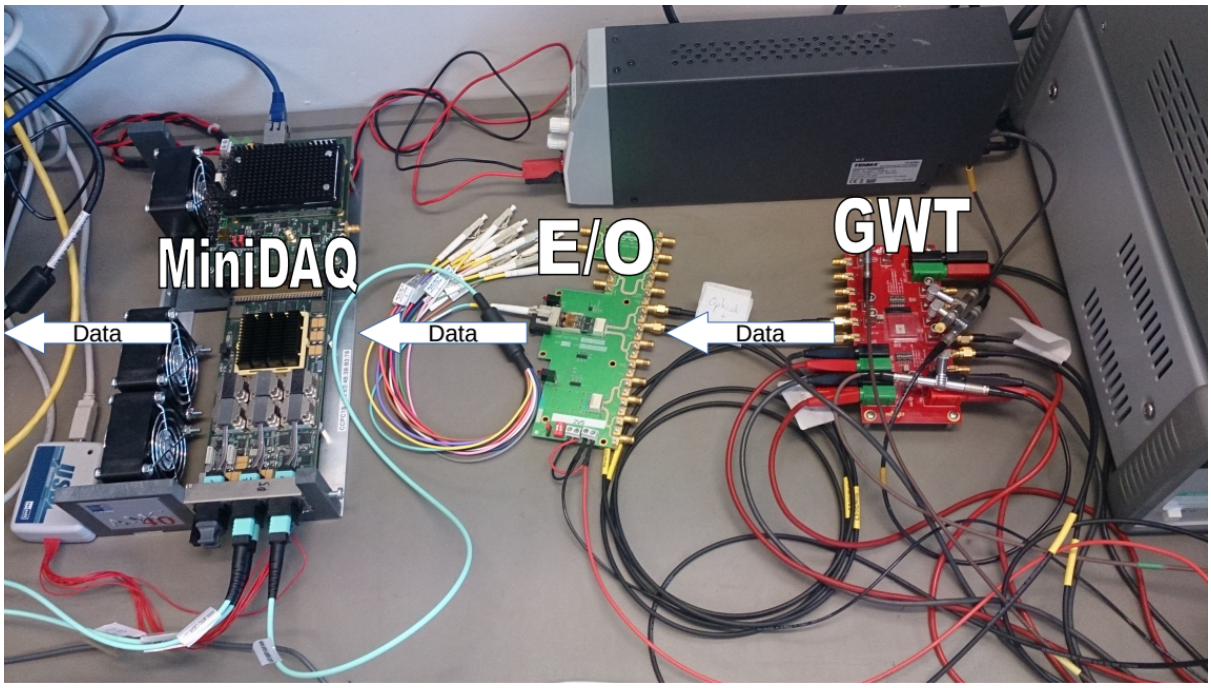

Figure 3: The data path for the MiniDAQ setup. The data is serialised in the GWT serialiser, converted to from electrical to optical (E/O), processed in the MiniDAQ, then sent to a computer via a 10 Gbit/s link.

The DAQ board that will be used in the VELO upgrade is named PCIe40, based on an Field Programmable Gate Array (FPGA) by Altera^{TM}, the ArriaX FPGA, mounted on a PCIe x 16 Generation 3 card. FPGAs are reprogrammable silicon chips that can be programmed using Hardware Descriptive Languages (HDLs), common examples of which are Verilog and VHSIC Hardware Descriptive Language (VHDL). The DAQ has 48 input and 48 output links, and an two-way in-out link for Timing and Fast Control (TFC). The DAQ will deliver a bandwidth of up to 100 Gbit/s. These are to be available to the LHCb group during 2016 for testing.

Presently a lower resource, 24 link MiniDAQ is being used for developing the data processing framework, and testing the serialiser. The set-up is shown in Figure 3.

The serialiser used in the VeloPix is a Gigabit Wireline Transceiver (GWT) compared the LHCb experiment standard GigaBit Transciever (GBT). The GWT sends 128 bit frames, that have fixed header and datalengths. It is also faster than the GBT, whilst being less power demanding.

3.2.3 Structure of the GWT frame

The structure of the GWT frame is shown in Figure 4. The header used is 0101. The parity bits, one for each of the SPPs. It is 1 if the signal is odd, 0 if even. The Bunch Cross ID (BCID) is the counter by which the router is aligned to the signal. The counter

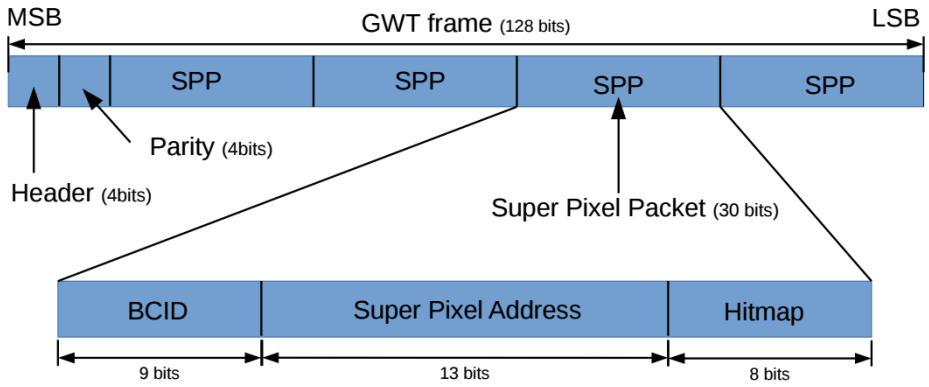

Figure 4: The structure of the 128 bit GWT frame. The SPP signal is expanded and shown. MSB and LSB stand for Most Significant Bit and Least Significant Bit respectively.

Decimal	Hexadecimal	Binary	Gray
0	0	0000	0000
1	1	0001	0001
2	2	0010	0011
3	3	0011	0010
4	4	0100	0110
5	5	0101	0111
6	6	0110	0101
7	7	0111	0100
8	8	1000	1100
9	9	1001	1101
10	A	1010	1111
11	B	1011	1110
12	C	1100	1010
13	D	1101	1011
14	E	1110	1001
15	F	1111	1000

Figure 5: Table showing the differing signals of a counter using hexadecimal, binary and Gray code.

is incremented by the LHCb experiment 40MHz clock. The BCID is implemented in Gray code rather than binary, see Figure 5, and has a maximum value of 511 as counting starts from 0. Gray code is more efficient than binary as in Gray code only a single bit changes in each incrementation. The Super Pixel Address contains location of the Super Pixel that the signal has been sent from. The Hitmap contains the particle hit information.

3.2.4 Data Processing

The data arrives at the MiniDAQ from the serialiser through an optical link. It is received by a transceiver which produces a 32 bit frame at 160 MHz. The output of the transceiver is passed to the Low Level Interference (LLI_GWT) which processes the data. The data processing chain is shown in Figure 6. The LLI_GWT locates the LHCb experiment

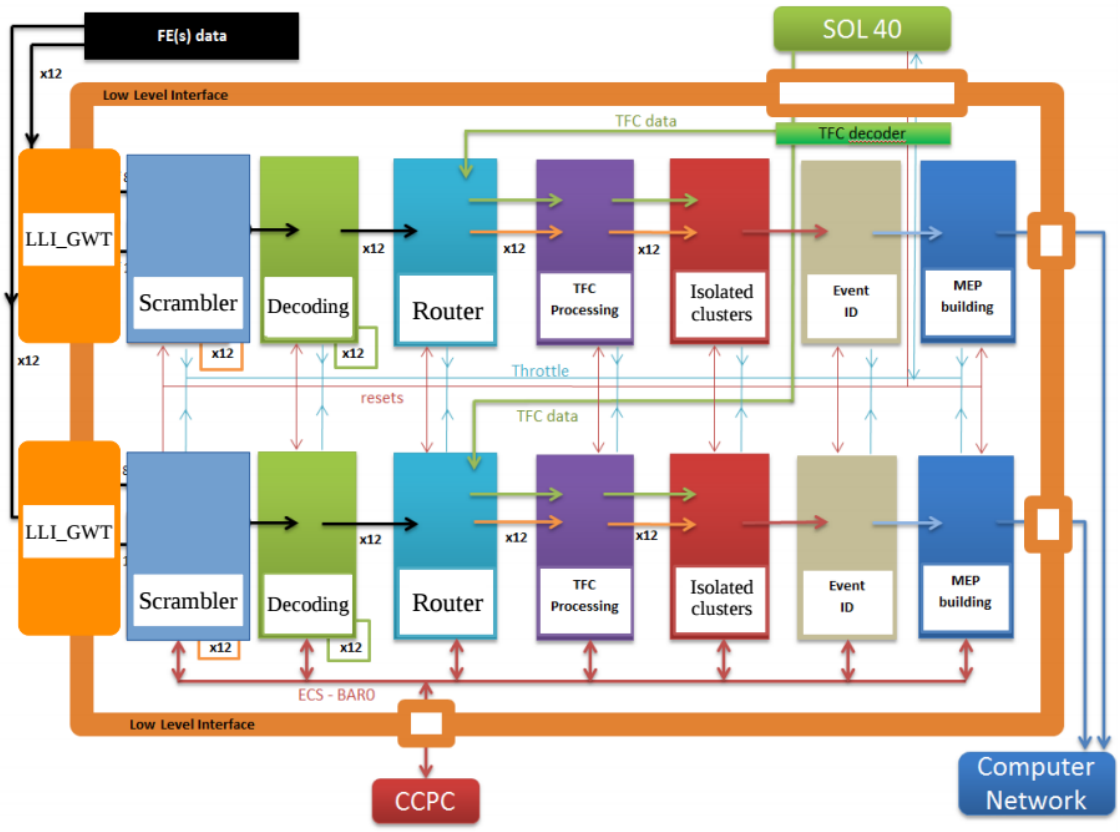

Figure 6: This figure shows the data processing schematic [8] for the upgraded VELO.

header and from this constructs the 128 bit GWT frame, shown in 4. The GWT frame is split up into its individual SPP, and each of these SPP is processed individually.

- Scrambler: Descrambles the data. The purpose of the Scrambler is discussed in Section 4.
- Decoding: Translates the BCID of the SPP from Gray code to binary, allowing the Router to order the SPPs by BCID.
- TFC Processing: Implements the BCID-veto signals.
- Isolated Clusters: Identifies isolated hits, i.e. hits that have no neighbours. This is discussed in Section 5.
- Event ID: Constructs the output frame back to the GWT frame structure, so that the data can be sent on to the computer network.

4 Scrambling Analysis

4.1 Need for scrambling

The DAQ needs to reconstruct the clock from transitions in the data signals, in order that it can provide a clock and data when it sends the data onwards. For the clock to be reconstructed accurately there needs to be a large number of transitions. In order to lower the chance of the clock becoming out of sync with the data, the largest possible number of transitions is wanted. This is achieved using a scrambling algorithm. There are several algorithms that have been used to fulfil this task, and analysis was performed comparing them.

4.2 Difference between additive and multiplicative scramblers

Additive scramblers depend on the current input, the previous input and the order in which the signals have come previously. This is effectively a time dependence of the scrambler. Multiplicative scramblers only depend on the current input and the previous input. The impact of this is that an additive scrambler keeps scrambling data indefinitely, regardless of whether the data being passed is new data.

A multiplicative scrambler will not keep scrambling if the data being passed to it is not changing. This difference becomes important when the data needs to be descrambled in order to be analysed later in the data processing chain. If desynchronisation occurs, an additive scrambler would not be able to recover data until a common reset signal had been sent, whereas the multiplicative scrambler would only lose two clock cycles of data.

4.3 Scrambler comparison

Three scramblers were compared in this analysis. The additive scrambler was the original scrambler used. It requires two blocks of two-input XOR gates.

The intermediate scrambler was developed by K.Hennessey, University of Liverpool, and is a multiplicative scrambler that was developed for use in software emulation.

The VeloPix scrambler is the scrambler that is going to be implemented in the upgraded VELO system, and is also a multiplicative scrambler. It was designed with the particular constraints of the ASIC used, in particular that it used less combinational logic.

4.4 Analysis

The analysis of the algorithms was performed in C++. This gave more flexibility, and increased speed of analysis. It was also easier to produce analysis plots. Random data that

was produced according to binomial statistics was also used for comparison. See *Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences* by R.J.Barlow [9] for a discussion on the binomial distribution and its properties. Before analysis was attempted, the scrambling products of the C++ and VHDL were compared to check they were identical. The data used for analysis was a Monte Carlo (MC) generated simulation sample provided by K. Hennessey. The MC simulation consisted of single file simulations for each ASIC, so for the analysis these files were combined into a single file, meaning analysis could be run over the largest number of frames possible. Note that for the analysis, the header and parity bits were not counted, as these are kept unscrambled.

4.4.1 Analysis of number of transitions

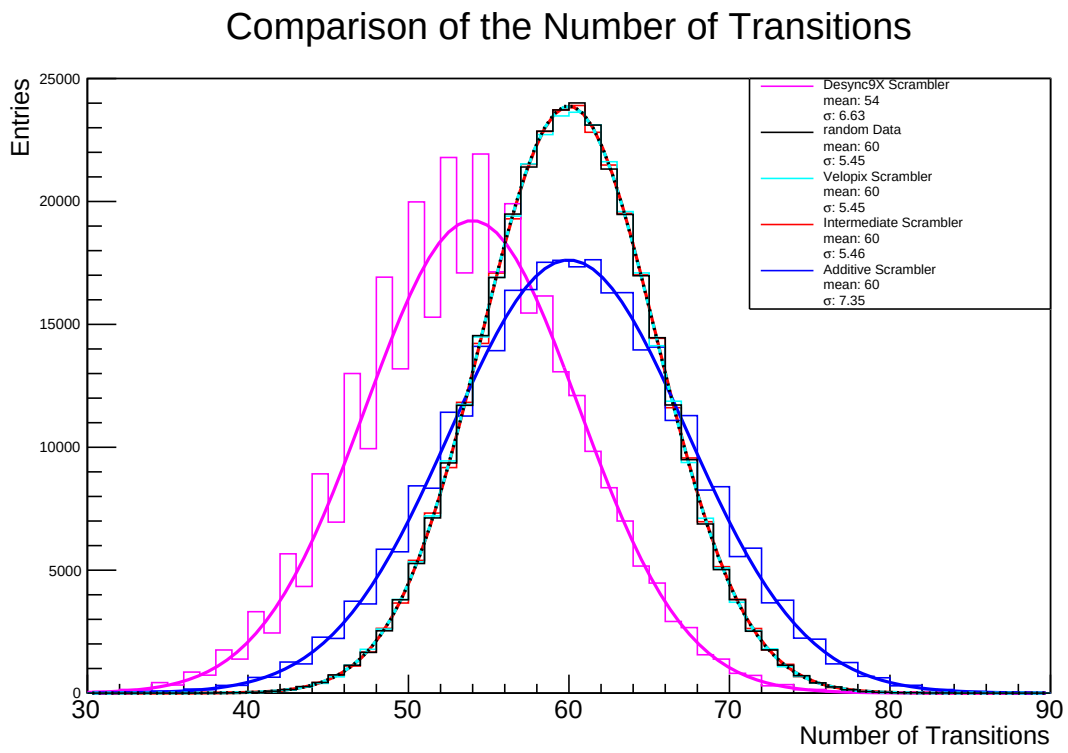

Figure 7: Graph showing the number of transitions in the GWT frame (not including header and parity bits) for the additive scrambler, the intermediate scrambler, the VeloPix scrambler, the unscrambled data, and a set of random data. All the scramblers increase the number of transitions compared to the unscrambled data. Both of the multiplicative scramblers perform better than the additive scrambler. The multiplicative scramblers have an output identical to that of the random data.

The first part of the analysis consisted of counting the number of transitions between 1 and 0, and 0 and 1, in each signal. The results are shown in Figure 7. Each of the curves was fitted with a Gaussian to allow for statistics to be generated. The results are tabulated in Table 2.

All scramblers increase the number of transitions when compared to unscrambled data,

Scrambler	Mean number of transitions	Standard deviation
Unscrambled	54	6.63
Additive	60	7.35
Intermediate	60	5.46
VeloPix	60	5.45
Random data	60	5.45

Table 2: Table of results of the analysis of the number of transitions

with the two multiplicative scramblers having near identical outcome to the randomly generated data. Both multiplicative scramblers have a lower standard deviation compared to the additive scrambler, which is significant as the signals most likely to cause desynchronisation of the reconstructed clock are those with a low number of transitions.

4.4.2 Chain length analysis

Figures 8 and 9 show that all of the scrambling algorithms reduce the frequency of the larger chain lengths. They also appear to increase the chains of 1s in the data. This can be explained by the fact that the scrambling is introducing more 1s into a dataset that has a heavy bias towards 0s. The results are evidence that all of the scramblers restore the balance of 1s and 0s when compared to the original unscrambled data.

4.4.3 Analysis of balance of 1s and 0s

The balance of a signal gives a measure of the difference between number of 1s and 0s in the signal. The balance is incremented up when there is a 1, and incremented down when there is a 0. It can be seen in Figure 10 that all the scramblers perform similarly, all having balances with a relatively low absolute value when run over the large number of frames. The unscrambled data can not usefully be shown on the graph, the balance becomes extremely negative quickly, and on the scale of the graph it appears to lie directly over the y-axis. This implies the signal is mostly 0s, which makes sense as the unscrambled SPPs contain mostly empty hitmaps.

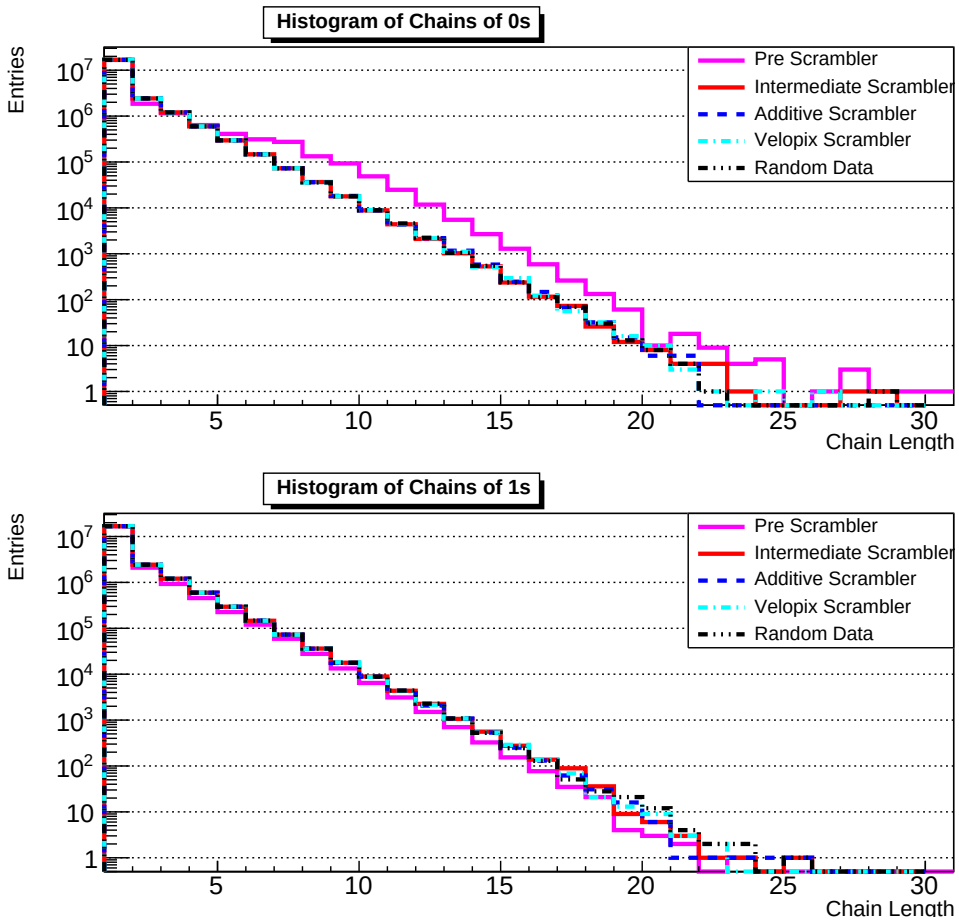

Figure 8: Histograms showing the frequency of chain lengths for chains of 0s (top graph) and 1s (bottom graph) with a logarithmic scale. It can be seen that the scrambling breaks up the long chains of 0s. This is supported by the results of the transition analysis. All of the scramblers perform similarly in reducing the chain lengths of 0s. Although the histogram of chains of 1s appears to show that the scramblers have increased the frequency of the chains of 1s. However this can be explained by the increase in the number of 1s in the scrambled data compared to the unscrambled data. This is confirmed by the analysis on the balance of the data.

5 Isolated event flagging

5.1 Purpose and desired outcomes

The isolated event flagging has the aim of identifying SPPs which have no hits in neighbouring columns. This is being developed to implement in the FPGA, so that the load on the computer network later on in the data path is reduced. It simplifies that task of track reconstruction. The CPU load for the LHCb experiment is very large, and reducing this load will reduce the chance of overloading the CPUs. The data in the FPGA is also analysed in real time as it comes in.

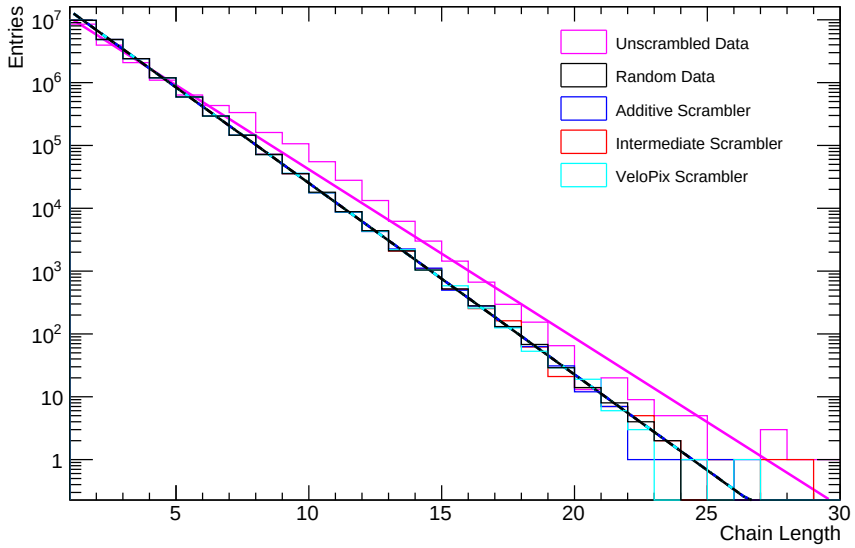

Figure 9: Histogram of the frequency of chains of either 1s or 0s, with a logarithmic scale on the y axis, fitted with a line of best fit for each dataset. This shows how all of the scramblers reduce the frequency of the longer chain lengths and increase the frequency of the shorter chain lengths. All of the scramblers perform similarly to a set of random data

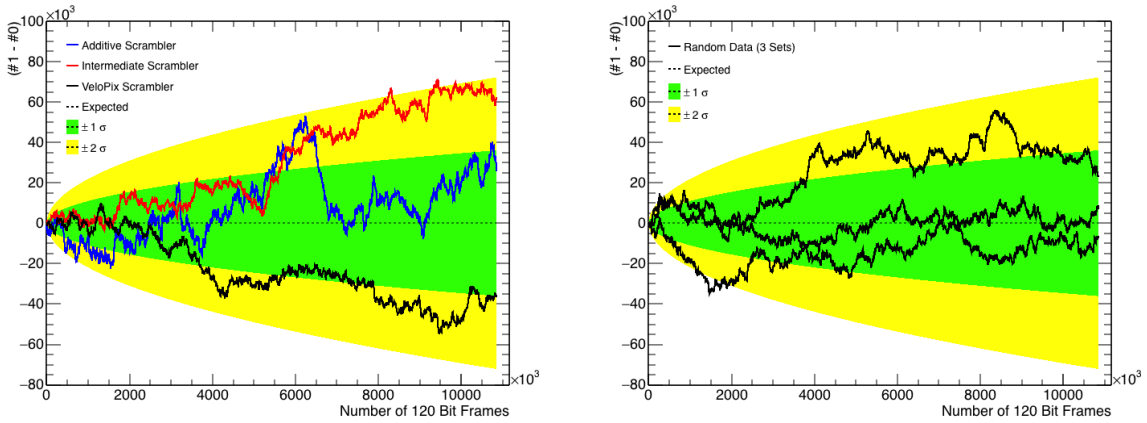

Figure 10: Graphs showing the balance of 1s and 0s. The graph on the left is for the three scramblers. The graph on the right shows 3 sets of randomly generated data. Error bands of 1 and 2 sigma for the random data have been added, showing how all of the scramblers stay within approximately 2 sigma of the random data.

The desired outcome of the project was to have developed and tested a full block containing the entire coded method for the event isolated flagging. If used this module will be able to be inserted into place in the full FPGA code with minimal changes.

5.2 Method

The basic method first involves sorting data packets of SPPs with the same BCID (called data trains) by order of chip ID and column ID. The data train is an array of SPPs, and is used for signal propagation. This sorting algorithm is an odd-even sort, a parallelised bubble sort. A representation of the odd-even sort is shown in Figure 11.

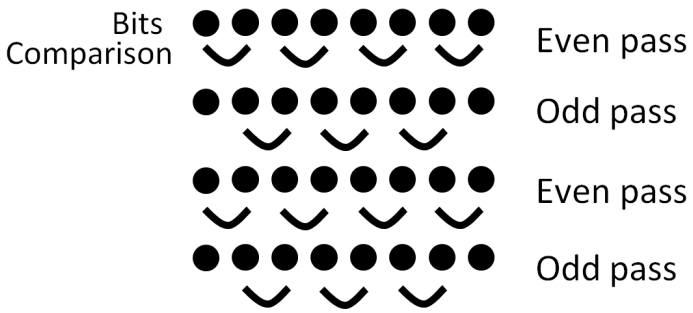

Figure 11: Diagram showing the pairwise comparisons of the odd-even sorting method. As the signals are binary, the bit labelling starts at 0, meaning the pairing (0,1) is an even sorting.

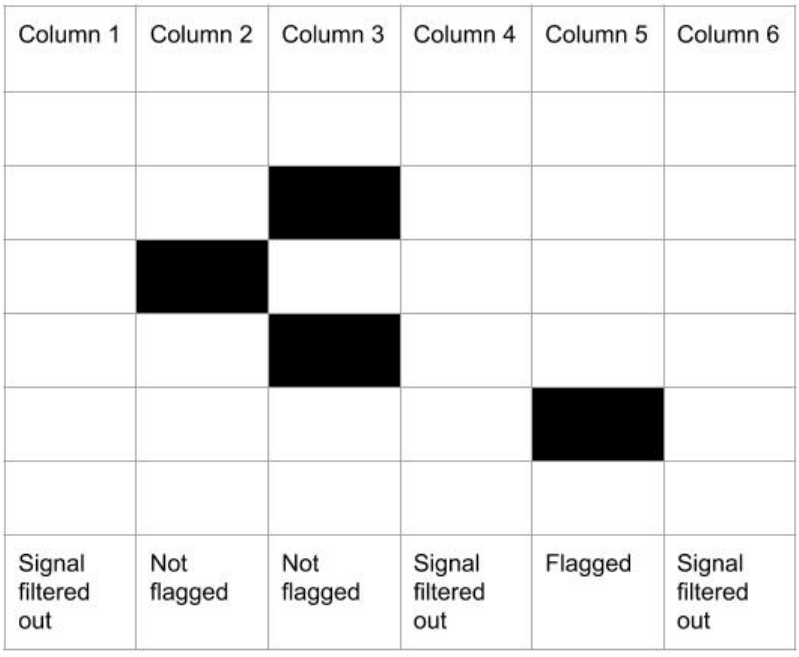

Figure 12: Simplified pixel grid. Columns 1, 4 and 6 would have been filtered out as they are empty. Columns 2, and 3 would not be flagged. Column 5 would be flagged as an isolated signal, since columns 4 and 6 are not present.

The limit on allowable resources and time taken means that a maximal number of SPPs can be sorted. If the number of SPPs in the BCID is larger than this number, the data is not sorted and bypasses the flagging process.

After the data train has been sorted, the data train is iterated over to locate SPP signals where the direct neighbouring columns are not present. This works because in the data processing in the DAQ all empty SPP signals are removed from the data. The shortcut of only sorting by column can be implemented as it is very unlikely that a single column will have more than one isolated hit. If an isolated hit is found, a flag bit is added to the signal to identify it. A simple grid showing isolated and non-isolated hits is shown in Figure 12;

5.3 Sorting analysis

5.3.1 Sort acceptance

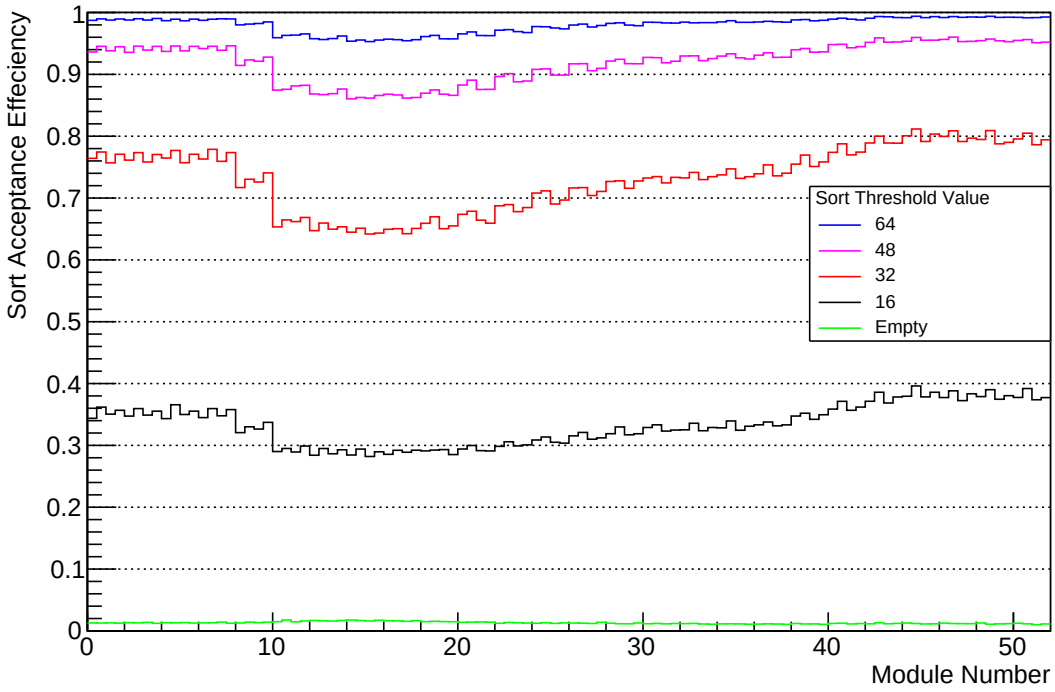

Figure 13: Sort acceptance with threshold values of 16, 32, 48 and 64

The impact of the maximal size of data train allowed in the flagging process on the acceptance rate per module was investigated. Threshold sizes of 16, 32, 48 and 64 were investigated. The results are shown in Figure 13.

The threshold of 64 performs well, in the hottest modules in VELO, modules between 10 and 20, the acceptance never drops below 0.95. Because of this, when the flagging code block was constructed, the threshold was set at 64.

5.3.2 Sort time

There are three options for sorting in terms of the amount of time taken to sort the data. Every clock cycle one pass of the odd-even sort is made. The options are static sorting, dynamic sorting, and semi-static sorting.

- Static: sort for a number of clock cycles equal to the threshold data train size.
- Semi-static : sort for a number of clock cycles equal to the number of SPPs in the data train.
- Dynamic: after each sort, check whether a switch has been made. If a switch has not been made in consecutive odd and even passes, the data is sorted.

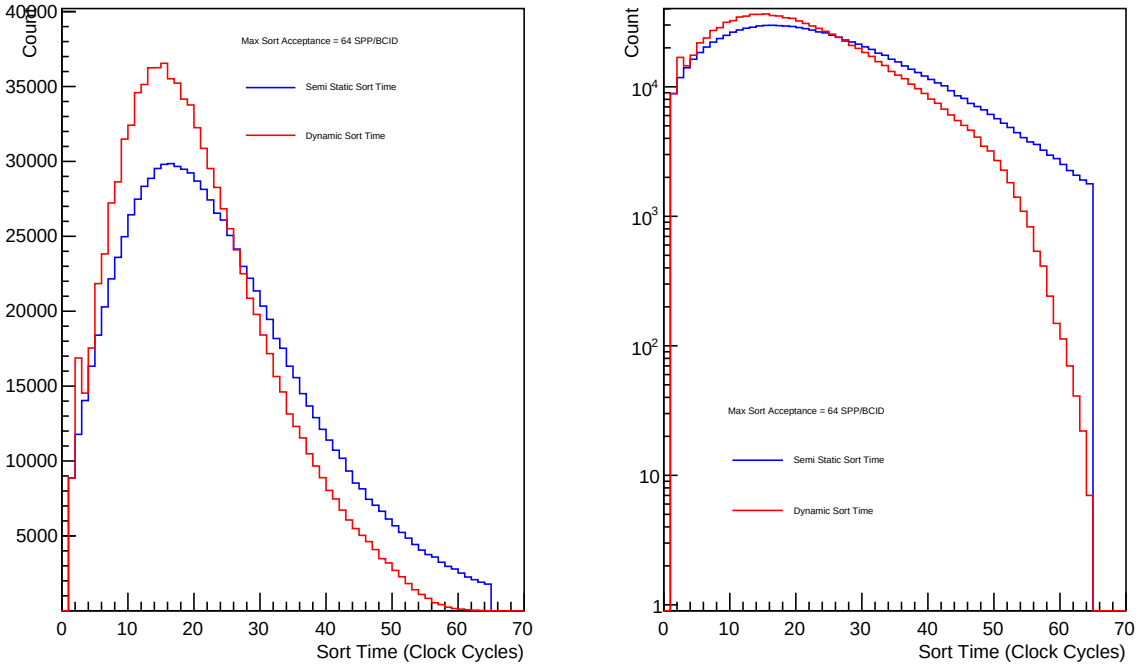

Figure 14: Sort time in clock cycles for the dynamic sorter compared to the semistatic sorter.

The static sorting timing is the absolute worst case scenario timing for every sorting, i.e. 64 clock cycles, so it was decided that static sorting could not be justified. The data train of SPPs are fixed sized arrays, and if they are not full with SPPs, padding of empty 0 signals are added at the end.

The semi-static array sorts for a number of clock cycles the number of SPPs in the data train, which is the worst case scenario for that number of SPPs. So for a data train containing 10 SPPs, sorting will take 10 clock cycles.

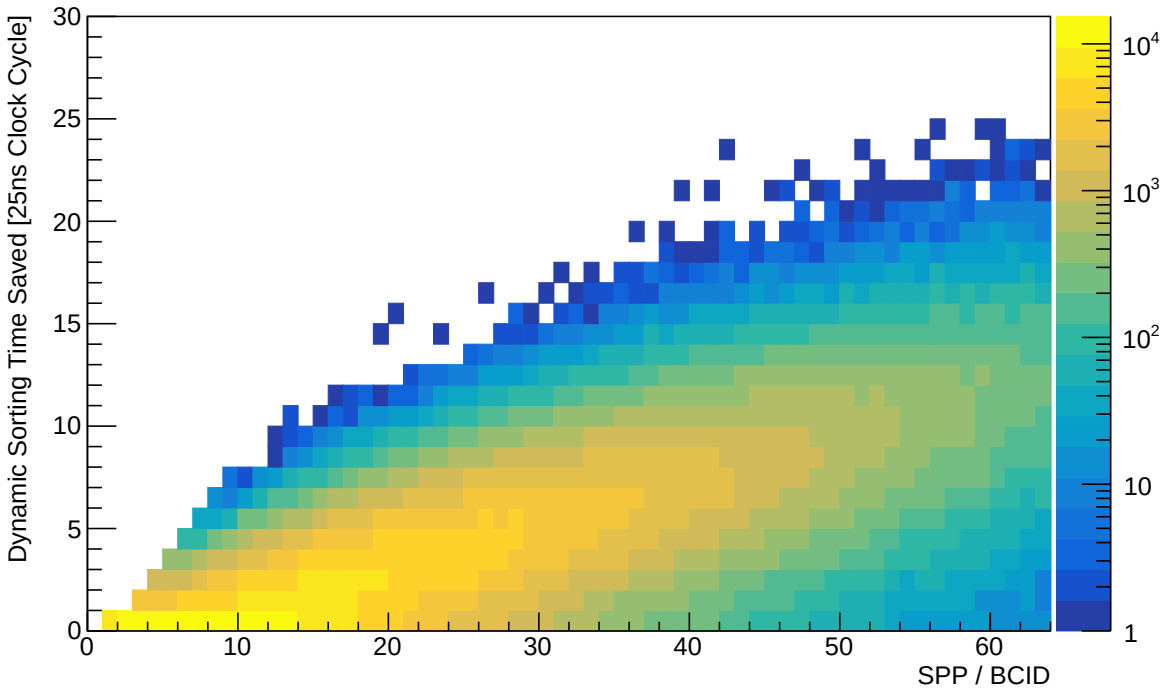

Figure 15: Heat map of the time saved by dynamic sorting as opposed to semi-static sorting. Note the logarithmic dependence of the heat map.

For the dynamic sorting, after each pass through the sort, a check is performed about whether there was a switch made. This adds extra logic checks at the end of each sort, and extra signals that need to be handled.

The number of clock cycles taken for dynamic sorting vs semi-static sorting was investigated. The results of this are shown in Figures 14 and 15. The dynamic sorting does save time for the sorting, however the time saved is largely below 5 clock cycles.

Due to the extra logic and signals required for the dynamic sorting, it was decided that the static sorting was the semi static sorting would be implemented. The semi-static sorting has the advantage that the information about the number of SPP in the BCID already has to be read to decide whether to bypass. This means the number of extra signals required is kept to a minimum.

5.4 Implementation

5.4.1 RAM units

The SPP data is stored in RAM units, and so these need to be read from at the start of the flagging process and flagged data needs to be written out to another set of RAM units at the end. The router, the unit in the DAQ before isolation flagging, orders the

data by BCID and fills the data into RAM units. There are 16 data RAMs, each covering 32 BCIDs. There is also a count RAM, which stores the number of SPP signals in each BCID.

5.4.2 Entity hierarchy

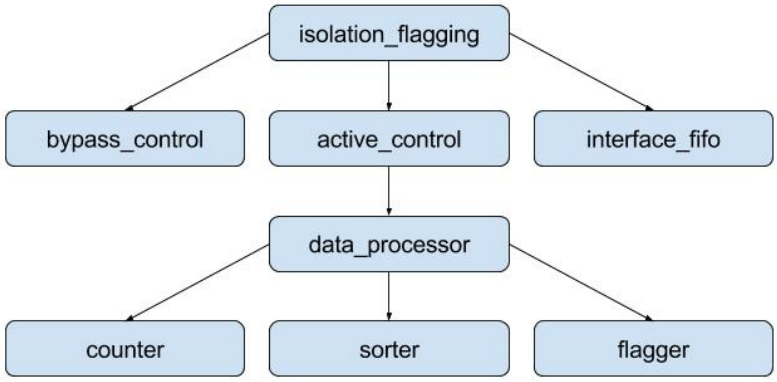

Figure 16: Hierarchy for the entities contained in the block.

There are several levels of entity within the code block that control various parts of the process. The hierarchy is shown in Figure 16.

5.4.3 Entity: data_processor

This is the controlling unit that passes the data between the sorter and flagger, and keeps track of the amount of time the data has been in the sorter, so it can ascertain when sorting has finished. The flow is shown in Figure 17.

The decision about whether new data has been read in is made using a signal that is sent from the active_control unit. When new data has been read in it is passed to the sorter and the counter is started. Every clock cycle one pass of the sorter is made and the counter is iterated. The sorting is finished when the counter value is equal to the number of SPP in the data train.

Once the counter has reached this value, the data is passed to the flagger, which takes one clock cycle to flag the SPPs as isolated or not isolated. After this the data is passed to the output port, where it stays until read by the active control. The data processor sends a signal to the active control so the active control knows that the data processor has finished with the data train. When the active control reads the data from the data processor, it sends a signal so the data processor knows to read in the next data train, and the process begins again.

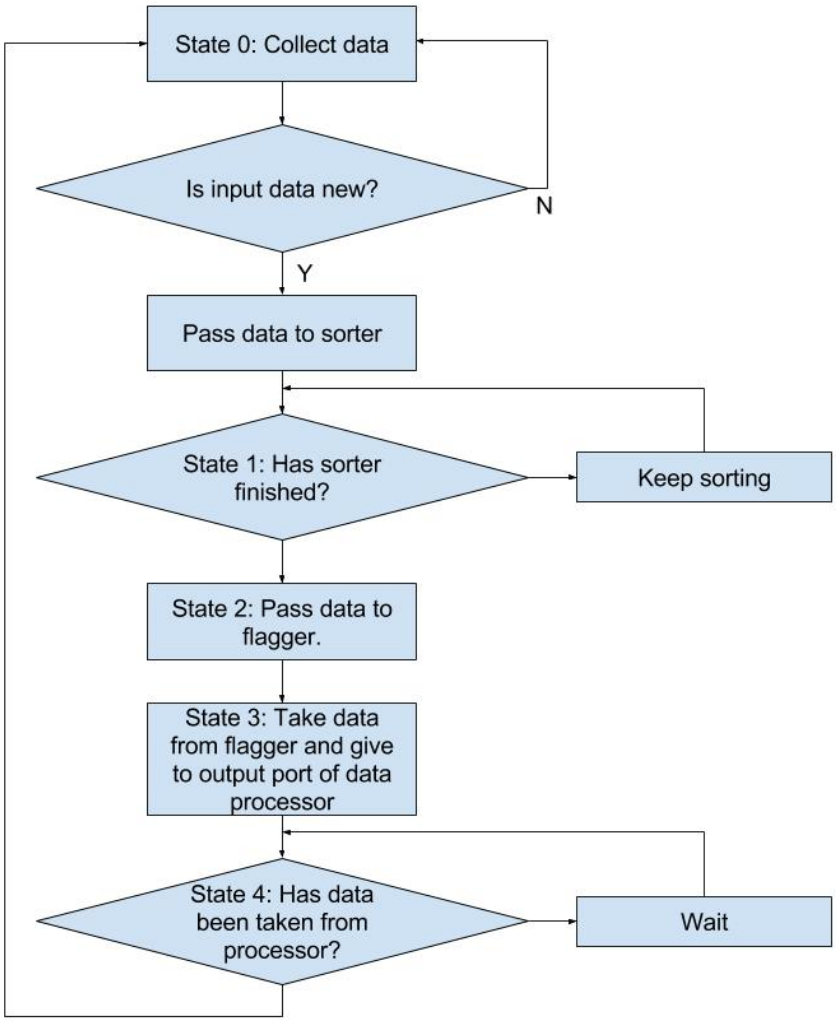

Figure 17: Flow diagram for the data processor entity

5.4.4 Entity: active_control

The active control entity generates a number of data processors, and is tasked with passing data to and from the data processors. It also reads the information from the count RAM about the number of SPPs in the BCID, and decides whether the the data is to be bypassed or processed. The active control also formats the data coming to and from the data processor, as the formats needed vary across the RAMs, and the data processors use the data train format. The input data RAM contains multiple 24 bit SPPs, whereas the output data RAM takes multiple 32 bit SPPs, with the 25th bit being the isolated flag. The data train is an array of 32 bit SPPs. To construct the data train, the read

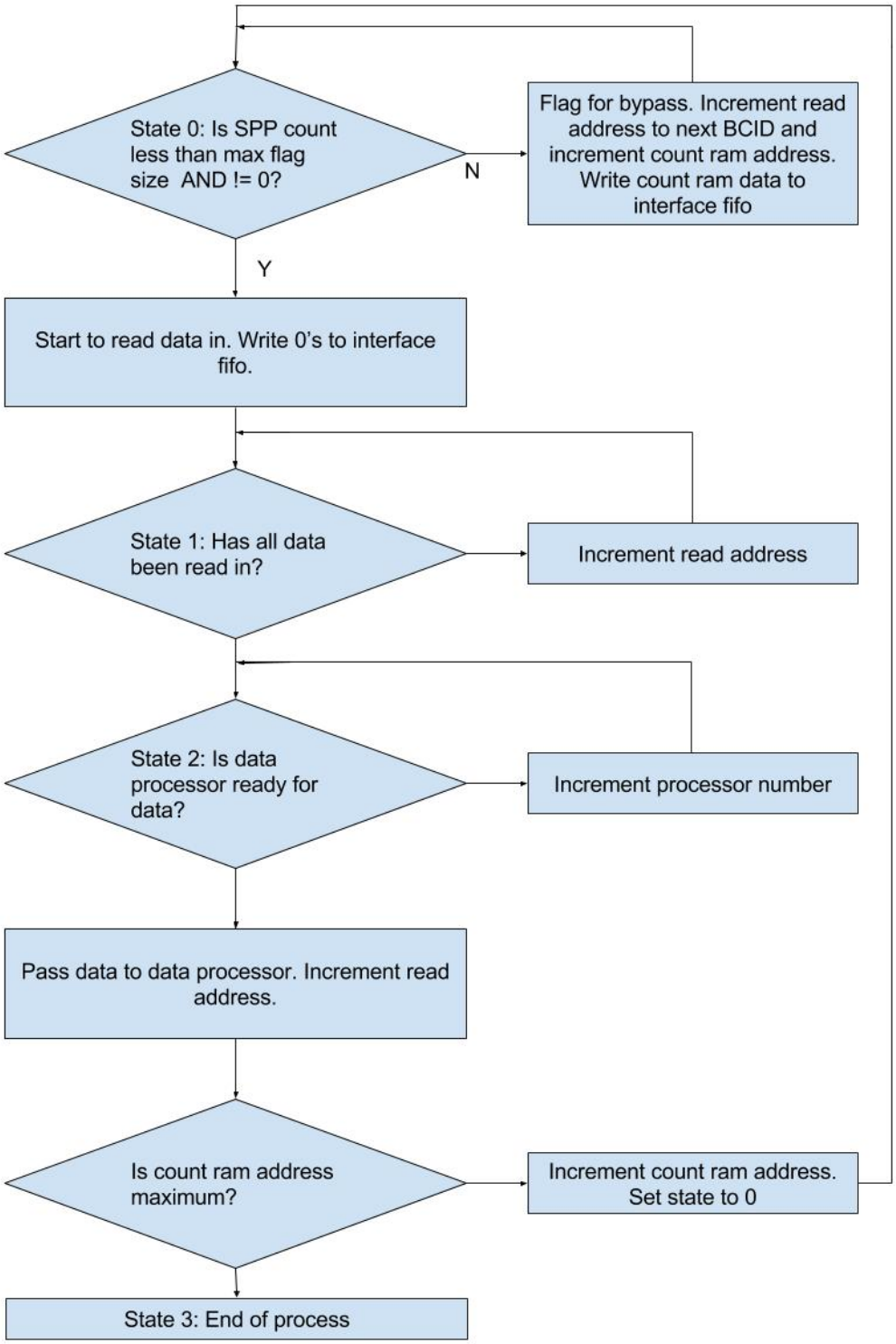

Figure 18: Flow diagram for the read process of the active control entity

process takes in the data from a read address, which consists of multiple SPPs. It splits up the SPPs, pads the SPPs with 0s in the most significant bits, and puts into an array.

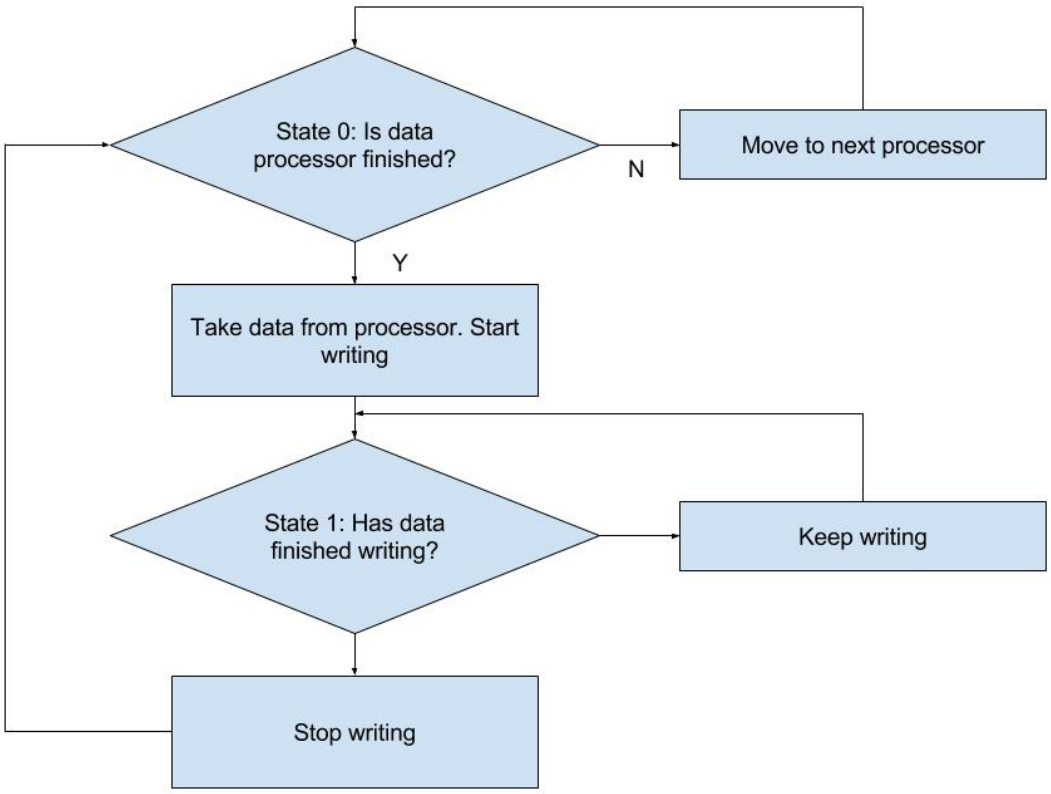

Figure 19: Flow diagram for the write process of the active control entity

The empty slots in the data train array are filled with 0s. The reason for using a statically sized data train array is simplicity, and reducing possibility for errors.

The read process and write process are described in Figures 18 and 19 respectively.

Read process The read process starts with reading the count RAM information and deciding if the BCID is going to be bypassed or not. If the data is going to be bypassed, the count RAM information is written to the interface FIFO (see Section 5.4.5), so the bypass unit in the top level isolation_flagging know which BCID needs bypassing. If the data is not going to be bypassed, the active control iterates over all the RAM addresses that the data for the BCID is contained in, and constructs the data train format that the data processors work with. A signal of all 0s is written to the interface FIFO, It then searches for a data processor that is ready to take data, and passes the data train to the first available one. The BCID and count RAM address are then increased, and the process starts again. If the count RAM address is 511, ie the maximum possible, all of the BCIDs have been processed.

Write process In the write process, the active control searches for a finished data processor, and when one is found the active control takes the data train from the data

processor. Once the data has been read from the data processor the active control frees up the data processor to take in new data. The active control then splits up the data train into the correct format for writing to the RAM, and begins writing. The writing continues until all the RAM addresses corresponding to that BCID have been written to. The BCID is then increased, and the active control searches through the data processors again for one that is finished flagging.

5.4.5 Entity: isolation_flagging

This is the top level that will directly interface with the various RAMs. It sends the write and read enable signals to the different RAMs, as well as carrying the data signals to the active control and bypass units. It also controls what we have called the swinging buffer. The RAMs cannot be written to and read from at the same time. To get past this, there will be 3 output rams for each isolation_flagging unit. The bypass will write to one, the active control to another, and the next block in the DAQ data processing will read from the third.

Entity: interface_fifo The interface FIFO constructs a First In First Out (FIFO) that is used to store the count information of the BCIDs that are to be bypassed. The interface FIFO is written to in the active control, and is read by the bypass control. The interface FIFO has 512 entries, each position corresponding to a BCID.

Entity: bypass_control The bypass control reads information from the interface FIFO, and if required carries out the bypass writing process. If the bypass control reads all 0s in the interface FIFO, it does nothing and moves onto the next address in the FIFO. If the bypass control reads a non-zero signal from the FIFO, the BCID corresponding that position needs to be passed through the bypass. The bypass control then reads in all of the data from the bypassed BCID of the input data RAM, and writes it out to the corresponding BCID of the output data RAM, after formatting the signal.

5.5 Time taken for process

Apart from resource and routing issues within the FPGA, the other main issue is timing, as in will the entire isolated flagging process have time to finish before the router block has overwritten the data in the BCID. The router can only write one SPP at a time, and only has time for a maximal of 2048 per RAM. This means that there are 2048 clock cycles available for the isolated flagging process to complete one RAM. The maximal time that it takes to process a RAM with one data processor occurs in the case that these 2048 SPPs are split evenly, 64 per BCID. The time taken for completely processing one RAM is then:

$$32 * (1 + 4 + 64 + 1 + 4) + 32 = 2400\ clock\ cycles.$$

Where:

- 32 = number of BCIDs per RAM = time taken for bypass to look at each address and see if needs bypassing
- 1 = time taken to make decision about bypass = time taken to flag data as isolated or not
- 4 = time taken to read 64 SPPs = time taken to write 64 SPPs
- 64 = time taken to sort 64 SPPs

This can be reduced by increasing the number of data processors. The processors can sort in parallel, but only one can read from the RAM at once. If there were 16 data processors, by the 17th BCID, the first data processor will have finished. The RAM is tied up for 5 clock cycles at a time, 1 for making the bypass decision, 4 for reading in data. After receiving data, the processors can all sort at the same time. The active control would take 16 * 5 = 80 clock cycles to reach the 17th RAM. The first data processor would always be finished in 74, clock cycles including read time in this worst case scenario, making it available for the 17th BCID. This reduces the total time for the isolated flagging process to:

$$32 * 5 + 69 + 32 = 261 \; clock \; cycles.$$

As the sorting is happening in parallel, the total time now equals the time taken to read in and decide whether to bypass the 32 BCIDs (32*5), plus the processing for the last BCID (69), plus the 32 clock cycles for the bypass to check.

This means that one isolated flagging unit with 16 data processors would be able to work through 9 RAMs. So two isolated flagging units with 16 data processors could be given half of the RAMs each, and there would be enough time to process all of the RAMs before they are overwritten.

5.6 Future Work

In order to be implemented into the wider code the isolation flagging block first needs to be tested. The testing needs to work out whether there are enough resources available for the number of units needed, and whether the tracks can be routed properly onto the FPGA without problem.

The issue of how changing the number of processors will impact on resources and routing will have to be investigated, and if a compromise can be found between routing and resources and having enough processors to be able to process the RAMs in time. If the amount of resources needed is too large the method may have to be changed, or the isolated clusters block may not be included in the final DAQ path. The unit also needs to be connected to the RAM units.

If any of the numbers associated with the size of RAM, read speed etc are changed, these will need to be changed in the detector constants file. They do not need to be changed in the individual entities, as these are coded using the constants in the constants file.

6 Conclusions

6.1 Scrambling analysis

The analyses of numbers of transitions, balance of 1s and 0s and the distribution of chain length show that all the scramblers improve these measures compared to unscrambled data, and thus decrease the risk of the generated clock in the DAQ becoming desynchronised. The scramblers all perform similarly for the balance and chain length analyses, however both of the multiplicative scramblers perform better than the additive scrambler in terms of transition count. The VeloPix scrambler will be used in implementation, as this is the scrambler designed to meet the needs of the ASIC, specifically that it uses less combinational logic.

6.2 Isolated Event Flagging

The isolated event flagging unit has been fully constructed. The next stage will be testing off the unit and a full analysis of the resource usage. The viability of the unit will be assessed, and whether any changes are needed to make the unit usable. If the unit is viable, then it will be immplemented into the full DAQ data processing code.

7 Acknowledgements

Scrambling analysis work was shared with Nicholas Mead and Toby Nonnenmacher. Isolated event flagging work was shared with Nicholas Mead. Many thanks to Dr. M. Gersabeck and Dr. P. Rodriguez for their help and guidance.

8 Risk Assessment

The project work took place between September 2015 and May 2016. The work took place in the Data Aquisition Lab on the 5th floor of the Schuster Building at the University of Manchester. The risk assessment shown in Table 3 was performed September 2015 at the start of the project and was reviewed in February 2016.

Risk	Potential Harm	Severity	Actions
Trips, slips and falls	Tripping over bags and cables	Low	Keeping workspace clear
Electricity	Electrocution	Low	Ensuring all electricals plugged in correctly and have been tested by University electricians.
Water damage	Water spillage onto equipment	Low	Being aware of liquids brought in. Keeping liquids as far away as practical from electrical equipment.
Display screen equipment	Eye strain	Low	Regular short breaks when using screens for long period.
Long periods of siting	Back and neck strain	Low	Retaining good posture in seat and taking regular breaks to stand and stretch out.

Table 3: Identified risks and proposed measures.

References

[1] CERN Communication Group. *LHC, The Guide*. January 2008.

[2] A. D. Sakharov. Violation of CP Invariance, c Asymmetry, and Baryon Asymmetry of the Universe. *Pisma Zh. Eksp. Teor. Fiz.*, 5:32–35, 1967. [Usp. Fiz. Nauk161,61(1991)].

[3] K. A. et al Olive. Review of Particle Physics. *Chin. Phys.*, C38:090001, 2014.

[4] Schematic of LHCb detector. `https://en.wikipedia.org/wiki/LHCb\#/media/File:Lhcbview.jpg`, Accessed January 2016.

[5] P. La Rocca and F. Riggi. The upgrade programme of the major experiments at the Large Hardron Collide. *Journal of Physics: Conference Series 515*, 2014.

[6] LHCb Collaboration. LHCb VELO Upgrade Technical Design Report. *Tech. Rep. CERN-LHCC-2013-021k*, 2013.

[7] T. Poikel et al. VeloPix: the pixel ASIC for the LHCb upgrade. *Journal of Instrumentation 10*, 2015.

[8] M. Gersabeck et al. FPGA data processing elements for the LHCb VELO upgrade. *LHCb-INT-2015-045*, 2015.

[9] R. J. Barlow. *Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences*. Wiley, 1 edition, 1989.