May 23, 2017

Isolated cluster flagging algorithm for the LHCb vertex locator upgrade

Dónal Murray[1]
Supervised by Prof. Chris Parkes[1] and Dr. Marco Gersabeck[1]

[1] *University of Manchester, Manchester, United Kingdom*

Abstract

An algorithm is presented for the flagging of isolated clusters during online data readout in the Phase-I LHCb vertex locator upgrade scheduled for the 2018 LHC long shutdown. A timing-constant solution was designed for implementation in the firmware for the data acquisition hardware. Incoming superpixel packets containing hitmap information from each bunch crossing are sorted spatially using a bubble sort algorithm before being checked for hits in neighbouring packets. Isolated packets are flagged for bypass, thereby reducing the load on the trigger when performing event reconstruction. The design has been tested with randomly-generated superpixel packets and is ready for further testing and implementation.

Contents

1 Introduction

The Large Hadron Collider (LHC), shown in figure 1 is a particle accelerator built in an underground chamber at the CERN accelerator complex on the border between Switzerland and France. The LHC is a proton-proton (pp) accelerator (which also has heavy ion runs with elements such as lead) with a circumference of 27 km, which makes it the largest particle accelerator in the world [1]. The LHC itself is not the only particle accelerator at CERN, it is connected to a chain of accelerators, also shown in figure 1. Each of these has a part to play in the acceleration of the particles to the full centre-of-mass energy. Initially atoms are stripped of their electrons using an electric field before being passed on to the first accelerator in the chain, which is Linac 2 for protons or Linac 3 for heavy ions. From here, protons are passed to the Proton Synchrotron Booster (Booster), the Proton Synchrotron (PS) and then finally arrive in the Super Proton Synchrotron (SPS) which accelerates the protons to their highest energy before entering the LHC [2]. The LHC is filled with the protons from the chain of accelerators which now have an energy of about 450 GeV. Two beams are created within the LHC travelling in opposite directions around the ring. The beams are gradually accelerated from their initial energy of 450 GeV until they reach their maximum centre of mass energy of 13 TeV [3].

Once the LHC has been filled and the beams have stabilised, bunches of protons are brought to collision at four points around the circumference of the ring, where the four biggest experiments are based.

The four main experiments are: A Toroidal LHC ApparatuS (ATLAS) experiment, the Compact Muon Solenoid (CMS) experiment, A Large Ion Collider Experiment (ALICE) and the LHC beauty (LHCb) experiment. The ATLAS [5] and CMS [6] experiments are both general-purpose particle detectors both of whom found evidence for the Higgs

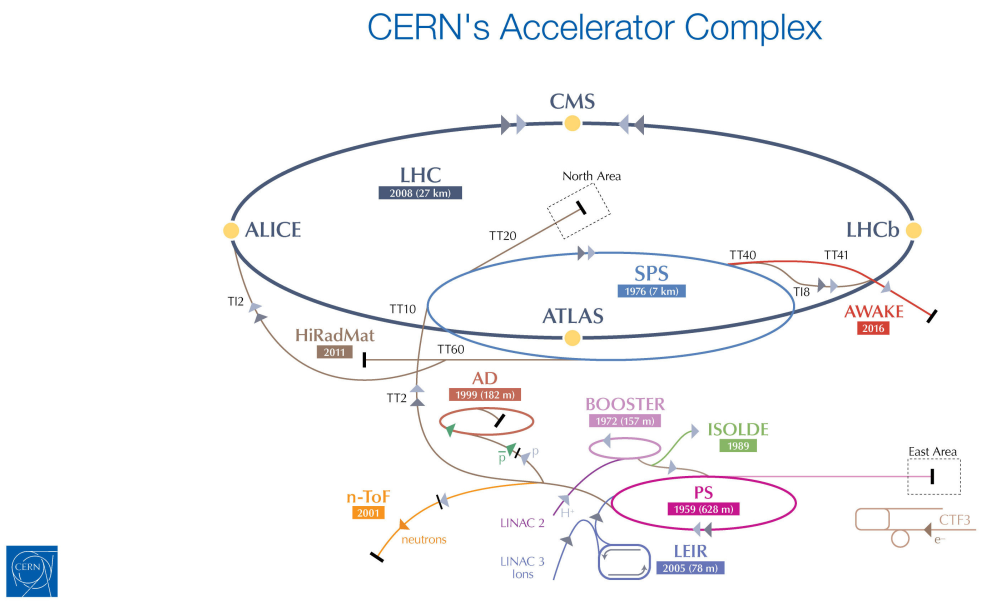


Figure 1: CERN accelerator complex, showing the LHC with its four main experiments and the chain of accelerators which produce the proton (or heavy-ion) beams. The path along which a particle will travel on its way from Linac 2 or 3 to the LHC can be seen. Adapted from [4].

Boson at an energy of 125 GeV in 2012 [7, 8]. ALICE is a general-purpose heavy-ion experiment which is primarily investigating QCD phenomena such as the quark-gluon plasma – deconfined quarks at high temperature [9]. Finally, LHCb is primarily interested in b and c physics and is investigating charge parity violation and rare decays. The LHCb experiment is discussed in more detail in section 2.

2 LHCb Experiment

The universe is currently dominated by matter despite matter and antimatter being present in equal amounts at the time of the Big Bang [10]. Upon coming into contact with each other, matter and antimatter annihilate producing photons. The Sakharov conditions describe the conditions necessary to go from having matter and antimatter in equal amounts in close proximity to each other to having only matter and photons left. One of the Sakharov Conditions is that charge parity (CP) violation must exist, which is the difference in behaviour of matter and antimatter, leading to a production asymmetry in favour of matter [10]. CP violation has been observed at LHCb and other particle detectors around the world in decays including B_s^0 but although it has been observed, the quantity of CP violation observed is not enough to satisfy the second Sakharov Condition [11]. This suggests that there is some other process, perhaps outside the Standard Model, which accounts for the rest of the required CP violation. The LHCb experiment was designed to observe the decays of particles containing b quarks, called B mesons and particles containing c quarks [12], to observe CP violation and rare decay in a bid to search for physics beyond the Standard Model, the current best description of the fundamental particles and three of the four fundamental forces [13]. The LHCb detector is well suited to performing precision tests of CP violation, with its ability to locate and distinguish secondary vertices very close to the primary vertex, enabling the decay of particles with short lifetimes, like the B mesons, to be observed [**?**].

Important and notable observations made at LHCb include:

- First observation of CP violation in the decay of the B_s^0 with the observation of the $B_s^0 \to K^-\pi^+$ decay channel [14];
- Largest observation of CP violation in the $B^\pm \to h^+h^-h^\pm$ decay channel of the charmless charged B meson [15, 16];
- The joint observation of the $B_s^0 \to \mu^+\mu^-$ rare decay with the CMS experiment [17].

2.1 Detector

The LHCb detector [19, 20] is a single-arm forward spectrometer, designed for the study of particles containing b or c quarks. The detector consists of several subdetectors, shown in figure 2. The two proton beams collide inside the vertex locator (VELO) [21], seen at the left of figure 2, which is a silicon-strip tracking system. When protons collide at energies as high as at the LHC, a shower of particles is created, some of which are B mesons. The B mesons are identified as particles that decay a distance away from the collision vertex that corresponds to the distance a B meson would travel in its lifetime. It falls to the VELO to identify the primary vertex and secondary vertex to enable the

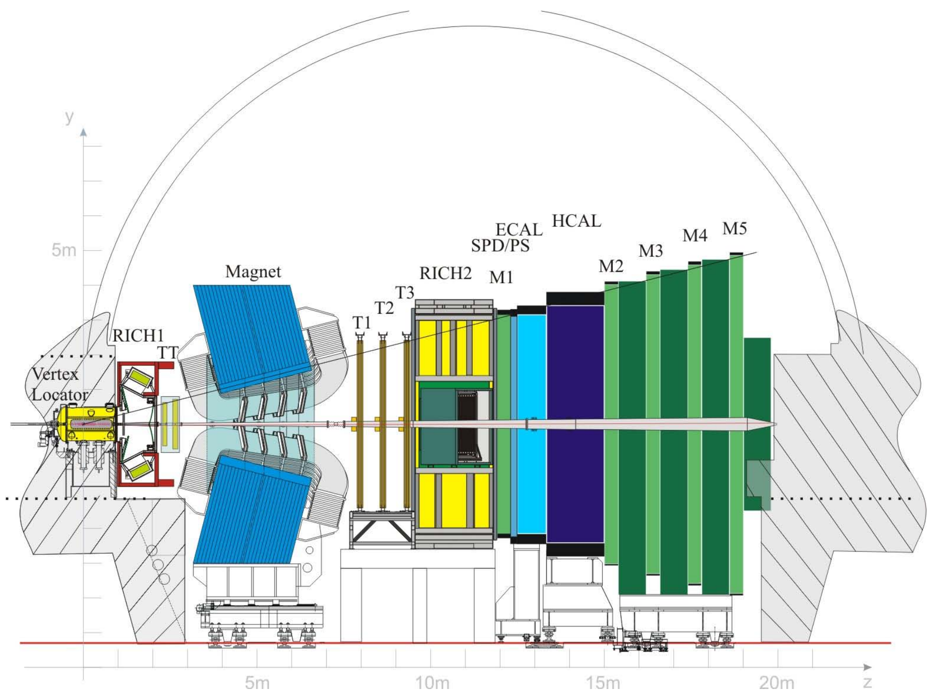


Figure 2: A side view of the LHCb detector showing the subdetectors. Taken from [18].

distance between them to be found. The VELO is discussed in more detail in section 4. Moving from left to right, a brief description of the each relevant part of the detector follows. The Ring Imaging Cherenkov detectors (RICH1 and RICH2) [22] determine the velocity of the decay products by measuring the ring caused by the light cone emitted due to the Cherenkov effect. The Cherenkov effect causes charged particles to emit a light cone if they are travelling through the detector medium faster than the speed of light in this medium [23]. The particles are deviated by the magnet in a direction dependent on their charge and three silicon-strip detectors and straw drift tubes (T1, T2, T3) [24] work together with the silicon-strip detector to the left of the magnet (TT) to determine the momentum of charged particles from the curvature of their path. Photons, electrons and hadrons are identified by a calorimeter system consisting of scintillating-pad and preshower detectors (SPD and ps respectively), an electromagnetic calorimeter (ECAL) and a hadronic calorimeter (HCAL). Muons are identified by the five muon detectors (M1 to M5) [25] at the far right of the figure. At this stage only muons and neutrinos are left, as the other decay products will be stopped by the iron filters between the calorimeters and muon detectors. A neutrino passes straight through as it has no charge while a muon's charge is detected by its effect on a mixture of gases within the muon detector. The trigger [26] selects events with low transverse momentum, which are particles that stay within a narrow angle of the primary interaction vertex, a trait associated with B mesons, and through this selection decreases the number of events which need to be reconstructed in the software stage.

3 LHCb upgrade

Due to the high levels of radiation involved in the running of the LHC, the parts of the LHC and the detectors wear out over time, reducing their effectiveness and sensitivity. To prevent the signal-to-noise ratio decreasing below an acceptable standard, and with the added benefit of being able to take advantage of new technologies, a schedule of upgrades has been decided for the collider and for the experiments. Between when collisions first started at the LHC in 2009 and 2030, three long shutdowns (LS) were planned during which the collider could be upgraded [27]. During LS1, which took place from 2013 until 2015, the superconducting magnets and other parts of the detector were upgraded to allow for the full beam energy of $6.5 - 7\,\text{TeV}$ and the nominal luminosity of $1 \times 10^{34}\,\text{cm}^{-2}\,\text{s}^{-1}$ to be reached [27]. During LS2, which is scheduled to take place from 2018 until 2021, upgrades to the beam injector and LHC will be completed to allow double the nominal luminosity to be achieved. The third major upgrade is scheduled to take place during LS3 and will be the start of the High Luminosity LHC (HL-LHC), which will include a $5 - 7$-fold increase in luminosity over the nominal luminosity [27].

As part of these scheduled upgrades to the LHC, the LHCb collaboration will also be upgrading the LHCb detector during LS2 to make use of the 2-fold increase in luminosity, which will allow for more events to be observed. Another major upgrade to LHCb was proposed in February 2017 to be carried out during LS4, planned for 2030. The proposed upgrade will take advantage of HL-LHC, with a planned capability to operate at ten times the luminosity possible with the upcoming Phase-I upgrade being carried out during LS2 [28].

After the Phase-I upgrade, the LHC will reach a peak luminosity of $2 - 3 \times 10^{34}\,\text{cm}^{-2}\,\text{s}^{-1}$ [3], however the LHCb Phase-I upgrade will run at a luminosity of $2 \times 10^{33}\,\text{cm}^{-2}\,\text{s}^{-1}$ [29]. As part of this upgrade, all of the LHCb subdetectors are being upgraded, including the vertex locator (VELO), which is discussed in more detail in section 4.

4 VELO upgrade

The LHCb VELO is currently the most precise vertex detector at the LHC but currently has a maximum readout frequency of 1 MHz [30]. The Phase-I upgrade plans to increase the readout frequency from 1 MHz to 40 MHz to allow for faster data acquisition and processing. The ability to obtain data at a high rate minimises the number of events that are missed and reduces the errors on measurements and observations due to the statistical nature of the errors.

An increase in readout speed leads to the possibility to use an increased luminosity while still maintaining the current performance, Phase-I VELO will operate at a luminosity twice the magnitude of the current detector [29].

The upgraded VELO consists of an array of 52 modules, shown in figure [?]. Each module has four pixel sensors which can be seen to the right, the constraint system can also be seen which supports the pipes feeding the integrated microchannel evaporative carbon dioxide cooling system. The pixel sensors will be moved to a position only 5.1 mm from the LHC beam once the beam has been stabilised, therefore the materials that the pixel sensors are made from must be able to withstand the radiation levels present which are expected to increase by a factor of 10 with the increase in luminosity [29].

The upgraded VELO will consist of 52 of these modules, arranged in two rows of 26 modules facing each other in such a way that the pixel sensors on the end of each module will tesselate with the pixel sensors on the corresponding module on the opposite side to ensure complete coverage and avoid loss of data, two sensors will overlap to ensure transverse tracks are not lost [29].

5 Data Readout

The LHCb VELO needs an entirely new readout system for the new hardware and planned increased speeds otherwise the luminosity will not be able to be used effectively. Without fast readout, tracks cannot be distinguished from one another accurately. The data readout system uses a combination of purpose-built ASICs (application-specific integrated circuits), which are discussed further in section 6 and versatile programmable integrated circuits called field programmable gate arrays (FPGAs) [29].

5.1 FPGA development

Field programmable gate arrays are versatile integrated circuits that can be programmed to suit a wide range of applications. The chips themselves are just an array of thousands of logic gates which can be combined using hardware description languages to accomplish the same task as ASICs. They are capable of running multiple processes at very high clock speeds and also are able to run processes at different clock speeds within the same chip. The FPGA used for the VELO upgrade is the Altera Stratix V GX, which is capable of an internal clock speed of 160 MHz [33]. The TELL40 data readout board, seen in figure 4, will be used to read data from the ASICs using FPGA chips, at least 78 of which will be required for the final VELO to be capable of dealing with the increased data throughput [29].

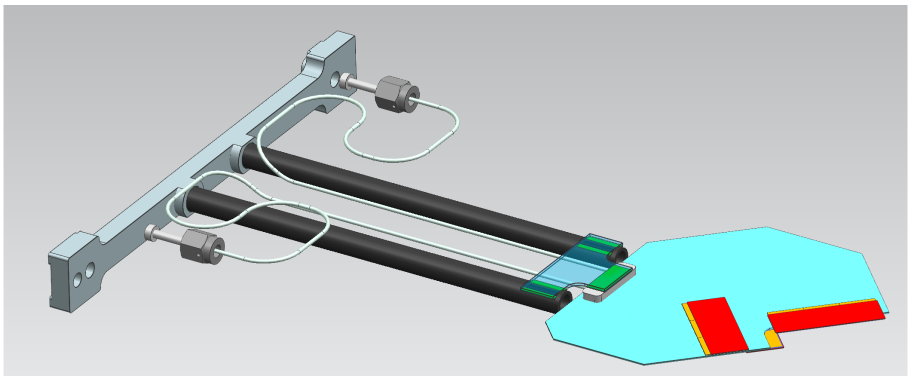


Figure 3: Computer aided design model of one of the 52 VELO modules. The substrate is shown as light blue (light shaded), the sensors as red (dark shaded), and the carbon dioxide cooling pipes as grey (medium shaded). Taken from [31].

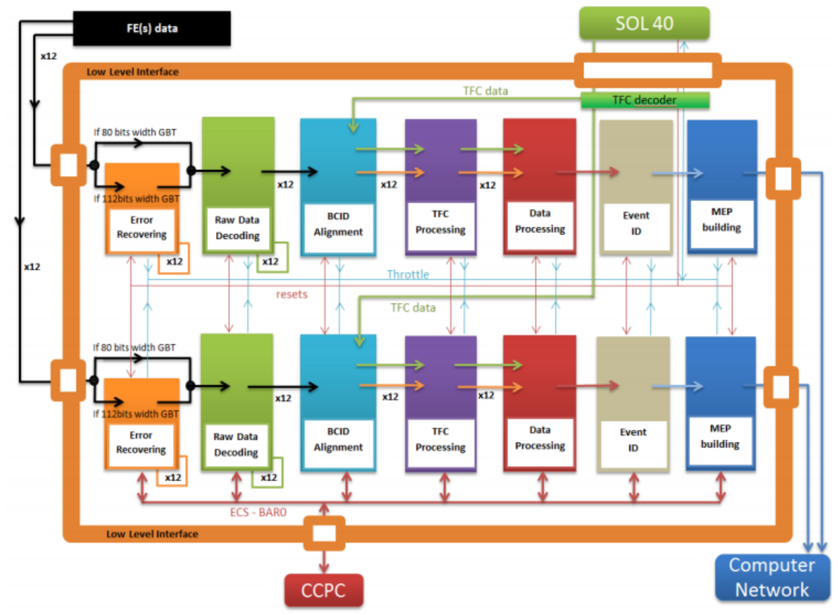


Figure 4: Diagram showing the data readout chain with the direction of flow of data from the VELO modules. Taken from [32]

5.1.1 VHDL and simulation tools

VHDL is a hardware description language used to program FPGA chips. Through defining virtual components with ports, internal signals and processes, extremely complex circuitry can be built with the versatile FPGA chips.

Modelsim [34] is a multi-language simulation environment which allows for VHDL code to be compiled and simulated. The waveform window shows the signals and their values. By issuing commands on the built-in interpreter, signals can be set and scheduled to take on a specified value at a specified time. The process can be automated using the TCL programming language [35] which quickens the process significantly.

VHDL runs multiple processes concurrently and therefore can complete tasks much more quickly than sequential programming languages. For this reason as much of the data processing as possible will be done during the hardware stage (using the FPGAs) to reduce the load on the LHCb event reconstruction farms.

6 Isolated Cluster Flagging

6.1 Aims

Each pixel sensor is made up of an array of pixels of semiconductor measuring 55 x 55 μm. These pixels are grouped together into so called superpixels (SPPs) which are 2x4 arrays of 8 pixels. This is to avoid redundant data from events hitting multiple neighbouring pixels, which is quite likely (55% of events) due to the small pixel size. The SPP concept is predicted to reduce bandwidth by up to 30% [29]. Since each ASIC measures 256 x 256 pixels, the resulting system of superpixels has 128 rows and 64 double columns per

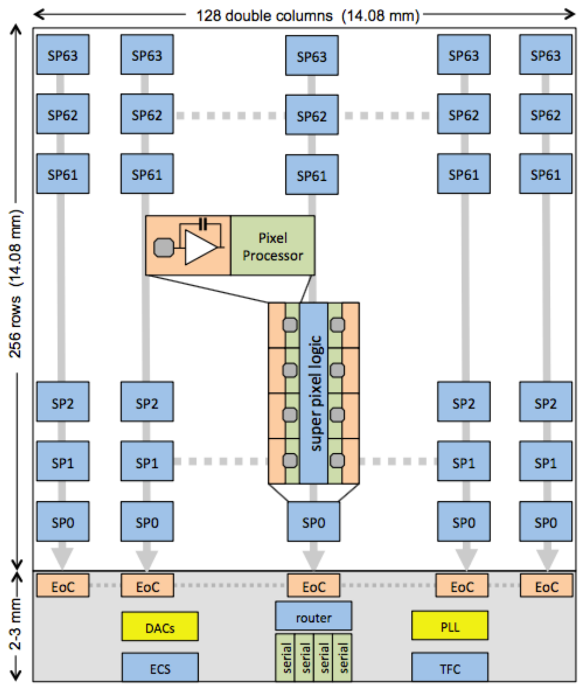


Figure 5: Superpixel layout on the ASIC, which makes up the pixel sensor of the module. The readout system is in the middle of the 256 x 256 pixel array which is organised into the superpixels. Taken from [29]

ASIC as shown in figure 5. When a particle hits one of the superpixels, a superpixel packet (SPP) is created, containing a timestamp and hitmap corresponding to the event. However, sometimes a hit will occur at the boundary between two or more superpixels, meaning that more than one SPP will be created for the same event and the data for that event will be split across these SPPs. This requires the CPU to check each SPP and its neighbours to see if any of the neighbouring packets have the same bunch crossing ID, which would require the CPU to combine the hitmap information from both SPPs in the software stage. This is very resource intensive and so to reduce the load on the CPU, the SPPs will be checked in the hardware stage within the FPGA to find SPPs from the edge of the column and also SPPs which have no neighbouring SPPs with hits.

6.2 Concept

At the stage of the data readout chain that the ICF block is inserted, the SPPs are in a stripped-down form of 24 bits, containing only the row ID, column ID and ASIC chip number, which relate to the ASIC structure shown in figure 5 and the hitmap. The 24-bit SPPs are grouped together in 384-bit datatrains, each containing 16 SPPs.

It was found to be computationally beneficial to make a cutoff point of 64 SPPs processed per BCID, which equates to a maximum of four of these 384-bit datatrains. Any BCIDs with more than 64 SPPs are bypassed without being sorted and flagged to keep the timing constant.

6.2.1 Bubble sorting

Bubble sorting is a method of sorting SPPs by making two types of pass which compare pairs of SPPs: odd and even passes. If the SPPs are numbered from 0 to n, an even pass compares object 0 with object 1, object 2 with object 3 and so on, whereas an odd pass compares object 1 with object 2, object 3 with object 4 and so on. When the SPPs are compared, if the column ID of the higher numbered SPP is greater than the lower one of the pair then they are switched. In this way the SPPs will be ordered in n^2 passes. However, since the FPGA can execute n processes concurrently, the sorting is finished after n passes.

6.2.2 Flagging

Once sorted, the SPPs are in the order of the superpixels whose data they carry. SPPs are only created when a particle interacts with the superpixel in question, therefore an isolated SPP is one whose neighbours in the sorted array are more than one column away from it. The most significant bit (MSB) of the SPP is the flagging bit, but first the 24-bit SPPs need to be padded out to 32-bits and the most significant bit (MSB) is set to 1 if the SPP is found to have no SPPs in adjacent columns. Again, due to the concurrent nature of FPGAs, all of the isolated clusters can be checked by the flagger at the same time, in just one clock cycle.

6.2.3 Top level

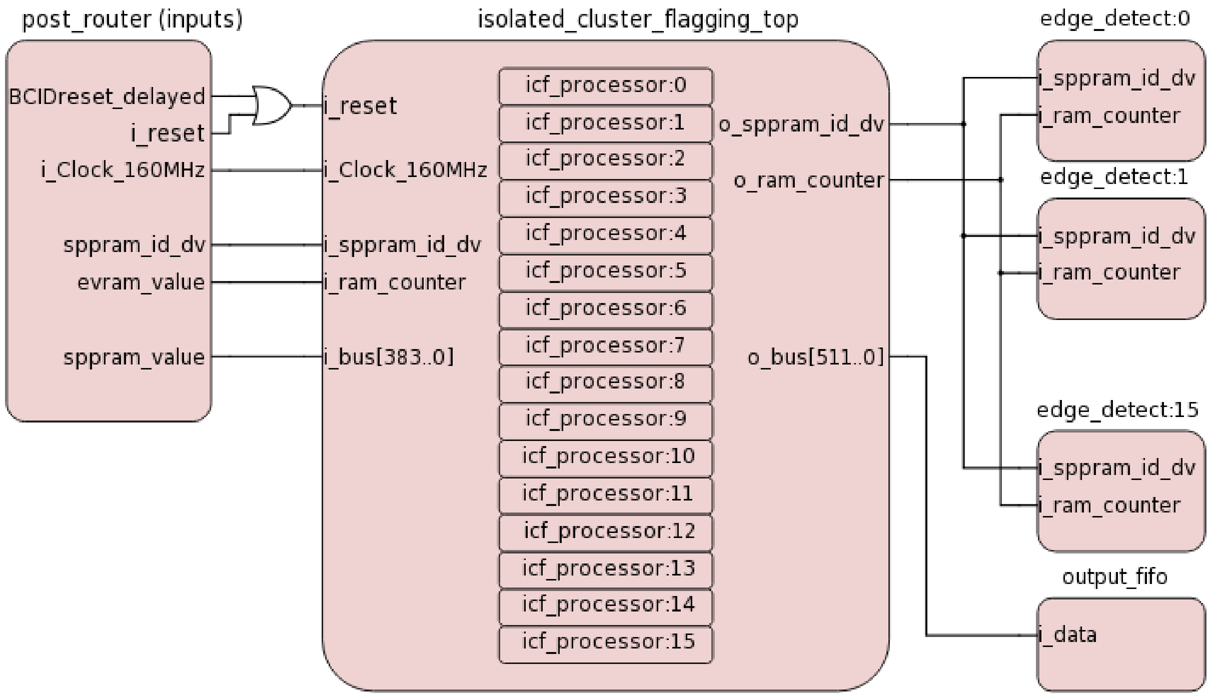


Figure 6: Top level block diagram of the ICF block. The ICF block is shown in position, with the inputs from the post-router and outputs to the 16 edge detectors and output FIFO. Created using [36].

6.2.4 Processor

The ICF processor coordinates the sorter and flagger by making use of an 8-bit counter and the concept of a state machine. A state machine is a way to overcome one major issue of concurrent programming – the chance that two instances of a process will simultaneously attempt to perform the same task or change the same variable, resulting in undefined behaviour. By using four states, this issue has been resolved.

In state `s0`, the processor waits for its enable to be driven high. When the enable is driven high, the counter is started and then on the rising edge of the next four clock cycles, a datatrain is read in, 8 zeroes are prepended to the 24-bit SPPs and they are passed into an array. Once all four datatrains have been read in, the array containing up to 64 32-bit SPPs is ready to be sorted. The state is then changed to state `s1`: the sorting stage.

In state `s1` the SPP array is passed to the sorter which performs one iteration of the bubble sort algorithm and returns the answer. The processor passes the output of the sorter back in as an input 63 times, at which stage the SPP array will be sorted in order. The state is then changed to state `s2`: the flagging stage.

In state `s2` the sorted SPP array is passed to the flagger which checks for isolated clusters using the method described in section 6.2 and flags any SPPs which are isolated before changing to state `s3`: the write out stage.

In state `s3` the sorted array of flagged SPPs is deconstructed into four 512-bit datatrains and written out.

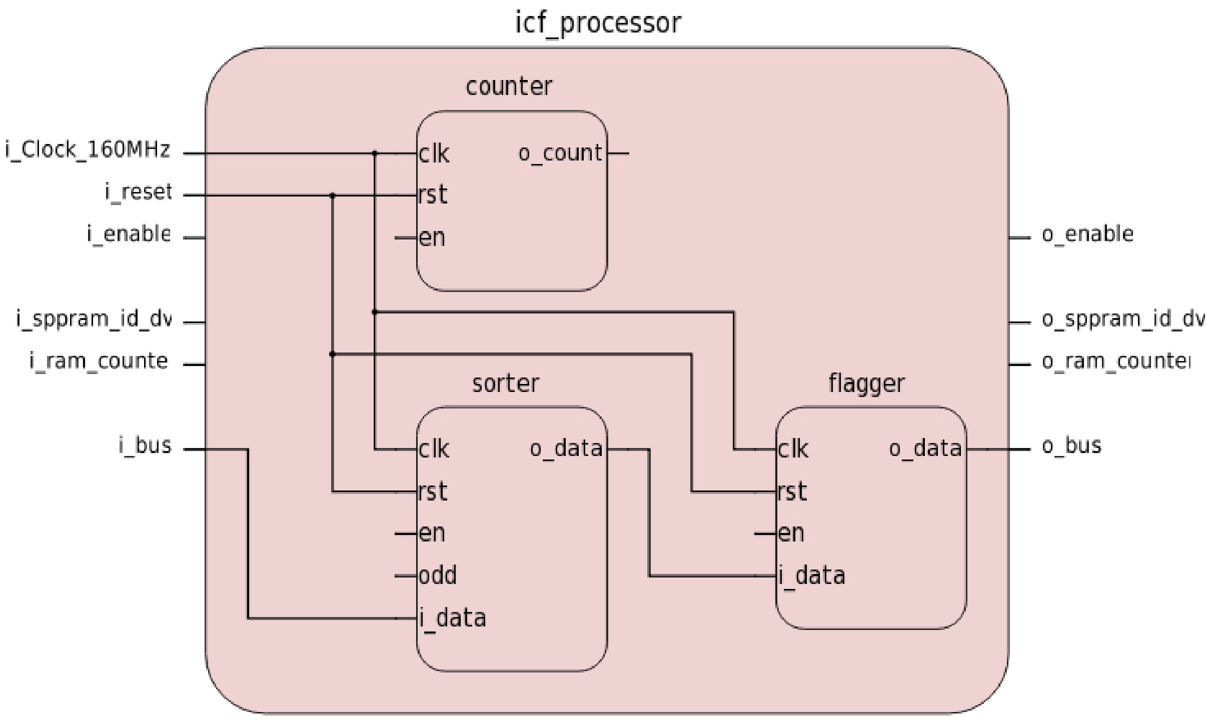


Figure 7: ICF processor which coordinates the sorter and flagger by making use of an 8-bit counter and the concept of a state machine. The flying legs of the lower level blocks are driven by internal signals in the data processor. Created using [36].

6.3 Timing

Each data processor within the ICF block needs the following number of clock cycles per BCID:

- 4 clock cycles to read in the 4 384bit frames associated with the BCID 65 clock cycles to sort the columns and flag isolated clusters 4 clock cycles to write the four datatrains out again

This equates to a final count of 73 clock cycles for the entire process. However, it was found in testing that due to the way the state machine transitions between states, 79 clock cycles are actually required, with the second processor fill beginning on the 80th clock cycle.

The BCID processing (sorting and flagging) is parallelised within the ICF block between 16 processors and in the time taken to write to all of the other 15 processors, the first processor will already be finished.

To make sure the timing is unaffected, delays to other signals will all be handled within the ICF block.

7 Testing

The aim of the testing stage was to ensure that the VHDL implementation matched the design. This is important when working with VHDL as it is often difficult to anticipate what order operations will happen in as some operations happen on a falling edge of the clock or after a specified delay and others happen immediately on a rising edge of the clock. By testing each block extensively in Modelsim [34] any ambiguous or unintentional ordering can be identified and fixed.

Throughout this section numbers in three bases are used, as each has a relevance in different contexts. To avoid ambiguity, binary (base 2) numbers are prefixed by the letter b and hexadecimal (base 16) numbers are prefixed by the letter h, while decimal (base 10) numbers are not given a prefix. The convention of using the letters A–F to represent 10–15 is used for hexadecimal. Additionally, zeroes are prepended to binary and hexadecimal numbers to indicate the bit length of a signal. As an example: b00001111, h0F and 15 are all equivalent labels for an 8-bit signal.

Programs were created using C++ to generate random data in the correct format for each block to be tested. The TCL [35] scripting language was used to automate signal allocation to inputs at a specified point in time. The programs written in C++ generated data already in the formate of TCL scripts such that they could be loaded directly into Modelsim with no manual input, resulting in a with the intended to simulate the normal working conditions of the unit.

All SPPs generated by the data generation programs were given a hitmap of b00000011. The value of the hitmap was free to be chosen for testing as it is not used by the ICF block. The value b00000011 was chosen for testing purposes as it is easily recognisable both in binary and hexadecimal (h03) and is distinguishable from the other end of a flagged SPP which is b10000000, to make it easy to ensure that the SPPs being read in the correct way around – MSB to LSB – as opposed to LSB to MSB.

7.1 Counter

The counter did not require any data to be generated, a simple TCL file was used to set the clock frequency and enable the counter after the initial reset.

7.2 Sorter

The program used to generate data for testing the sorter used a random number generator to generate a random value between 0 and 255. This decimal number was converted to binary to represent the column and chip IDs in the 8 bits from bit 23 down to bit 14, these were padded with zeroes to form a 32-bit SPP. The SPPs are then output to a file in groups of 16 to simulate the 384-bit datatrain coming from the router. These datatrains are then output to a file in a random order and output to a file, the SPPs then undergo one iteration of a bubble sort algorithm and are output to a different file, so that the output from the simulation can be compared with the output file to see if the VHDL code is behaving as expected.

7.3 Flagger

The program used to generate data for testing the flagger was completely analagous to the program for the sorter except the random value was generated between 0 and 63 to represent the column and chip IDs in the 6 bits from bit 13 down to bit 8. The datatrains of padded SPPs are assembled in the same way before being fully sorted output to a file.

7.4 Processor

The program used to generate data for testing the data processor had a combination of the functionality of the programs for the sorter and flagger so that a full SPP could be constructed, populating bits 23 down to 8 with random data. After prepending h00 and appending h03 to give the full 32-bit SPPs, datatrains were contructed and output to a file in groups of four. The constituent datatrains of these groups of four were one clock cycle apart, while the groups themselves were separated by 76 clock cycles. This 76 clock cycle separation was to represent the time spent filling the other processors before looping back to fill this processor after it had successfully written out its sorted and flagged data

7.5 Top level

The top level used the same generated data as the processor but instead of having groups of four datatrains separated by 76 clock cycles, there were 128 datatrains separated by one clock cycle. This allowed for two full cycles of the full program.

8 Results

This section presents the results of the tests done in Modelsim that are outlined in section 7. The lower level blocks performed well in the testing stage, functioning exactly as intended. The ICF processor block ran into several problems with the number of clock cycles required for each state but after a restructure of the state machine outlined in section 6 the wasted clock cycles were removed. The top level testing was not completed and as such the results for the tests specified in section 7 are incomplete. These are left as future work and discussed in section 9.

For all Modelsim simulations included in figures 9–13, the clock was set to 250 MHz instead of 160 MHz for ease of reference to the time while describing the waveforms in the body of this report. The former clock frequency leads to a 4 ns signal whereas the latter leads to a 6.25 ns signal. Since the graduations on the scope are in 5 ns intervals, and to reduce figure width to allow larger text in figures, it was decided that an integer value would be a more sensible choice for clarity. A possible negative side effect of this is discussed further in section 8.5.

In each test the clock was initialised by forcing the clock (`clk` or `i_Clock_160MHz`) high for half of the period and then forcing it low again. The reset (`rst` or `i_reset`) was first pulsed high at time 0 ns to initialise all of the bits in the signals and outputs to zeroes which prevented unassigned bits being passed through the block and prepares the signals for the processes.

8.1 Counter

The enable (en) was forced high after the initial reset and the counter started incrementing the output signal `o_count` on the rising edge of the next clock cycle, as shown in figure 8. The counter continued to increment while the enable was kept high, and reset the count when the enable was pulsed low. The 8-bit counter functioned as expected, with no wasted clock cycles and no unassigned bits being passed as output.

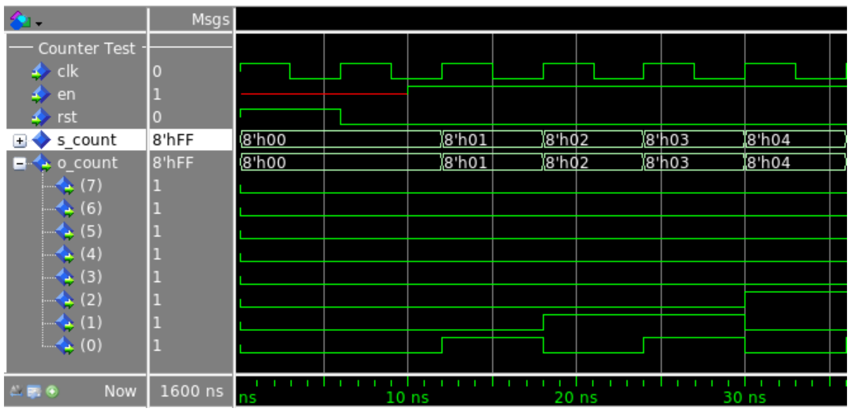
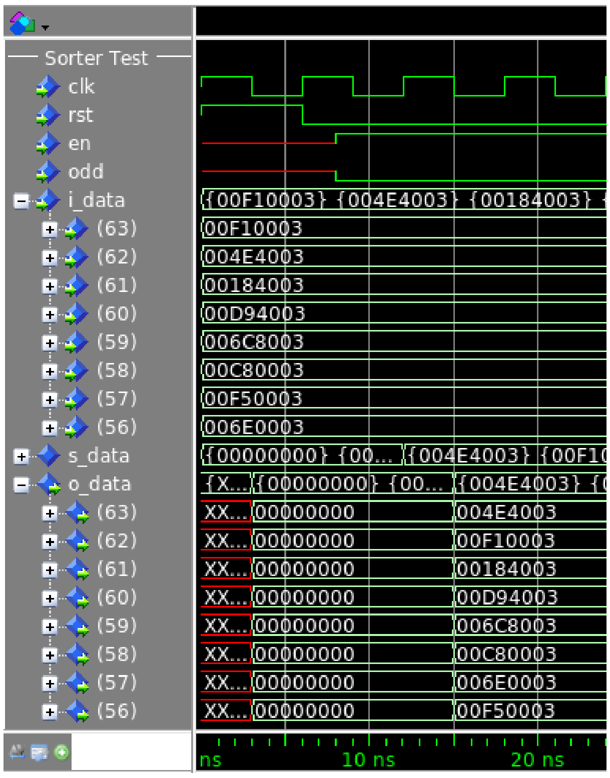


Figure 8: Modelsim simulation showing the output of the counter after forcing the input signals using a TCL script file. Simulated using [34].

8.2 Sorter

The sorter takes input on the `i_bus` signal and performs one iteration of the bubble sort algorithm described in 6.2, an even (odd) pass if `odd` is low (high). In the simulation shown in figure [?], the sorter is initially reset and then enabled at time 10 ns with `odd` forced low, initiating an even pass of sorting.

For each SPP in `i_data`, the first two hexadecimal digits represent the 8-bit zero padding, the last two represent the 8-bit hitmap and therefore it is only the four middle digits that are compared by the sorter. Therefore when comparing `i_data(63))` with `i_data(62)`, the sorter is checking which of these four digit numbers is greater. It can be seen from the figure that `i_data(63))` was switched with `i_data(62)` while `i_data(63)` and `i_data(62)` were not switched for this reason.



Sorter Test
clk
rst
en
odd
i_data
{00F10003} {004E4003} {00184003}
(63) 00F10003
(62) 004E4003
(61) 00184003
(60) 00D94003
(59) 006C8003
(58) 00C80003
(57) 00F50003
(56) 006E0003
s_data
{00000000} {00... {004E4003} {00F1
o_data
{X... {00000000} {00... {004E4003} {
(63) XX... 00000000 004E4003
(62) XX... 00000000 00F10003
(61) XX... 00000000 00184003
(60) XX... 00000000 00D94003
(59) XX... 00000000 006C8003
(58) XX... 00000000 00C80003
(57) XX... 00000000 006E0003
(56) XX... 00000000 00F50003
ns
10 ns
20 ns

Figure 9: Simulation showing the input and output of the sorter after forcing the input signals using a TCL script file and using unsorted randomly-generated data from the program described in section 7.2. Simulated using [34] and two images altered and combined to show only the top 8 array elements of i_data and o_data.

8.3 Flagger

It can be seen from the simulation in figure 10 that the only SPP to be flagged (`o\_data(56)`) is also the only one whose binary value between bit 14 and 8 is more than 1 away from the SPPs next to it in the SPP array.

8.4 Data processor

The first two states of the processor block are shown in figure 11. First the reset was pulsed high, which set all signals and outputs to 0, reset the components within the processor and set the state to s0. On the next clock cycle the enable was driven high, which started the process of reading in the datatrains and padding them with zeroes to create the 32-bit SPPs. On the first clock cycle of state s0, the counter enable was driven

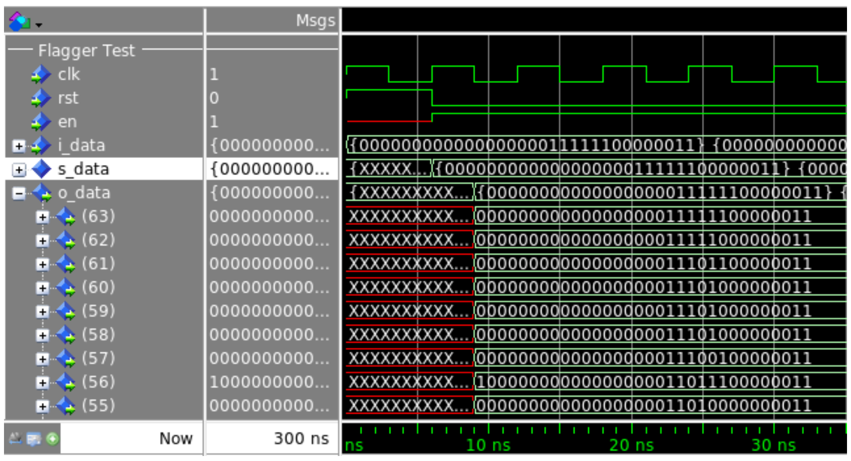

Msgs
Flagger Test
clk 1
rst 0
en 1
i_data {000000000...
s_data {000000000...
o_data {000000000...
(63) 0000000000...
(62) 0000000000...
(61) 0000000000...
(60) 0000000000...
(59) 0000000000...
(58) 0000000000...
(57) 0000000000...
(56) 1000000000...
(55) 0000000000...
{000000000000000000011111100000011} {000000000000
{XXXXX...}{000000000000000000011111100000011} {0000
{XXXXXXXXXX...}{000000000000000000011111100000011} {
XXXXXXXXXX... 000000000000000000011111100000011
XXXXXXXXXX... 000000000000000000011111000000011
XXXXXXXXXX... 000000000000000000011101100000011
XXXXXXXXXX... 000000000000000000011101000000011
XXXXXXXXXX... 000000000000000000011101000000011
XXXXXXXXXX... 000000000000000000011101000000011
XXXXXXXXXX... 000000000000000000011100100000011
XXXXXXXXXX... 100000000000000000011011100000011
XXXXXXXXXX... 000000000000000000011010000000011
Now 300 ns
ns
10 ns
20 ns
30 ns

Figure 10: Simulation showing the output of the flagger after forcing the input signals using a TCL script file. Input data was generated and spacially sorted by the program described in section 7.3. Simulated using [34].

high to start the count which keeps track of the current stage of each state. Once the four frames are read in, the SPPs are padded out and then put into an array of SPPs before the processor transitions to state s1 and passes the array of SPPs to the sorter to be sorted using the bubble sorting algorithm described in section 6.

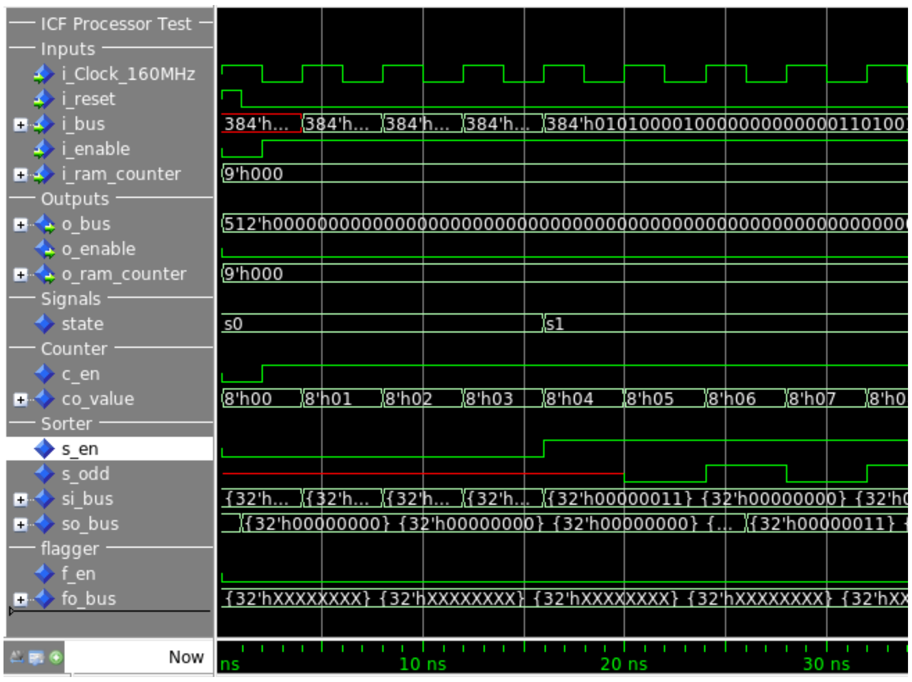

ICF Processor Test
Inputs
i_Clock_160MHz
i_reset
i_bus
i_enable
i_ram_counter
Outputs
o_bus
o_enable
o_ram_counter
Signals
state
Counter
c_en
co_value
Sorter
s_en
s_odd
si_bus
so_bus
flagger
f_en
fo_bus
384'h... 384'h... 384'h... 384'h... 384'h010100001000000000000110100
9'h000
512'h000000000000000000000000000000000000000000000000000000
9'h000
s0 s1
8'h00 8'h01 8'h02 8'h03 8'h04 8'h05 8'h06 8'h07 8'h0
{32'h... {32'h... {32'h... {32'h... {32'h00000011} {32'h00000000} {32'h0
{32'h00000000} {32'h00000000} {32'h00000000} {... {32'h00000011}
{32'hXXXXXXXX} {32'hXXXXXXXX} {32'hXXXXXXXX} {32'hXXXXXXXX} {32'hXX
Now
ns
10 ns
20 ns
30 ns

Figure 11: Simulation showing the first two states of the processor state machine.

Figure 12 shows the final two states of the processor block and the start of the next cycle of data being loaded in. State 2 can be seen starting at the rising edge of the 69th clock cycle (h45 on the co_value signal). This is the flagging process which takes one clock cycle. State 3 is then started on the next clock cycle which waits until the 76th clock cycle to deconstruct the array of up to 64 SPPs into four 512-bit datatrains, each containing up to 16 32-bit SPPs, which have now been sorted spatially and isolated SPPs have been flagged.

8.5 Top level

The first test of the top level of the ICF block is shown in figure 13. In this figure the way that the datatrains are allocated to the first two processors using each processor's enable which is connected to one of the bits of dp_i_enable. The top level process fills each processor and then when all processors have been filled, it loops back and fills the first one again.

9 Discussion and future work

9.1 Future work

Due to time constraints with the master's project, the ICF block testing was not completed for the top level block which interfaces with the edge detectors in the post-router. The ICF block top level will need to be tested with realistic simulation data to ensure that the ICF block succeeds in flagging a high enough percentage of SPPs and that there are no false positives – non-isolated SPPs being flagged as being isolated – as this would lead

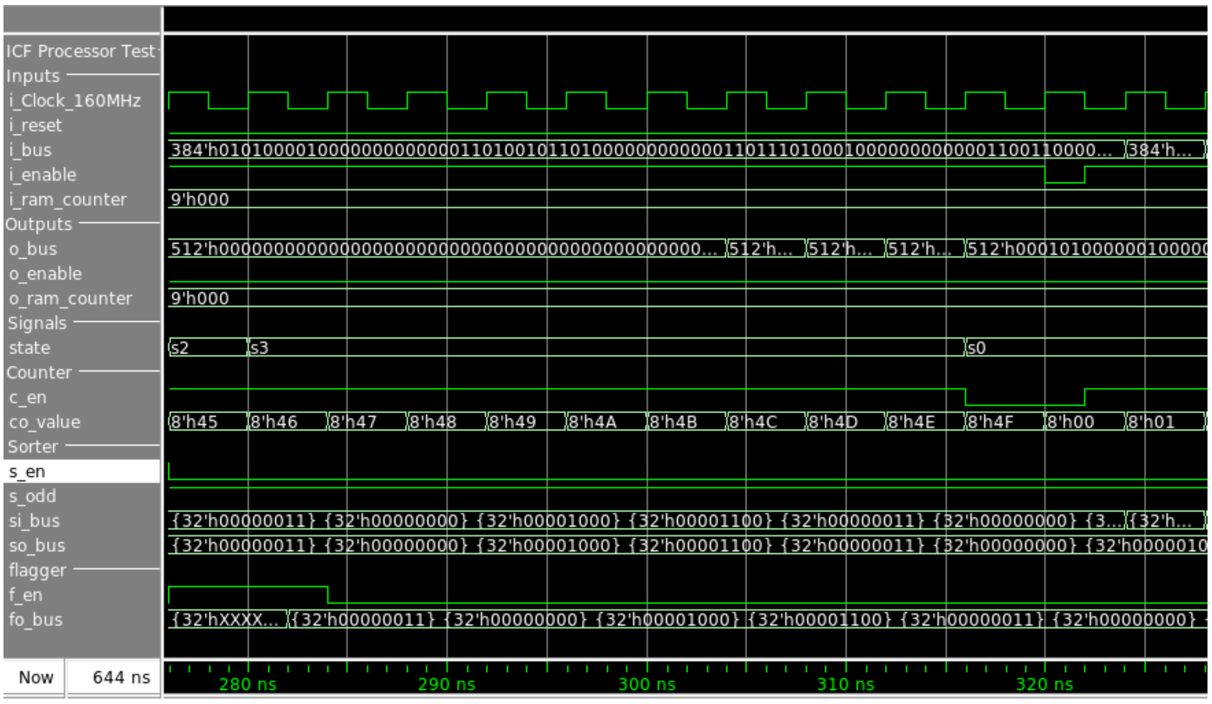

ICF Processor Test
Inputs
i_Clock_160MHz
i_reset
i_bus
i_enable
i_ram_counter
Outputs
o_bus
o_enable
o_ram_counter
Signals
state
Counter
c_en
co_value
Sorter
s_en
s_odd
si_bus
so_bus
flagger
f_en
fo_bus
384'h0101000010000000000000110100101101000000000000110111010001000000000000011001100000...
384'h...
9'h000
512'h00000000000000000000000000000000000000000000000...
512'h...
512'h000101000000100000
9'h000
s2
s3
s0
8'h45
8'h46
8'h47
8'h48
8'h49
8'h4A
8'h4B
8'h4C
8'h4D
8'h4E
8'h4F
8'h00
8'h01
{32'h00000011} {32'h00000000} {32'h00001000} {32'h00001100} {32'h00000011} {32'h00000000} {3... {32'h...
{32'h00000011} {32'h00000000} {32'h00001000} {32'h00001100} {32'h00000011} {32'h00000000} {32'h0000010
{32'hXXXX... {32'h00000011} {32'h00000000} {32'h00001000} {32'h00001100} {32'h00000011} {32'h00000000}
Now
644 ns
280 ns
290 ns
300 ns
310 ns
320 ns

Figure 12: Simulation showing the final two states of the processor state machine and the start of the next cycle of data being loaded in. Note how the next four datatrains are read in as soon as the previous data has been written out. Simulated using [34].

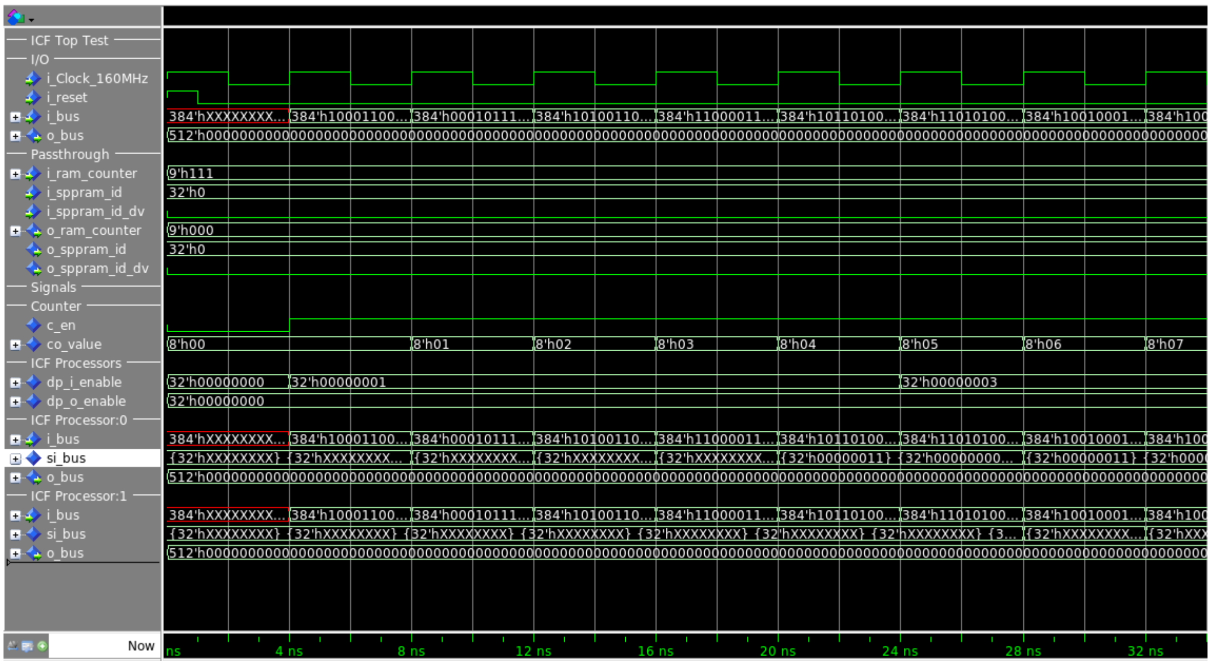


Figure 13: Simulation showing the top level and how it allocates the incoming datatrains from the router to the first two processors using each processor's enable which is connected to one of the bits of dp_i_enable. Simulated using [34].

to a loss of data, which is to be avoided at all costs.

An extension to the ICF block diagram which needs to be implemented is a check to make sure that the number of SPPs associated with each BCID is less than the current cut off figure of 64. If this is the case, the BCID will have to be bypassed after the correct number of clock cycles to keep the data in the correct order.

Once the top level is tested and is confirmed to be working to an acceptable standard, the code can be inserted into the post-router of the AMC40 firmware and has been designed to slot in and interface with the existing components with minimal changes to existing firmware files to ensure it is easy to bypass the block, which is a necessary feature for testing other blocks being developed as a full compile of the firmware in Quartus usually takes around 12 and a half hours even on a powerful desktop computer. With a wait of this length between bug fixes, it is extremely beneficial to reduce this time.

When the full AMC40 firmware is compiled in Quartus the increase in occupancy of the FPGA can be measured, as discussed in section 5.1it is important to keep the occupancy lower than about 60% due to hardware constraints on the FPGA and therefore the benefit in reducing the load in the software stage needs to be weighed up with the increase in occupancy and a decision made about whether to reduce the cutoff for the number of SPPs per BCID to be sorted, as this is by far the most computationally expensive part of the ICF algorithm, as is discussed in section 6.

A risk assessment for this project was not neccessary as all of the work was done on a computer and the whole project was programming based. The only health risks involved were from sitting for long periods of time and from eye or neck strain. These were avoided by standing and stretching while looking away from the screen every half an hour and by ensuring that the screen was at the correct height.

9.2 Conclusion

Tests on the ICF block will continue at the top level to confirm that it interfaces properly with the rest of the firmware. When testing has finished, the ICF block will become part of the firmware and will run on the upgraded VELO until it is upgraded again. The development of the firmware will be continued while the electronics and pixel sensors are finalised until in December 2018, when the LHC closes down for LS2 and the upgrade is installed. The LHCb will then take data with the new setup for 10 years until the upgrades to LHCb for HL-LHC are installed during LS4 at some point in the 2030s.

References

[1] L. M. Lederman and C. T. Hill, *Beyond the God particle*, Prometheus Books, New York, 2013.

[2] L. Evans and P. Bryant, *Lhc machine*, Journal of Instrumentation **3** (2008), no. 08 S08001.

[3] P. L. Rocca and F. Riggi, *The upgrade programme of the major experiments at the large hadron collider*, Journal of Physics: Conference Series **515** (2014), no. 1 012012.

[4] F. Marcastel, *CERN's Accelerator Complex. La chane des acclrateurs du CERN*, , General Photo.

[5] ATLAS, G. Aad *et al.*, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003.

[6] CMS Collaboration, S. Chatrchyan *et al.*, *The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment*, J. Instrum. **3** (2008) S08004. 361 p, Also published by CERN Geneva in 2010.

[7] ATLAS, G. Aad *et al.*, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Phys. Lett. **B716** (2012) 1, `arXiv:1207.7214`.

[8] CMS, S. Chatrchyan *et al.*, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Phys. Lett. **B716** (2012) 30, `arXiv:1207.7235`.

[9] ALICE Collaboration, B. Alessandro, F. Antinori, J. A. Belikov, and C. Blume, *ALICE: Physics Performance Report, volume II*, J. Phys. G **32** (2006) 1295.

[10] A. D. Sakharov, *Violation of cp invariance, c asymmetry, and baryon asymmetry of the universe*, Soviet Physics Uspekhi **34** (1991), no. 5 392.

[11] X.-G. He, S.-F. Li, and H.-H. Lin, *CP Violation in $B_s^0 \to K^-\pi^+$, $B^0 \to K^+\pi^-$ Decays and Tests for SU(3) Flavor Symmetry Predictions*, JHEP **08** (2013) 065, `arXiv:1306.2658`.

[12] M. Gersabeck, *Brief Review of Charm Physics*, Mod. Phys. Lett. **A27** (2012) 1230026, `arXiv:1207.2195`.

[13] D. H. Perkins, *Introduction to High Energy Physics*, Cambridge University Press, 4 ed., 2000. doi: 10.1017/CBO9780511809040.002.

[14] LHCb, R. Aaij *et al.*, *Measurement of b-hadron branching fractions for two-body decays into charmless charged hadrons*, JHEP **10** (2012) 037, `arXiv:1206.2794`.

[15] LHCb, R. Aaij *et al.*, *Measurement of CP violation in the phase space of $B^{\pm} \to K^{\pm}\pi^{+}\pi^{-}$ and $B^{\pm} \to K^{\pm}K^{+}K^{-}$ decays*, Phys. Rev. Lett. **111** (2013) 101801, `arXiv:1306.1246`.

[16] LHCb, R. Aaij *et al.*, *Measurement of CP violation in the phase space of $B^{\pm} \to K^{+}K^{-}\pi^{\pm}$ and $B^{\pm} \to \pi^{+}\pi^{-}\pi^{\pm}$ decays*, Phys. Rev. Lett. **112** (2014), no. 1 011801, `arXiv:1310.4740`.

[17] LHCb, R. Aaij *et al.*, *First Evidence for the Decay $B_s^0 \to \mu^{+}\mu^{-}$*, Phys. Rev. Lett. **110** (2013), no. 2 021801, `arXiv:1211.2674`.

[18] R. Lindner, *LHCb layout_2. LHCb schema_2*, , LHCb Collection.

[19] LHCb collaboration, A. A. Alves Jr. *et al.*, *The LHCb detector at the LHC*, JINST **3** (2008) S08005.

[20] LHCb collaboration, R. Aaij *et al.*, *LHCb detector performance*, Int. J. Mod. Phys. **A30** (2015) 1530022, `arXiv:1412.6352`.

[21] R. Aaij *et al.*, *Performance of the LHCb Vertex Locator*, JINST **9** (2014) P09007, `arXiv:1405.7808`.

[22] M. Adinolfi *et al.*, *Performance of the LHCb RICH detector at the LHC*, Eur. Phys. J. **C73** (2013) 2431, `arXiv:1211.6759`.

[23] A. A. Watson, *The discovery of Cherenkov radiation and its use in the detection of extensive air showers*, Nuclear Physics B Proceedings Supplements **212** (2011) 13, `arXiv:1101.4535`.

[24] R. Arink *et al.*, *Performance of the LHCb Outer Tracker*, JINST **9** (2014) P01002, `arXiv:1311.3893`.

[25] A. A. Alves Jr. *et al.*, *Performance of the LHCb muon system*, JINST **8** (2013) P02022, `arXiv:1211.1346`.

[26] R. Aaij *et al.*, *The LHCb trigger and its performance in 2011*, JINST **8** (2013) P04022, `arXiv:1211.3055`.

[27] G. Apollinari *et al.*, *High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report*, CERN Yellow Reports: Monographs, CERN, Geneva, 2015.

[28] LHCb collaboration, *Expression of Interest for a Phase-II LHCb Upgrade: Opportunities in flavour physics, and beyond, in the HL-LHC era*, CERN-LHCC-2017-003.

[29] LHCb collaboration, *LHCb VELO Upgrade Technical Design Report*, CERN-LHCC-2013-021. LHCb-TDR-013.

[30] LHCb collaboration, *LHCb VELO (VErtex LOcator): Technical Design Report*, CERN-LHCC-2001-011. LHCb-TDR-005.

[31] S. D. Capua, *Lhcb velo upgrade module cad*, .

[32] G. Vouters, *Lhcb upgrade minidaq handbook*, .

[33] Altera, ©Intel Corporation, *Stratix v device datasheet sv53001-3.8*, https://www.altera.com/en_US/pdfs/literature/hb/stratix-v/stx5_53001.pdf.

[34] Mentor Graphics, *Modelsim*, https://www.mentor.com/products/fv/modelsim/.

[35] TCL Developer Xchange, *Tcl progamming language*, https://www.tcl.tk/about/language.html.

[36] Digi-Key Corporation, *Scheme-it: Design and share schematics*, https://www.digikey.com/schemeit/.