

# Isolated Cluster Flagging

## MPhys Presentation

Dónal Murray

[donal.murray@cern.ch](mailto:donal.murray@cern.ch)

26 May 2017

# Presentation Overview

## Background

- Introduction

- LHCb experiment

- VELO upgrade

## Isolated cluster flagging

- Concept

- Implementation

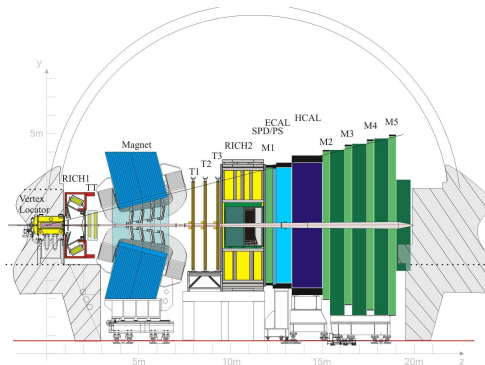
- Testing

## Summary

# Introduction

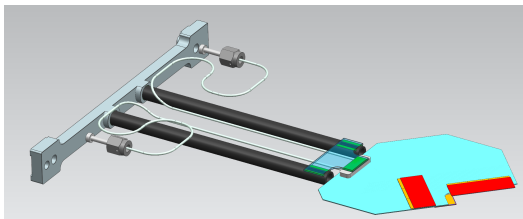
- ▶ Algorithm development for the LHCb vertex locator upgrade
- ▶ One semester MPhys project with Prof Chris Parkes and Dr Marco Gersabeck
- ▶ Majority of the project was spent learning VHDL and learning to run FPGA simulations for testing

# The LHCb experiment



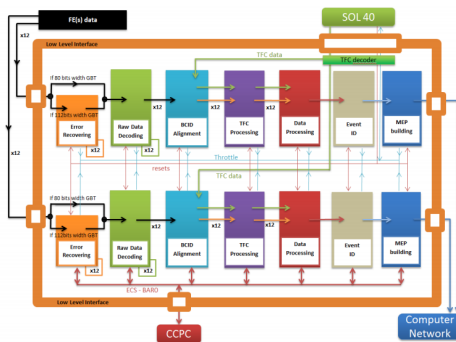
- ▶ Investigating  $b$  and  $c$  quark physics
- ▶ Consists of multiple subdetectors
- ▶ Due to be upgraded during LS2

# VELO upgrade



- ▶ 52 modules in two arrays of 26 facing each other
- ▶ Pixel sensors
- ▶ Liquid carbon dioxide cooling system
- ▶ Data readout system

# Data readout



- ▶ Readout electronics and FPGA
- ▶ Within the FPGA as much processing as possible is done
- ▶ The ICF block lies within "Data Processing"

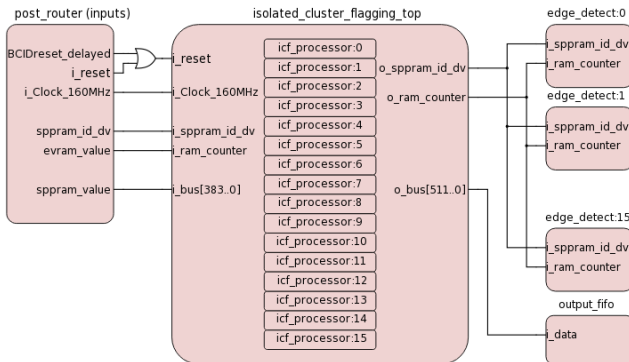
# Isolated cluster flagging

## Aims

- ▶ Clustering algorithm – do as much as possible in FPGA
- ▶ Sort the incoming superpixel packets (SPPs)
- ▶ Identify SPPs which have no neighbours
- ▶ Flag and pass on to the next stage

# Isolated cluster flagging

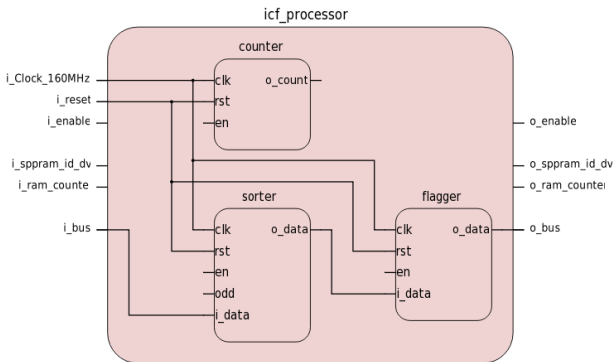
## Block diagram





# Isolated cluster flagging

## Block diagram



# Isolated cluster flagging

## Flagging

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
Signal filtered out	Not flagged	Not flagged	Signal filtered out	Flagged	Signal filtered out

- ▶ The flagging algorithm identifies SPPs with no neighbours
- ▶ All comparisons are made simultaneously

# Isolated cluster flagging

## 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 it out
- ▶ The BCID processing is parallelised within the ICF module
- ▶ At this rate, 32 million BCIDs could be processed per second

# Isolated cluster flagging

## Implementation

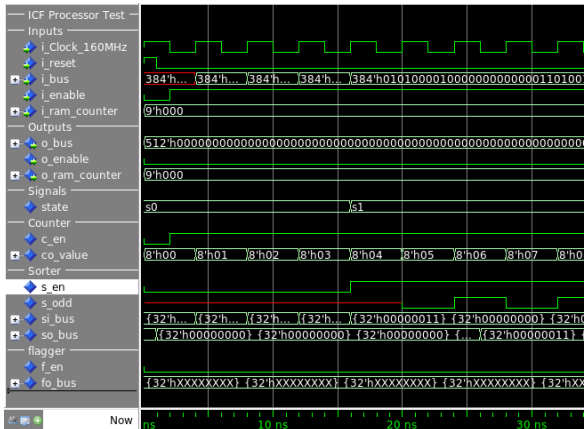
- ▶ Inherited a previous attempt at implementing the ICF block
  - ▶ Did not compile
  - ▶ Was not timing constant and dropped packets
- ▶ First rewrite from scratch: based on old design
- ▶ Gained developer access to the full firmware file
- ▶ Second rewrite from scratch: new design to fit up to date specification

# Testing

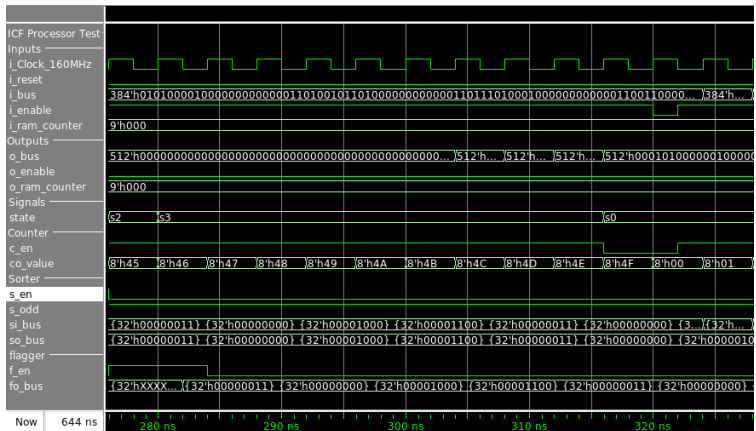
- ▶ Created C++ programs to generate test data
  - ▶ Specific data format for each section
  - ▶ Programs output both input data and output data
- ▶ Created script files to automate signal assignment

# Results

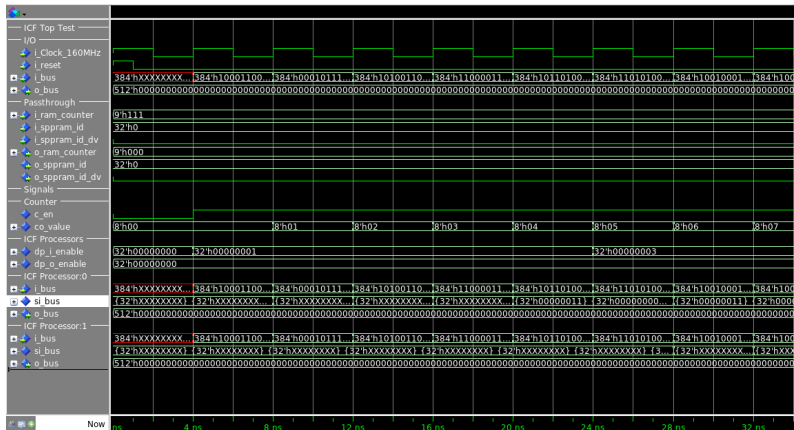
## Processor



## Processor



## Top level





# Summary

- ▶ Redesigned to interface with the new firmware modules
- ▶ Block now timing constant and order-preserving
- ▶ Tested all but top level in Modelsim

# Future work

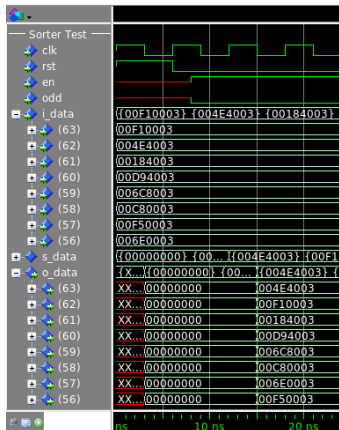
- ▶ Test top level
  - ▶ Ensure no clock cycles are wasted
  - ▶ Ensure no pileup of BCIDs
- ▶ Optimise number of processors
  - ▶ Previous research indicates 64 SPPs best cut off point
  - ▶ 16, 32 or 20 processors candidates for implementation
- ▶ Incorporate into the post router of the AMC40 firmware
  - ▶ Working version of AMC40 firmware required

# Questions

- ▶ Any questions?

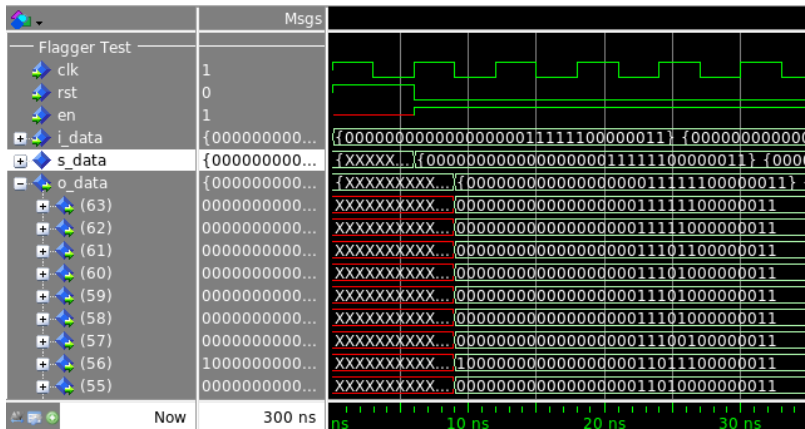
# Backup slides

## Sorter results



# Backup slides

## Flagger results



# Backup slides

## Top level test fail

