

CONTINUING THE SEARCH FOR NOTHING: INVISIBLE HIGGS BOSON DECAYS AND HIGH LUMINOSITY UPGRADES AT THE ATLAS DETECTOR

Benjamin John Rosser

A DISSERTATION

in

Physics and Astronomy

Presented to the Faculties of The University of Pennsylvania

in

Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

2021

Supervisor of Dissertation

Elliot Lipeles, Associate Professor, Physics and Astronomy

Graduate Group Chairperson

Ravi K. Sheth, Professor, Physics and Astronomy

Dissertation Committee

I. Joseph Kroll, Professor, Physics and Astronomy

Joshua R. Klein, Edmund J. and Louise W. Kahn Professor, Physics and Astronomy

Bhuvnesh Jain, Walter H. and Leonore C. Annenberg Professor in the Natural
Sciences, Physics and Astronomy

Jonathan Heckman, Associate Professor, Physics and Astronomy



CONTINUING THE SEARCH FOR NOTHING: INVISIBLE HIGGS BOSON DECAYS
AND HIGH LUMINOSITY UPGRADES AT THE ATLAS DETECTOR

COPYRIGHT
2021
Benjamin John Rosser

All rights reserved.

Acknowledgements

There are many people I could acknowledge, and I can't possibly list them all here— from the entirety of the ATLAS collaboration and its support staff at hundreds of institutions and funding agencies around the world, to many teachers and mentors over the years, to friends and acquaintances I made both at and before Penn. Please forgive any omissions!

First, I'd like to thank the Penn ATLAS faculty, Elliot Lipeles, Evelyn Thomson, Joe Kroll, and Brig Williams for being fantastic mentors and advisors. I really appreciated the open and friendly structure of the Penn group, in which students are able to work with and learn from all of the faculty members. I know I particularly benefited from being able to work with Elliot on physics while working with Joe and Evelyn on detector electronics; it was always valuable to be able to get their different perspectives on all sorts of topics, from technical issues to high-level physics questions to career advice. On a similar note, I really enjoyed being able to work closely with the Penn HEP electronics team on instrumentation projects, including Mitch Newcomer, Adrian Nikolica, Nandor Dressnandt, and especially Paul Keener and Bill Ashmanskas— I didn't have much experience with this type of work when I started at Penn, but I came to really enjoy it, in large part thanks to the strength of the instrumentation group here.

I'd also like to thank one of my undergraduate advisors, Petar Maksimovic, at Johns Hopkins University, who helped kindle my interest in high energy physics through a research project there with the CMS collaboration.

I should thank all the postdocs and older graduate students at Penn for welcoming me into the Penn ATLAS group; I came to really value them as friends and mentors over the years, and it was quite intimidating to step into the shoes of the older students when they graduated! I always looked forward to our interesting lunch conversations over the years. In particular, thanks to Elliot's former students Christian Herwig, Elodie Resseguie, and Bill Balunas, who humored me as an office-mate

for several years, and whose theses were invaluable resources when I started working on writing my own. Also thanks to Joey, Bijan, Khilesh, Leigh, Joana, Shion, and the rest of the group for their friendship and for helping to develop a very nice thesis template.

I'd also particularly like to thank Jeff Dandoy and Doug Schaefer, for being great postdoctoral mentors– Jeff started at Penn around the same time as me and we worked closely together on hardware, and Doug was a former student of Elliot's now at the University of Chicago with whom I collaborated closely on analysis. Also, of course, thanks to the entire ITk Strips ASICs and VBF+MET analysis teams, with whom I collaborated on the projects described in this thesis.

More recently, it was great having the opportunity to meet the next generation of Penn ATLAS students. I'd especially like to thank Riley Xu and James Heinlein for helping me practice and refine my thesis defense talk; also special thanks to Luis for taking over the somewhat thankless task of organizing our group meetings from me, as well as Gwen and Bobby, who I believe have inherited at least some of my research projects in the group– as well as everyone else. I've really appreciated being able to get to know you all over the last few years.

I'm grateful to the close friends I made at Penn during our first year, in particular, but not limited to, Jesse, Pedro, Stephen, and Jack– moving somewhere new is always intimidating, and it was great to meet so many people with whom I had so much in common. I hope we'll be able to stay in touch in the future. I'm also grateful to the support from Galen, Greg, Quinn, Chase, and Dan; their friendship has meant a lot to me over the years and has undoubtedly helped keep me going.

And lastly, thanks to my parents, Liz and Andy, for their all their love and support over the years.

ABSTRACT

CONTINUING THE SEARCH FOR NOTHING: INVISIBLE HIGGS BOSON DECAYS AND HIGH LUMINOSITY UPGRADES AT THE ATLAS DETECTOR

Benjamin John Rosser

Elliot Lipeles

This thesis presents two main projects involving the ATLAS experiment at the Large Hadron Collider: a search for invisible decays of the Higgs boson produced via vector boson fusion, and the design and simulation of an application specific integrated circuit produced for the Inner Tracker Strip detector upgrade project, the Hybrid Controller Chip (HCCStar). The HCCStar will be installed in the ATLAS detector around 2026 for High Luminosity LHC operations, which will see the rate of collisions increased to 200 every 25 ns. Verification of the HCCStar design was performed using cocotb, a Python framework for testing digital logic. The search for invisible Higgs decays was conducted using 139 fb^{-1} of recorded proton-proton collision data with a center-of-mass energy of $\sqrt{s} = 13 \text{ TeV}$ collected between 2015 and 2018. Observed (expected) upper limits were set on the branching ratio of the Higgs boson to an all-invisible final state at $\mathcal{B}_{\text{H} \rightarrow \text{inv.}} = 0.15$ (0.11) at a 95% confidence level. No significant disagreement from the Standard Model, which predicts $\mathcal{B}_{\text{H} \rightarrow \text{inv.}} \sim 1.0 \times 10^{-3}$, was observed. This result is then reinterpreted in the context of Higgs portal dark matter and compared to various direct detection experiments searching for evidence of weakly interacting massive particles (WIMPs).

Contents

Acknowledgements	iii
Abstract	v
Contents	vi
List of Tables	xiii
List of Illustrations	xv
Preface	xviii
1 Introduction	1
2 Theoretical Framework	3
2.1 Introduction to the Standard Model	3
2.2 Calculating Cross Sections	7
2.2.1 Cross Sections and Decay Rates	7
2.2.2 Feynman Diagrams	9
2.2.3 Renormalization	12
2.3 Electroweak Mixing and the Higgs Field	12
2.3.1 Electroweak Symmetry Breaking	13
2.3.2 W and Z Bosons	15
2.3.3 Higgs Boson	16
2.4 Dark Matter	18
2.4.1 Astrophysical Evidence	19

2.4.2	Dark Matter Candidates	20
2.4.3	Higgs Portal Dark Matter	21
3	LHC and the ATLAS Detector	23
3.1	The Large Hadron Collider	23
3.2	The ATLAS Detector	27
3.2.1	Inner Detector	30
3.2.1.1	Pixel Detector	31
3.2.1.2	Semiconductor Tracker	33
3.2.1.3	Transition Radiation Tracker	33
3.2.2	Calorimeters	34
3.2.2.1	Electromagnetic Calorimeter	35
3.2.2.2	Hadronic Calorimeter	36
3.2.2.3	Forward Calorimeters	37
3.2.3	Muon Spectrometer	37
3.3	Trigger and Data Acquisition	39
3.4	Object Reconstruction	41
3.4.1	Tracking	42
3.4.2	Electrons and Photons	43
3.4.3	Muons	43
3.4.4	Jet Finding	44
3.4.5	Missing Transverse Momentum	46
3.4.6	Jet Vertex Tagging	46
4	High Luminosity Tracker Upgrade	48
4.1	ATLAS and the High Luminosity LHC	49
4.1.1	Inner Tracker	50
4.1.2	Other Detectors	51
4.1.3	Trigger and Data Acquisition	51
4.2	Inner Tracker Strips Detector	52
4.2.1	Strips Layout	52
4.2.2	Modules and Hybrids	54
4.3	ATLAS Binary Chip	57

4.3.1	ABCStar Pipeline	58
4.3.2	Clustering Algorithm	60
4.4	Hybrid Controller Chip	61
4.4.1	Control Path	63
4.4.2	Input Channel	68
4.4.3	Packet Builder	71
4.5	Hybrid Startup and Resets	76
4.5.1	Clocking	76
4.5.2	Resets	77
4.6	Strips ASIC Development Process	78
5	HCCStar Verification	81
5.1	Functional Verification	82
5.1.1	LCB Testbench	82
5.1.2	Standalone Testbench	84
5.1.3	Code Coverage	90
5.1.4	Continuous Integration	93
5.2	Hybrid and Module-Level	94
5.2.1	Hybrid-Level Simulations	94
5.2.2	Module-Level Simulations	97
5.3	Single Event Errors	98
5.3.1	Upsets and Transients	99
5.3.2	Proton Irradiation of the Prototype	101
5.3.3	Triple Modular Redundancy	104
5.4	SEE Simulations	106
5.4.1	Diagnostic SEE Simulations	107
5.4.2	Rapid SEE Simulations	110
5.4.3	Rapid Simulation Results	115
6	VBF+MET Overview	122
6.1	Introduction to VBF+MET	123
6.1.1	Vector Boson versus Gluon-Gluon Fusion	124
6.1.2	Previous Results	126

6.2	Analysis Overview and Strategy	128
6.2.1	Signal Region Definition	129
6.2.2	Use of Monte Carlo Simulation	129
6.2.3	Background Estimation	130
6.2.4	Fitting and Uncertainties	131
6.2.5	Analysis Timeline	132
6.3	Event Selection	132
6.3.1	Triggers	133
6.3.2	Object and Variable Definitions	133
6.3.2.1	Jets	134
6.3.2.2	Leptons and Photons	135
6.3.2.3	Overlap Removal	136
6.3.2.4	Missing Transverse Momentum	137
6.3.3	Signal Region Definition	138
6.3.4	Binning	139
6.4	Experimental Uncertainties	141
6.4.1	Triggers	142
6.4.2	Leptons	143
6.4.3	Jets and MET	144
6.5	Invisible Higgs Signal Modelling	144
6.5.1	Signal Monte Carlo	145
6.5.2	Blinded Signal Region	146
6.5.3	Background Processes	148
7	Monte Carlo Generation	150
7.1	Monte Carlo Methods	151
7.1.1	Matrix Element Calculations	152
7.1.2	Parton Showers	153
7.1.3	Parton Distribution Functions	154
7.1.4	Scale Variations	154
7.1.5	MC@NLO Matching and Multijet Merging	155
7.1.6	Simulation and Pileup Reweighting	157
7.2	Overview of Samples	157

7.2.1	Signal Monte Carlo	157
7.2.2	V+Jets Monte Carlo	159
7.2.3	Other Backgrounds	162
7.3	Generating QCD V+Jets at High M_{jj}	163
7.3.1	Matrix Element Filtering	164
7.3.2	Emulating the Filter	166
7.3.3	Alternate Merging Criteria	169
7.3.4	Optimizing M_{jj} Slicing	173
7.3.5	Limitations and Future Developments	175
7.4	Jet Veto Efficiency in QCD V+Jets	177
7.4.1	Z/W Comparison	179
7.4.2	Other Differences Between Z and W	182
7.4.3	Parton-Level Issues	183
7.4.4	Parton Multiplicity Tests	185
7.5	Issues with EWK V+Jets	190
8	Background Estimation	192
8.1	V+Jets Estimate	193
8.1.1	W Control Regions	194
8.1.2	Z Control Regions	195
8.1.3	V+Jets Uncertainties	198
8.2	Reweighting W+Jets to Constrain Z+Jets	201
8.2.1	Theoretical Uncertainties	205
8.2.2	Parton Shower Uncertainties	205
8.2.3	Reweighting Uncertainties	207
8.3	Misidentified Leptons in W Control Region	207
8.3.1	Fake Electron Estimate	209
8.3.2	Fake Muon Estimate	213
8.4	Multijet Estimate: Rebalance and Smear	216
8.4.1	Generating Events with Fake MET	217
8.4.2	Pileup vs Hard Scatter Jets	219
8.4.3	Closure and Normalization	221
8.5	Multijet Estimate using FJVT	223

8.5.1	Comparison with Rebalance and Smear	226
9	Results and Interpretation	228
9.1	Likelihood Fit Details	229
9.1.1	Basic Methods	229
9.1.2	Likelihood Function	231
9.1.3	Setting Limits	234
9.2	Validation Fits	235
9.2.1	Control Region Only	236
9.2.2	High Delta Phi Validation Region	237
9.3	Unblinded Signal Region	240
9.4	Unblinded Fit Results	243
9.4.1	Limit Setting	244
9.4.2	Background-Only Fit	246
9.4.3	Postfit Yields and Distributions	248
9.4.4	Uncertainties	248
9.5	Interpretations	250
9.5.1	Higgs Portal Dark Matter	251
9.5.2	Heavier Scalar Mediators	253
9.6	Comparison to CMS Results	254
10	Conclusion	256
A	Introduction to Cocotb	258
A.1	Digital Logic and Verification	259
A.2	Introduction to Cocotb	262
A.2.1	Example Design	264
A.2.2	Example Test Code	264
A.2.3	Makefile Configuration	266
A.3	Cosimulation	267
A.4	Coroutines	268
B	Spy Buffer Firmware	272
B.1	Hardware-Based Tracking in ATLAS	273

B.2	Hardware Tracking for the Trigger	275
B.3	Spy Buffer	277
B.3.1	Flow Control	278
B.3.2	Monitoring and Readout	279
B.3.3	Event Boundaries and the Event List	280
B.3.4	Playback Mode	281
B.4	Spy Buffer Status and Verification	282
C	Preliminary VBF+MET Results	284
C.1	Pre-Fit Signal Region	285
C.2	Post-Fit Results	286
C.2.1	Limits Setting	288
C.2.2	Yields and Distributions	288
C.2.3	Uncertainties	289
C.3	Comparison to Final Analysis	290
C.4	Combination with Other Limits	291
D	Glossary	293
	Bibliography	297

List of Tables

2.1	Charge and weak hypercharge for each type of fermion in the Standard Model.	16
4.1	Summary of ITK strip module types, and the number of sensors and ASICs on each. . .	56
4.2	Summary of the different layers and disks of the ITK strips detector.	57
4.3	Summary of the LCB fast commands used by the HCCStar.	64
4.4	Full list of packet types that can be produced by the HCCStar.	74
5.1	Code coverage scores for the HCCStar v0 and v1 designs.	92
5.2	Table listing the numbers of registers (flip-flops) and wires targeted by SEUs and SETs in HCCStar v1.	116
5.3	SEE injection performance for the standalone and hybrid-level setups.	116
5.4	Error rates seen during the non-triplicated standalone simulations.	119
5.5	Error rates seen during the non-triplicated hybrid-level simulations.	119
5.6	Error rates seen during the non-triplicated module-level simulations.	120
6.1	Cutflow table showing the efficiency of each cut on the VBF and ggF signal processes. .	148
7.1	Example cross sections for different m_{jj} slices of V+Jets processes, as generated at NLO using Sherpa 2.2.	164
7.2	High m_{jj} event yields from each parton m_{jj} slice compared between default, k_t , and anti- k_t merged samples.	171
7.3	Ratio of jet veto efficiency between $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ in each m_{jj} bin.	180
7.4	Ratio of jet veto efficiency between $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ in each m_{jj} bin, calculated separately for W^+ and W^- events.	182
7.5	Z/W jet veto efficiency ratios calculated from the two-parton and four-parton test samples.	188
8.1	Fake lepton estimate transfer factors, R_S , computed with $E_T^{\text{miss}} > 160$ GeV.	212
8.2	Fake lepton estimate transfer factors, R_S , computed with $E_T^{\text{miss}} > 200$ GeV.	212
8.3	Normalization factors for the rebalance and smear multijet estimate.	222
8.4	Summary of the multijet prediction from the rebalance and smear method.	223
8.5	Summary of multijet prediction from the FJVT control region.	226
9.1	Best fit values of the V+Jets transfer factors in the high- $\Delta\phi_{jj}$ validation region.	239
9.2	Post-fit yields recorded in the high- $\Delta\phi_{jj}$ validation region.	240

9.3	Observed and expected 95% confidence level limits on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$ from the high- $\Delta\phi_{\text{jj}}$ validation region.	240
9.4	Pre-fit yields recorded in the signal region and all control regions.	242
9.5	Best-fit values of the V+Jets, fake lepton, and multijet transfer factors in each bin. . . .	245
9.6	Observed and expected limits on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$, set at a 95% confidence level.	246
9.7	Post-fit yields recorded in the signal region and all control regions.	248
9.8	Size and impact of each statistical and systematic uncertainty on the limit.	251
9.9	Uncertainties on the observed and expected limit from the full run 2 CMS analysis. . . .	255
C.1	Pre-fit yields in the signal and control regions in the preliminary analysis.	286
C.2	Best-fit values of the V+Jets and fake lepton transfer factors.	288
C.3	Observed and expected limits on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$, set at a 95% confidence level.	288
C.4	Post-fit yields in the signal and control regions in the preliminary analysis.	289
C.5	Impact of each statistical and systematic uncertainty on the preliminary limit.	290
C.6	Change in the expected and observed limits between the preliminary and final analyses. .	291

List of Illustrations

2.1	Visual representation of the fundamental particles that make up the Standard Model. . .	4
2.2	Example Feynman diagrams for a $2 \rightarrow 2$ scattering process in QED.	11
3.1	Diagram of the CERN accelerator complex.	24
3.2	Diagram of the proton bunch structure used to fill the Large Hadron Collider.	26
3.3	Plots showing the integrated luminosity of proton-proton collisions collected by the ATLAS detector.	27
3.4	General cut-away view of the ATLAS detector.	28
3.5	Illustration of different types of particles passing through the ATLAS detector.	29
3.6	General cut-away view of the inner detector.	31
3.7	Layout plot of one ϕ slice of the Inner Detector.	32
3.8	General cut-away view of the LAr and Tile calorimeters.	35
3.9	Diagram of a barrel module in the liquid argon calorimeter.	36
3.10	General cut-away view of the ATLAS muon system.	38
3.11	Layout plot of one ϕ slice of the muon spectrometer.	39
3.12	Diagram of the ATLAS Run 2 TDAQ system.	40
3.13	Sample event clustered using both the k_t and anti- k_t algorithms.	45
4.1	Plot showing a 2D slice of the ITk strip detector layout in R and z	53
4.2	Renderings of an example stave and petal for the ITk strip detector.	54
4.3	Cutaway diagram of an example ITk strip barrel module.	55
4.4	Top-level block diagram of the ABCStar ASIC.	58
4.5	Top-level block diagram of the HCCStar ASIC.	62
5.1	Example performance plots from the hybrid simulation testbench for HCCStar v0. . . .	97
5.2	SEU rate measurement performed for HCCStar v0 at the second TRIUMF irradiation. .	103
5.3	Illustration showing examples of the triple modular redundancy technique.	106
5.4	Flowchart demonstrating the “recovery and reset” process in the rapid SEE simulations.	112
5.5	Average numbers of SEUs per flip-flop and SETs per wire in the TMR and non-TMR standalone SEE simulations over a two-week period.	118
6.1	Feynman diagrams for invisible Higgs decay in the four main channels.	125
6.2	Previous limits on $\mathcal{B}_{\text{H}} \rightarrow \text{inv.}$ set by the ATLAS collaboration.	126
6.3	Previous limits on $\mathcal{B}_{\text{H}} \rightarrow \text{inv.}$ set by the CMS collaboration.	127

6.4	3D rendering showing a candidate VBF Higgs to invisible event.	130
6.5	Signal region binning scheme used in the preliminary full run 2 analysis.	140
6.6	Signal region binning scheme used in the final full run 2 analysis.	141
6.7	Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and $E_{\text{T}}^{\text{miss}}$ in the VBF+MET signal region.	147
7.1	Example QCD and electroweak Feynman diagrams for the LO $Z(\nu\nu)+2\text{jet}$ process.	160
7.2	Illustration showing different LO and NLO matrix elements for V+jets processes.	161
7.3	Truth-level m_{jj} calculated from $Z \rightarrow \nu\nu$ MC generated with m_{jj} slicing, using Sherpa 2.2 and the default merging criterion.	166
7.4	Truth-level and matrix-element level m_{jj} plotted against each other from $Z \rightarrow \nu\nu$ test samples.	168
7.5	Truth-level m_{jj} calculated from $Z \rightarrow \nu\nu$ MC generated with m_{jj} slicing, using Sherpa 2.2 and the k_t merging criterion.	170
7.6	Comparison of matrix element parton multiplicity between default, k_t , and anti- k_t samples.	171
7.7	Comparison of truth m_{jj} between the default and k_t -merged $Z \rightarrow \nu\nu$ samples.	172
7.8	Comparison of truth p_{T}^V between the default and k_t -merged $Z \rightarrow \nu\nu$ samples.	173
7.9	Estimated fractional uncertainty as a function of the relative size of each m_{jj} slice.	174
7.10	Comparison between inclusive and m_{jj} -sliced $Z \rightarrow \nu\nu$ test samples.	177
7.11	Third jet veto efficiency comparison between $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ as a function of m_{jj}	179
7.12	Z/W third jet veto efficiency comparison performed separately for $W \rightarrow e\nu$ and $W \rightarrow \mu\nu$	181
7.13	Comparison of N_{jets} between $Z \rightarrow \nu\nu$ and $W \rightarrow e\nu$ at low- and high- m_{jj}	183
7.14	Comparison of η^V between $Z \rightarrow \nu\nu$ and $W \rightarrow e\nu$ before and after applying a jet veto.	183
7.15	$W \rightarrow \mu\mu$ and $Z \rightarrow \nu\nu$ samples divided by the flavour of their incoming partons.	184
7.16	Z/W third jet veto efficiency comparison performed separately for qq and qg events.	185
7.17	Comparison of invariant mass of the vector sum of the incoming partons between $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ samples.	186
7.18	Comparison of matrix element parton multiplicity between $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ samples.	187
7.19	Jet veto efficiencies calculated for $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ test samples in four-parton LO and NLO configurations.	188
7.20	Jet veto efficiencies calculated for $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ test samples in two- and three-parton LO and NLO configurations.	189
7.21	Ratio of $Z \rightarrow \nu\nu$ and $Z \rightarrow \mu\mu$, taken from electroweak V+Jets MC.	191
8.1	Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and the transverse momentum of the vector sum of $E_{\text{T}}^{\text{miss}}$ and the visible lepton in the $W \rightarrow e\nu$ control region.	196
8.2	Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and the transverse momentum of the vector sum of $E_{\text{T}}^{\text{miss}}$ and the visible lepton in the $W \rightarrow \mu\nu$ control region.	197
8.3	Pre-fit plot showing the missing transverse momentum, $E_{\text{T}}^{\text{miss}}$, in the $Z \rightarrow ll$ control region.	198
8.4	Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and the transverse momentum of the vector sum of $E_{\text{T}}^{\text{miss}}$ and the visible lepton in the $Z \rightarrow ll$ control region.	199
8.5	Relative resummation and merging scale uncertainties calculated for the Sherpa QCD V+Jets samples with a loose truth-level selection applied.	200
8.6	Transfer factor uncertainties calculated separately for W+Jets and Z+Jets samples on the renormalisation and factorisation scale variations, evaluated separately in bin.	202
8.7	Comparison between $W \rightarrow l\nu$ and $Z \rightarrow ll$ yields in the one- and two- lepton control regions as a function of m_{jj}	204
8.8	Plots of the function used to reweight W+jets MC simulation to Z+jets, along with corresponding theoretical uncertainties.	206

8.9	Plots of the missing transverse momentum significance, S_{MET} , in the $W \rightarrow e\nu$ anti-ID region, for each data-taking period.	210
8.10	Plots of the fake electron transfer factor, R_S	211
8.11	Inclusive missing transverse momentum significance, S_{MET} , in the $W \rightarrow e\nu$ anti-ID region.	212
8.12	Comparison of cut efficiency between the object- and non-object-based definitions of S_{MET} in the $W \rightarrow e\nu$ control region.	213
8.13	Plots of S_{MET} in a hypothetical $W \rightarrow \mu\nu$ anti-ID region.	215
8.14	Plots of the inclusive transverse mass, m_T , in the $W \rightarrow \mu\nu$ anti-ID region.	216
8.15	Plots of the fake muon estimate transfer factor, R_M	216
8.16	Sketch outlining the rebalance and smear procedure used for the multijet estimate.	218
8.17	Post-fit plots of $\Delta\phi_{jj}$ and m_{jj} in the loose multijet control region.	220
8.18	Plots of $\Delta\phi_{jj}$ and m_{jj} in the first multijet normalization region.	221
8.19	Plots of $\Delta\phi_{jj}$ and m_{jj} in the second multijet normalization region.	222
8.20	Multijet control region, defined by inverting the FJVT requirement on the leading jet.	224
8.21	Low-MET multijet validation region plot showing the FJVT estimate.	225
8.22	Comparison between multijet predictions from the R&S and FJVT control region methods.	227
9.1	Results from the “control region only” fit, showing bin-by-bin data/MC agreement.	236
9.2	Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and E_T^{miss} in the blinded high- $\Delta\phi_{jj}$ validation region.	238
9.3	Ranking plot showing the 20 largest nuisance parameters in the high- $\Delta\phi_{jj}$ VR.	241
9.4	Fit results from the high- $\Delta\phi_{jj}$ validation region, showing bin-by-bin data/MC agreement.	242
9.5	Plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and E_T^{miss} in the unblinded VBF+MET signal region.	243
9.6	Pre-fit agreement between data and MC in all regions and bins.	244
9.7	Post-fit agreement between data and MC in all regions and bins.	245
9.8	Post-fit agreement between data and MC in all regions and bins for a background-only fit.	247
9.9	Post-fit m_{jj} and $\Delta\phi_{jj}$ distributions in the signal region.	247
9.10	Ranking plot showing the 30 largest nuisance parameters in the fit.	249
9.11	Higgs portal dark matter interpretation compared to limits on the spin-independent WIMP-nucleon cross section from dark matter direct detection experiments.	253
9.12	Limits on the invisible decay of a new non-Higgs scalar mediator as a function of mass.	254
9.13	Observed and expected limits on $\mathcal{B}_{H \rightarrow \text{inv.}}$ in the vector boson fusion channel with the full run 2 CMS dataset.	255
B.1	Schematic diagram showing the architecture of an HTT unit.	276
C.1	Pre-fit agreement between data and MC in all bins and regions in the preliminary analysis.	286
C.2	Post-fit agreement between data and MC in all bins and regions of the preliminary analysis.	287
C.3	Post-fit plots of m_{jj} and $\Delta\phi_{jj}$ in the preliminary signal region.	288
C.4	Observed and expected limits on $\mathcal{B}_{H \rightarrow \text{inv.}}$ combined with a full run 2 ttH analysis and run 1 data.	292

Preface

In this thesis, I have tried to document a complete record of everything I did as a member of the ATLAS collaboration [1] during my time as a graduate student at Penn. In addition to a reasonably standard introduction to the theory of the Standard Model of particle physics and the ATLAS detector itself, most of the remaining text is dedicated to the two main projects I was involved with, a detector electronics upgrade and a search for new physics. This preface just gives a brief overview of my personal involvement in these projects:

New members of the ATLAS collaboration must perform a “qualification task”, in which they perform some service to the collaboration as a whole before they become an author and put their name on physics papers. I came to Penn in 2016, and after finishing my first-year physics classes in 2017, began working on my qualification task: the [Hybrid Controller Chip \(HCCStar\) application specific integrated circuit \(ASIC\)](#). The HCC was being developed by the Penn high energy physics engineering group, including (but not limited to) Paul Keener, Nandor Dressnandt, Bill Ashman-skas, Mitch Newcomer, and Adrian Nikolica, for use in the ATLAS Inner Tracker upgrade project. ASICs are usually developed using a special type of programming language, a [hardware description language \(HDL\)](#), that compiles down into logic gates. It then takes several months and hundreds of thousands of dollars to turn this representation of a circuit into a physical chip, which then cannot be reprogrammed. Therefore, it is critical that the digital logic be carefully tested prior to fabrication to ensure that it is correct. I joined the design team for the chip to work on this verification, along with a new Penn postdoc, Jeff Dandoy. Together, we developed a simulation framework to test the correctness of the chip’s logic prior to submission. Both the simulation framework and the chip architecture itself are described here in Chapters 4 and 5.

I had studied both computer science and physics as an undergraduate, and I had some prior experience in high energy physics research with the CMS collaboration at Johns Hopkins University.

I had no previous experience in hardware or digital design before I started working on the HCC, however, so this turned out to be a very interesting learning opportunity for me. I came to find this work, which involved writing software to run tests on a digital simulation of the circuit, to be a really interesting blend of my computer science background with the actual design of a particle physics detector. After completing my qualification task in 2018 (when we submitted an initial version of the chip for fabrication), I then began work on my second major project— a search for new physics. But I stayed involved with the HCCStar as well, helping to perform a beam test of the chip in 2019 and working on more simulations in 2020 and 2021 on a second and (hopefully) final revision of the design. I really enjoyed the chance to learn about chip design and work closely with the engineering group here at Penn on this project.

In 2018, I started working on a physics analysis, a search for “invisible” decays of the Higgs boson. I’ve always been interested in the nature of dark matter, and if there is some new particle or particles that comprise dark matter, one possibility is that those new particles would couple to the Higgs boson. If so, the Higgs might decay “invisibly” to these particles, and it might be possible to see evidence of this at the LHC. Run 2 of the LHC was underway at the time, and a version of this search with data taken in 2015 and 2016 that a previous Penn student, Bill Balunas, had worked on was just wrapping up. Elliot was interested in staying involved, and so I joined the analysis team and began working on the full run 2 analysis, with four times as much data recorded between 2015 and 2018. I became specifically involved in issues related to the generation and use of Monte Carlo datasets in order to estimate Standard Model background processes, and spent a lot of time over the last few years working on these topics, among other areas. We released a “conference note” with a preliminary result in March 2020, and are currently preparing a paper, which will contain the final result. That paper should, ideally, be public within the next few months, depending on how long the rest of the internal ATLAS approval process takes.

Most analyses in ATLAS are done by teams of people from different institutions, and this one is no exception. I worked quite closely with Doug Schaefer at the University of Chicago, Ben Carlson at the University of Pittsburgh, and Othmane Rifki and Christian Sander at DESY, but many other people from these and other institutions were involved as well. I was heavily involved in the Monte Carlo generation presented in Chapter 7, the V +jets and fake lepton background estimation presented in parts of Chapter 8, and the validation region fit introduced in Chapter 9, but other areas of the analysis like the multijet background estimate, the statistical model, and the dark matter interpretation were primarily developed by others. I have tried to give a holistic overview of all the work that was done on all areas of the analysis, including at least as much detail as is

included in the paper draft and conference note (primarily written by Elliot Lipeles and Christian Sander) and significantly more for topics that I was heavily involved in personally. All members of the analysis team contributed to what's known as an "internal note", a more detailed internal record of all the work that was done and all the studies performed. The text of this thesis was of course solely written by me, but I did use the note as a guide when writing, and included additional plots and tables from that document to give additional detail where necessary.

I have tried to write in a somewhat pedagogical style, in the hopes that this thesis might be useful to future generations of graduate students working on either the next phase of the Inner Tracker upgrade or on the next iteration of the Higgs to Invisible analysis. I've found that there is a lot of highly specialized language and terminology, including many acronyms, which are not necessarily well documented in a way that's easily accessible to new students. Therefore, I've tried to explain as much as I could here, both generally and specifically, about how an analysis like the invisible Higgs search is conducted and how digital logic verification of an ASIC like the HCCStar can be done. My hope is that a graduate student with little prior experience in the field might be able to learn something from this, but even if that never happens, it was an incredibly valuable exercise for me to try and firm up my own knowledge of all these topics. I look forward to continuing to perform similar work at the next stage of my career.

Benjamin John Rosser

Philadelphia, November 2021

CHAPTER 1

Introduction

The true nature of dark matter is one of the major unanswered questions in high energy physics. While the Standard Model of particle physics has been quite successful, it is known to be an incomplete theory. It describes all known fundamental subatomic particles and three of the four fundamental forces, but among other issues, does not include a quantum explanation for the fourth force, gravity; it does not include an explanation of the matter/antimatter asymmetry that explains why the universe is dominated by matter, and not antimatter; and it does not include an explanation for dark matter. Astrophysical evidence from the rotation curves of galaxies [2] and gravitational lensing [3] appears to suggest that either something is fundamentally wrong with Newtonian theories of gravity, or that a large fraction of the universe’s mass is actually some form of non-interacting “dark matter” comprised of one or more new, undiscovered particles.

High energy physics experiments at particle colliders are one way to try and answer these questions. By accelerating particles like protons to a very high energy and then colliding them together, experimentalists can try and create new, undiscovered particles that have been proposed by theorists, or set limits on their existence if they fail to find them. Additionally, very precise measurements can be made of various parameters, like masses and decay rates, and comparisons can be performed between observation and theory prediction in order to look for any evidence of disagreement with the Standard Model. In 2012, the ATLAS [1] and CMS [4] experiments at the Large Hadron Collider [5], the world’s most powerful particle accelerator as of this writing, reported the observation of a Higgs boson [6] [7], one of the last missing pieces of the Standard Model. The Higgs couples to the quarks, charged leptons, and massive bosons that make up the rest of the Standard Model, and the strength of that coupling gives those particles their mass¹. If dark matter really consists of a new,

¹Neutrinos, while technically massless in the Standard Model, have been discovered to have mass. Depending

undiscovered massive particle, one possibility is that it also couples to the Higgs in a similar way [8]. If this theory, known as “Higgs portal” dark matter, is correct, it may be possible to produce and observe dark matter particles at the LHC by means of the Higgs boson. This thesis presents work intended to answer this question.

If this interaction can really occur, it— along with any other new physics beyond the Standard Model— might be quite rare. That means it might be necessary to perform a very large number of collisions at the LHC in order to see this or any other new physics occur. Currently, the ATLAS detector observes around tens of protons collide every 25 ns², or 40 million times every second. Work is underway to increase the collision rate and collide up to 200 protons every 25 ns, in order to collect more data in the same amount of time and therefore potentially increase the chances of observing new physics. The current ATLAS detector was not designed for such a high collision rate, and so the “high luminosity” upgrade program to the LHC will require major upgrades to the detector electronics in order to support it. This thesis additionally presents an overview of some of this upgrade work.

In addition to the introduction, which also includes Chapters 2 and 3, the thesis can be thought of as being divided into two parts. The first part covers the high luminosity upgrade work, in Chapters 4 and 5, along with associated Appendixes A and B. The second part covers the search for the invisible Higgs boson decay, in Chapters 6, 7, 8, and 9, along with Appendix C. The actual results of the analysis, in which we set an upper limit on the possible branching ratio, or decay rate, of the Higgs boson to invisible particles can be found in Chapter 9. Both parts are designed to be somewhat standalone; readers only interested in the upgrade work should be able to just read those chapters, while readers interested in the physics analysis should be able to skip them and jump right to Chapter 6. Each chapter includes its own short introduction and outline of the content included, with cross-references between chapters and sections where appropriate.

The remainder of the introduction consists of a more detailed overview of the theory of the Standard Model, of the nature of the Higgs boson and Higgs mechanism, and of evidence for dark matter in Chapter 2. Then Chapter 3 outlines how the LHC and ATLAS detectors work, and describes how particles and other objects can be detected from the data recorded by ATLAS.

on the nature of that mass, and whether the neutrino is Majorana or Dirac (meaning whether the antineutrino is identical to the neutrino), neutrinos might couple to the Higgs as well, though the coupling will be extremely small.

²More precisely, as many as 50-60 per 25 ns at the moment, but the detector was only designed for a collision rate of around 20 protons every 25 ns. See Section 3.1 for more details on this.

CHAPTER 2

Theoretical Framework

This chapter outlines some of the theory of the Standard Model of particle physics, the underpinning of the experimental work discussed in the rest of this thesis. Section 2.1 gives an introduction to the Standard Model, and lists the known particles and fields and some of their properties. Section 2.2 explores how this theory can be used to calculate observable decay rates and cross sections at colliders, touching on the meaning of Feynman diagrams and the ideas behind renormalization. Section 2.3 then talks about electroweak symmetry breaking, the way in which two of the three fundamental forces—electromagnetism and the weak force—can be combined, with help from the Higgs field. The properties of the weak force and nature of the Higgs boson are also discussed. Finally, Section 2.4 covers one of the problems that the Standard Model does not explain: the nature of dark matter.

Note that the quantum field theory introduced in Sections 2.2 and 2.3, in particular, was based on the treatment given to the introductory particle physics and quantum field theory books written by Schwartz [9], Peskin and Schroeder [10], and Thomson [11]. The math in these sections was adapted from certain chapters of these books, which are referenced inline. Readers interested in more detail about these topics should consult those sources.

2.1 Introduction to the Standard Model

The Standard Model describes all of the known elementary particles that make up matter, as well as three of the four fundamental forces— the strong and weak nuclear forces, as well as the electromagnetic force— that govern the interactions between those particles [12]. A visual diagram of these particles is shown in Figure 2.1, formatted like a “periodic table” of particle physics, as the

Standard Model is sometimes known. The mass, spin, and charge of each particle is listed in this image.

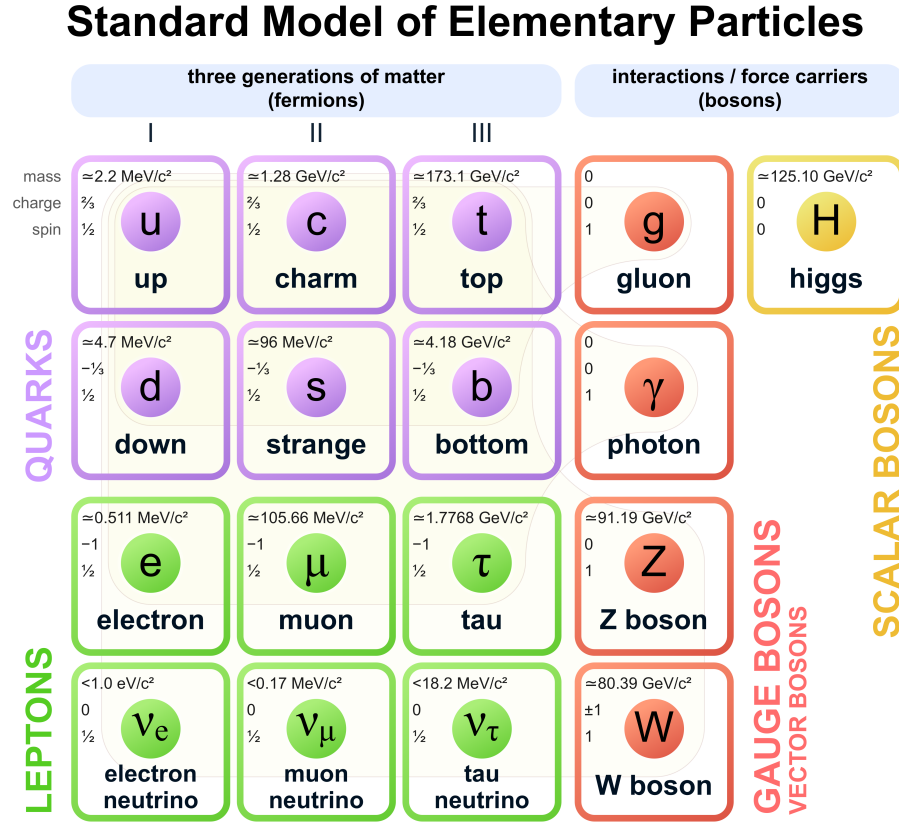


Figure 2.1: Visual representation of the fundamental particles that make up the Standard Model, taken from Wikimedia [13]. The mass, charge, and spin of each particle—the three generations of quarks and leptons, the four force carrier bosons, and the Higgs boson are all listed.

Particles in the Standard Model can be divided into fermions and bosons according to their spin. Fermions are those particles which have half-integer spin ($\pm 1/2$, etc.), and include both the leptons (containing charged leptons and neutral neutrinos, and highlighted in green in Figure 2.1) and the quarks (highlighted in purple). Bound states of two or three³ quarks form composite particles known as hadrons: a two-quark hadron is commonly referred to as a “meson”, and a three-quark hadron a “baryon”. The protons and neutrons, which along with electrons make up the atoms that comprise ordinary matter, are baryons, with two up quarks and a down quark or two down quarks and an up

³Or more; experiments like LHCb [14] have discovered tetraquark bound states with four quarks [15], and there is no theoretical reason why bound states of five (pentaquarks) or more quarks should not also exist.

quark, respectively. Ordinary matter is therefore sometimes known as “baryonic” matter. But there are other baryons and mesons that can be formed from other combinations of quarks. The lepton and quark families are also divided into three “generations”: the particles which make up ordinary matter (the electron and electron neutrino, and the up and down quark) are the “first” generation, with subsequent “second” (the muon and muon neutrino, and the strange and charm quarks) and “third” generations (the tau and tau neutrino, and the bottom and top quarks) discovered later. In principle, there is no reason there should only be three generations of fermions, but various experimental evidence has tightly constrained the existence of a fourth generation.

Bosons are particles with integer spin ($0, \pm 1, \pm 2$, etc.). Note that mesons will have integer spin, because they contain exactly two quarks with half-integer spin, and so they are a composite boson. There are also elementary bosons in the Standard Model that each represent the quanta of a given field. There are two types of these bosons: the force-carrier gauge bosons (shown in orange in Figure 2.1), which are each associated with one of the fundamental forces; and the Higgs boson, which is the quanta of the Higgs field that gives these fundamental particles their masses. More on the nature of the Higgs boson and the Higgs mechanism can be found below in Section 2.3. Note that the bosons with spin zero—just the Higgs in the Standard Model—are called “scalars” or “scalar bosons”, because the field that they are associated with is a scalar field. The bosons with spin greater than zero are called “vectors” or “vector bosons”, because the field they are associated with is a vector field. When particles interact via one of the fundamental forces, they do so by exchanging the corresponding gauge boson of that force.

Each force in the Standard Model is associated with a gauge symmetry and, by Noether’s theorem [16], a conserved quantum number. Mathematically speaking, the bosons are the generators of the gauge group of that symmetry. The electromagnetic force is a $U(1)$ symmetry, with one generator, the photon, and the electric charge is the quantum number. The weak force is associated with the “flavour” of particles, which distinguishes the different generations in the same families from each other. A conserved quantity known as the “weak isospin” is defined, and there is a $SU(2)$ symmetry with three generators, the charged W^+ and W^- bosons, as well as the neutral Z boson. More on the weak interaction and weak isospin can be found in Section 2.3 below, where the unified $SU(2) \times U(1)$ electroweak symmetry is discussed. Finally, the strong force has a $SU(3)$ symmetry, with eight generators and a conserved quantum number called “color”. Quarks carry color in addition to electric charge, and there are three possible color states, which are conventionally labelled red, green, and blue, and composite hadrons must be color neutral. Gluons, which are the bosons associated with the strong force, each carry one of eight linear combinations of the three colors (and corresponding

anti-colors), and form the basis of eight generators of $SU(3)$: $r\bar{g}, g\bar{r}, r\bar{b}, b\bar{r}, g\bar{b}, b\bar{g}, \frac{1}{\sqrt{2}}(r\bar{r}-g\bar{g}), \frac{1}{\sqrt{6}}(r\bar{r}+g\bar{g}-2b\bar{b})$. The entire Standard Model can then be said to have a $SU(3) \times SU(2) \times U(1)$ symmetry.

The strengths of these three forces vary. At low energies, the strength of the electromagnetic coupling is given by the fine-structure constant, $\alpha = 1/137$. while a corresponding coupling constant to the strong force is much larger, with $\alpha \approx 1$ at the QCD energy scale. The corresponding weak coupling constant is $\alpha_W \approx 1/30$, but the weak bosons, unlike the gluon and photon, have nonzero mass. This nonzero mass arises due to the Higgs mechanism, which causes the $SU(2) \times U(1)$ electroweak force to spontaneously break, as described in Section 2.3, and creates massive weak bosons while preserving a massless photon. The mass of the W and Z bosons suppresses the strength of the weak force considerably, and causes it to propagate slower than the strong and electromagnetic forces. Note that these coupling constants vary as the energies of the interactions increase: at the energies of the [Large Hadron Collider \(LHC\)](#), the strong coupling is closer to $\alpha_s \approx 1/10$ [11].

The Standard Model is a quantum field theory, and so accordingly can be written as a Lagrangian with terms for each possible interaction between particles. For example, a Lagrangian for the theory of [Quantum electrodynamics \(QED\)](#)— just governing the electromagnetic force— can be written below in Equation 2.1 [10]. Here, $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$ is the normal electromagnetic field tensor, $A_\mu = (\phi, \vec{A})$ the electromagnetic four-potential, and the notation $\not{\partial} = \gamma^\mu \partial_\mu$ shorthand for contraction with the γ matrices:

$$\mathcal{L}_{\text{QED}} = \bar{\psi}(\not{\partial} - m)\psi - \frac{1}{4}F_{\mu\nu}^2 - e\bar{\psi}\gamma^\mu\psi A_\mu \quad (2.1)$$

This Lagrangian governs the interaction of a fermionic particle ψ , which in this simplified case could just be the electron, e , with mass m and charge $Q = -1$, electromagnetic field $F_{\mu\nu}$, the quanta of which is a photon. The strength and form of that interaction is determined by the final term, $e\bar{\psi}\gamma^\mu\psi A_\mu$. This indicates that the interaction can consist of an electron e^- (or positron, e^+) radiating a photon γ through the Bremsstrahlung process, $e^- \rightarrow e^-\gamma$, or a reversed process in which a photon can pair-produce an electron and positron, $\gamma \rightarrow e^-e^+$. These sorts of interactions are often visually represented through Feynman diagrams. The exact meaning of Feynman diagrams, and how they can be used to perform calculations in quantum field theory, is explained in Section 2.2 below.

More complex Lagrangians can be written to describe the strong and weak interactions. Quantum electrodynamics is an Abelian gauge theory, while the theories of the strong ([Quantum chromodynamics \(QCD\)](#)) and weak (sometimes historically called “quantum flavourdynamics”⁴) are

⁴Along with these terms, some authors, such as Griffiths, give “quantum geometrodynamics” (QGD) for a potential

non-Abelian gauge theories, meaning that their generators do not commute. The practical implication of this is that gluons and weak bosons can self-interact with each other in 3-boson and 4-boson vertices, while photons cannot. A specific consequence of QCD, with massless, self-interacting gluons, is that it is impossible for free particles to exist that are not color neutral, something that is referred to as confinement. It is energetically favorable for a free quark or gluon to create more free quarks or gluons, which leads to hadronization, the formation of composite bound states like mesons or baryons [9].

2.2 Calculating Cross Sections

When performing experiments at a particle collider, we are generally trying to measure the rate at which a given interaction between particles occurs. This is commonly done by calculating the cross section, σ . The concept of a cross section, which has units of area, from fixed-target scattering experiments as the area over which an interaction could occur, or, put another way, the probability of an interaction occurring relative to a given flux through some area. In collider experiments, that flux is the intersection of two beams, which collide over some interacting area. Cross sections are often written in units of barns⁵, with $1 \text{ b} = 10^{-28} \text{ m}^{-2}$. The amount of data gathered is then expressed in units of inverse cross section, as a number of events recorded per interaction area. As a measure of the probability of some process occurring, cross sections can be computed using the quantum field theory that describes the Standard Model. Unfortunately, like many quantum mechanics problems, it is not possible to compute them analytically. This section describes the perturbation theory and renormalization approaches that are employed in order to do calculations. The theory of quantum electrodynamics is used as a concrete example, but the rules and techniques shown here can be generalized to the entire Standard Model.

2.2.1 Cross Sections and Decay Rates

Collider experiments involve a $2 \rightarrow n$ interaction, in which two incoming particles collide and produce some number of final state particles⁶. This can be represented quantum mechanically using the scattering matrix S , which is just the time evolution operator acting on an initial state i and

quantum field theory of gravity [17]—although there are issues involved in constructing one, and the Standard Model does not contain an explanation of gravity.

⁵This famously dates back to the Manhattan project. The name comes from the phrase “couldn’t hit the broadside of a barn”, and the size of 1 b is roughly the cross section of a uranium atom; the name was adopted as a covert way to talk about nuclear physics research.

⁶The math presented in this section is adapted from Chapter 5 of Schwartz [9], and Chapter 3 of Thomson [11]. In particular, the Lorentz-invariant phase space notation used here is from Schwartz.

evolving it into a final state f . The differential cross section can be related to this probability as shown below in Equation 2.2 [9]. Here, $d\Pi$ is a momentum phase space that would need to be integrated over in order to get a full cross section, and $V/(T\Delta\vec{v})$ represents the flux, as the volume of the experiment V divided by the total experiment time T and difference in velocity $\Delta\vec{v}$ between the two incoming particles.

$$d\sigma = \frac{V}{T} \frac{1}{|\vec{v}_1 - \vec{v}_2|} \frac{|\langle f|S|i\rangle|^2}{\langle f|f\rangle \langle i|i\rangle} d\Pi \quad (2.2)$$

This expression can be simplified using perturbation theory to rewrite the scattering matrix $S = 1 + iT$. Here, the transfer matrix T is a perturbation relative to the identity matrix, where no interaction occurs and the initial state is unchanged. Imposing momentum conservation between the initial and final states, the transfer matrix can be written as follows [9]:

$$\langle f|T|i\rangle = (2\pi)^4 \delta^4(\sum p) \mathcal{M} \quad (2.3)$$

$\mathcal{M}$ is generally known as the “matrix element”, and calculating it is quite involved. But the other terms, involving the definition of the phase space, volume, and momentum conservation, can be simplified considerably. Equation 2.4 gives the cross section for a $2 \rightarrow n$ scattering process in terms of $\mathcal{M}$ as an integral over a [Lorentz invariant phase space \(LIPS\)](#), the expression for which is given in Equation 2.5 [9]:

$$d\sigma = \frac{1}{(2E_1)(2E_2)|\vec{v}_1 - \vec{v}_2|} |\mathcal{M}|^2 d\Pi_{\text{LIPS}} \quad (2.4)$$

$$d\Pi_{\text{LIPS}} = (2\pi)^4 \delta^4(\sum p) \prod_{\text{final states } j} \frac{d^3 p_j}{(2\pi)^3} \frac{1}{2E_{p_j}} \quad (2.5)$$

This expression can be simplified for $2 \rightarrow n = 2$ scattering in the center-of-mass frame, as shown below in Equation 2.6 [9] [11]. The Lorentz-invariant phase space can be integrated, and many terms cancel, leaving a derivative over a solid angle Ω . The differential cross section is then defined in terms of the center-of-mass energy, E_{CM} , as well as the total momentum of the initial and final states. $\theta(x)$ is the Heaviside function, $\theta(x) = 1$ if $x > 0$, and 0 otherwise.

$$\left(\frac{d\sigma}{d\Omega}\right)_{\text{CM}} = \frac{1}{64\pi^2 E_{\text{CM}}^2} \frac{p_f}{p_i} |\mathcal{M}|^2 \theta(E_{\text{CM}} - m_3 - m_4) \quad (2.6)$$

A related quantity of interest is the decay rate of a single particle, Γ , sometimes also known as the width. The differential decay rate of a particle with energy E and mass m can also be written with respect to the Lorentz-invariant phase space given above in Equation 2.5 [9] [11]:

$$d\Gamma = \frac{1}{2E} |\mathcal{M}|^2 d\Pi_{\text{LIPS}} \quad (2.7)$$

The differential decay rate in Equation 2.7 will also simplify, for a two-body decay in the center-of-mass frame, to Equation 2.8. In the center-of-mass frame, the decaying particle will have zero momentum, and so p is the momentum of one of the emitted particles (with $-p$ the momentum of the other) [11]:

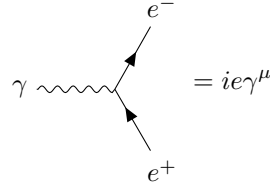
$$\left(\frac{d\Gamma}{d\Omega}\right)_{\text{CM}} = \frac{p}{32\pi^2 m^2} |\mathcal{M}|^2 \quad (2.8)$$

The above expressions can be used to calculate cross sections and decay rates, and predict or make comparisons to measurements made at colliders, provided the matrix element $\mathcal{M}$ can be determined. The next section discusses how this can be done using perturbation theory.

2.2.2 Feynman Diagrams

Feynman diagrams are a common way to visualize the interactions between particles. Each diagram, however, is actually a visual representation of a mathematical expression representing a probability amplitude. That probability amplitude is one term in the perturbative expansion of the matrix element $\mathcal{M}$ ⁷. The matrix element for a given state $\langle f | \mathcal{M} | i \rangle$ can be defined as the sum of the terms represented by all valid diagrams that can connect the initial state i to the final state f [9]. A Feynman diagram consists of three pieces: external legs, which are initial- or final- state particles; propagators, which are internal particles that do not leave the diagram, and vertices, where multiple propagators and/or external legs interact with each other. The vertices that are allowed in a given field theory are entirely determined by the structure of the Lagrangian describing that theory. For instance, for the QED Lagrangian given above in Equation 2.1, with one fermion, the allowed vertex is $\gamma \rightarrow e^+ e^-$ as given by $e\bar{\psi}\gamma^\mu\psi A_\mu$. In a Feynman diagram, that vertex would be drawn as shown in Equation 2.9 below, and in momentum space, the term $ie\gamma^\mu$ would enter the expression represented by the diagram in question [10].

⁷The math presented in this section is adapted from Chapter 13 of Schwartz [9] and Chapter 4 of Peskin and Schroeder [10], as well as Appendix A; note that Appendix A summarizes Feynman rules for all the quantum field theories presented in the book, and is a useful reference. This section gives a simplified overview of how to do calculations using Feynman diagrams, but for more detail, including derivations of these rules for this and other theories, interested readers should consult these references.



$$\gamma \text{ (wavy line)} \rightarrow \begin{matrix} e^- \\ e^+ \end{matrix} = i e \gamma^\mu \quad (2.9)$$

Fermion and photon propagators are then defined in Equations 2.10 and 2.11, respectively [10]:

$$\text{Feynman diagram of an electron propagator} = \frac{i(\not{p} + m)}{(p^2 - m^2 + i\epsilon)} \quad (2.10)$$

$$\text{Feynman diagram of a photon propagator} = \frac{-i g_{\mu\nu}}{(p^2 + i\epsilon)} \quad (2.11)$$

External fermion legs are then written using Dirac spinors, with $u^s(p)$ and $\bar{v}^s(p)$ for incoming fermions and anti-fermions, and $\bar{u}^s(p)$ and $v^s(p)$ for outgoing fermions and anti-fermions, respectively. Finally, external photons are defined using the photon polarization vector, $\epsilon_\mu(p)$ [10].

These expressions for propagators and external legs depend on the momentum of the particle in question. When writing down the probability amplitude represented by a Feynman diagram, momentum conservation must be imposed at each vertex, such that the sum of incoming momenta equals the sum of outgoing momenta. Diagrams can contain loops, where the momentum of the propagators comprising the loop is not fixed, and so a momentum-space integral in $\int \frac{d^4 p}{(2\pi^4)}$ must be performed. Additionally, loops involving fermions incur an overall factor of -1 , due to Fermi statistics. A symmetry factor also needs to be added to the probability amplitude if a diagram contains any interchangeable components: two indistinguishable that can be interchanged, for instance, means that the overall expression needs to be scaled by factor of $\frac{1}{2}$. Finally, note that Feynman diagrams have a crossing symmetry, which allows an incoming and outgoing leg to be swapped, although doing so does cause fermions to be replaced with anti-fermions and vice versa. Thus the diagrams $e^- \rightarrow \gamma e^-$ and $\gamma \rightarrow e^+ e^-$ are equivalent [10].

Using these rules, the matrix element $\mathcal{M}$ can be evaluated as the sum of all valid diagrams. Valid in this context means that the diagrams are both connected (the incoming and outgoing external legs are all connected to each other through propagators and vertices) and amputated (external legs are “cut” to remove any propagators or loops that do not involve another external leg). As an example, consider a $2 \rightarrow 2$ interaction between fermions through QED. The three simplest possible diagrams that could be drawn are shown in Figure 2.2. To refer to the total momentum in each

⁸The Feynman diagrams included in this thesis were drawn using the tikz-feynman package [18].

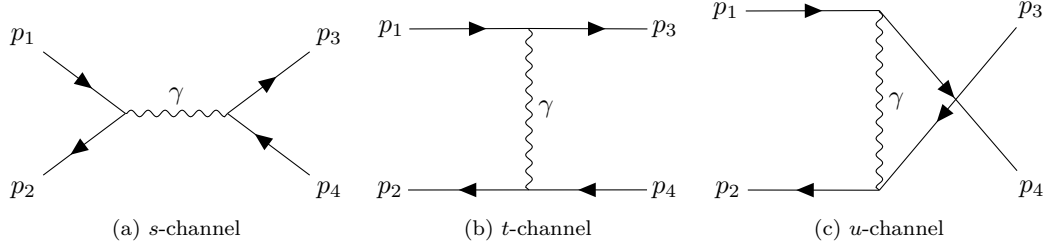


Figure 2.2: Example Feynman diagrams for a $2 \rightarrow 2$ scattering process in QED⁸; two incoming fermions interact via a single photon propagator and then produce two outgoing fermions. This can occur through three ways: the s -channel process (left), the t -channel process (center), or the u -channel process (right).

diagram, we define the Mandelstam variables $s = (p_1 + p_2)^2$, $t = (p_1 + p_3)^2$, and $u = (p_1 + p_4)^2$, and the diagrams are correspondingly labelled s , t , and u channels. The s -channel process is a collision between two incoming particles, while the t and u channels show scattering between two incoming particles, driven by the exchange of a boson [10]. This terminology is commonly used to classify Feynman diagrams, and in a collider experiment like the LHC, $\sqrt{s}$ is the variable used to define the center-of-mass collision energy available to produce new particles.

The Feynman rules for QED introduced in this section can be used to write down the probability amplitudes for each of the diagrams shown in Figure 2.2. At leading order in the QED vertex, the matrix element $\mathcal{M}$ would be the sum of the s , t , and u channel diagrams. However, there are more possible diagrams for the $2 \rightarrow 2$ process than just the leading order, or tree-level, ones. For instance, all three diagrams could be modified by the insertion of a fermion loop in the center, if the photon pair-produced two fermions, which then recombined back into a photon. This higher order diagram also contributes to the matrix element $\mathcal{M}$, as do even higher order diagrams, and so on, provided they obey the connected and amputated properties mentioned above. As diagrams with more and more vertices and more and more loops are added, it becomes increasingly difficult to perform calculations and evaluate integrals. These higher order diagrams also depend on more and more powers of the electromagnetic coupling $\alpha \approx 1/137$, though, and so their overall contribution to the matrix element becomes smaller and smaller.

Similar Feynman rules can be written down using the non-Abelian Lagrangian for QCD and weak interactions. The fact that the coupling $\alpha_s \approx 1$ at low energies means that it is difficult to use perturbation theory methods to describe the strong force. However, at the GeV scale and beyond, where $\alpha_s \approx 1/10$, it becomes possible to model QCD interactions using these methods.

2.2.3 Renormalization

One problem emerges when diagrams with loops are considered: the integrals that make up $\mathcal{M}$ will contain infinite divergences. The methods involved in dealing with these infinite divergences are known as *renormalization*. These divergences are unphysical, as loops are comprised of virtual particles, and so need to be removed or regulated in some way. One method for doing this is to introduce “counterterms” and work in renormalized perturbation theory [10]. In renormalized perturbation theory, additional terms are added to the Lagrangian that give rise to counterterm Feynman diagrams. These counterterms are included when calculating $\mathcal{M}$, and defined in such a way that they cancel the infinite divergences at every order. Doing so requires changing the Lagrangian to contain renormalized versions of the mass, charge, and electromagnetic coupling; these renormalized quantities receive corrections from their “bare” equivalents.

The renormalization group method, developed by Kenneth Wilson [19] is a powerful approach for dealing with these divergences. The idea is derived from atomic lattice calculations in condensed matter physics, where observed quantities should be independent of the spacing between atoms in the lattice, which can therefore be varied while performing calculations. The same principles can be applied to quantum field theory calculations. A renormalization scale μ can be introduced, and parameters like the couplings said to “run” as a function of this scale as higher and higher order correction are applied [9]. The running of the coupling constants is governed by the Callan-Symanzik equation [20] [21]. This is why, for instance, the QCD coupling varies from $\alpha_S \approx 1$ at low energy scales to $\alpha_S \approx 1/10$ at the GeV scale.

2.3 Electroweak Mixing and the Higgs Field

The theory of electroweak symmetry breaking was developed by Glashow [22], Weinberg, and Salam [23] in order to explain a fundamental problem with the weak interaction. The weak force appeared explainable by a $SU(2)$ non-Abelian gauge theory, but the gauge bosons associated with that theory would need to have masses in order to explain the range of the weak interaction. However, gauge bosons in such a theory are massless, like the gluons from $SU(3)$. To solve this problem, the idea of spontaneous symmetry breaking was introduced. If a field has some nonzero vacuum expectation value in a given direction, it can be said to “break” a symmetry. This breaking gives rise to additional degrees of freedom in a system, which are known as massless “Goldstone bosons” [24]. However, it is possible to absorb these extra degrees of freedom into the massless gauge bosons, which themselves have two transverse degrees of freedom, to create a massive gauge boson with a third longitudinal

degree of freedom. The Goldstone boson equivalence theorem has proven that this rearrangement, in which the gauge boson is said to “eat” the Goldstone boson, is valid. This mechanism by which a massless particle, in this case the W and Z bosons of the weak force, can gain mass through spontaneous symmetry breaking is known as the Higgs mechanism [25] [26] [27] [28]. The field used to break the spontaneous symmetry is the Higgs field, of which the scalar Higgs boson is the quantization. This section will review some of the math behind this and explore the properties of the W , Z , and h bosons⁹.

2.3.1 Electroweak Symmetry Breaking

Electroweak symmetry breaking does not just explain how the weak bosons have mass, but it also unifies the theory of electromagnetism and the weak force together. The Higgs field breaks a $SU(2) \times U(1)$ symmetry, which gives rise to the weak and electromagnetic forces. Note that this $U(1)$ symmetry is not actually the electromagnetic field pre-breaking; it’s another field, with another symmetry known as the weak hypercharge, Y . This field, and its gauge boson, are commonly labelled B , though the B boson does not exist after symmetry breaking. Similarly, the gauge bosons of the $SU(2)$ weak isospin symmetry here are labelled W^a for $a = 1, 2, 3$, and do not exactly correspond to the W and Z bosons post-breaking.

Symmetry breaking occurs due to the vacuum expectation value of a complex doublet, H , known as the Higgs multiplet, with hypercharge $Y = 1/2$. The Lagrangian for these fields is given below in Equation 2.12, in terms of the covariant derivative in Equation 2.13. Here, σ are the Pauli matrices, the generators of the group $SU(2)$, and g and g' the relative couplings for $SU(2)$ and $U(1)$ [9].

$$\mathcal{L} = -\frac{1}{4}(W_{\mu\nu}^a)^2 - \frac{1}{4}B_{\mu\nu}^2 + (D_\mu H)^\dagger(D_\mu H) + m^2 H^\dagger H - \lambda(H^\dagger H)^2 \quad (2.12)$$

$$D_\mu H = \partial_\mu H - \frac{1}{2}igW_\mu^a\sigma^a H - \frac{1}{2}ig'B_\mu H \quad (2.13)$$

The vacuum expectation value of the Higgs field, in terms of a new scalar h , is given below in Equation 2.14 [9]:

$$\langle\phi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + h \end{pmatrix} \quad (2.14)$$

⁹The math in this section is adapted from Chapter 29 of Schwartz [9]; Chapter 20 of Peskin and Schroeder [10], and Chapters 15 and 17 of Thomson [11].

Evaluating the covariant derivative for this vacuum expectation value, we find that the mass terms of the Lagrangian can be written as $\Delta\mathcal{L} = |D_\mu H|^2$ in Equation 2.15, where the terms depending on h are ignored for the time being [9] [10]:

$$\Delta\mathcal{L} = \frac{v^2}{8} \left((gW_\mu^1)^2 + (gW_\mu^2)^2 + (-gW_\mu^3 + g'B_\mu)^2 \right) + \mathcal{O}(h) \quad (2.15)$$

This expression is not diagonal, but it can be diagonalized into the following basis, which we label as the Standard Model W and Z , as well as the electromagnetic field, A_μ [10].

$$W_\mu^\pm = \frac{1}{\sqrt{2}} (W_\mu^1 \mp iW_\mu^2) \quad (2.16)$$

$$Z_\mu^0 = \frac{1}{\sqrt{g^2 + (g')^2}} (gW_\mu^3 - g'B_\mu) \quad (2.17)$$

$$A_\mu = \frac{1}{\sqrt{g^2 + (g')^2}} (g'W_\mu^3 + gB_\mu) \quad (2.18)$$

This diagonalized basis is a mass eigenstate, and it becomes possible to read the masses of the W and Z bases from the Lagrangian in terms of g' and g , as shown below. The electromagnetic field, however, remains massless, as expected [10]:

$$m_W = \frac{1}{2}gv \quad (2.19)$$

$$m_Z = \frac{1}{2}v\sqrt{g^2 + g'^2} \quad (2.20)$$

Instead of using the couplings g and g' , it is more convenient to introduce the weak mixing angle, $\tan(\theta_W) = g'/g$, and define the relationship between the (Z, A) and (W^3, B) bases in terms of this angle, as is done below in Equation 2.21 [10]. Then the masses of the Z and W are related to each other by $m_W = m_Z \cos(\theta_W)$. This angle is then easy to calculate by measuring the masses of the two bosons. Likewise, the electric charge can be written as $e = gg'/\sqrt{g^2 + (g')^2}$.

$$\begin{pmatrix} Z \\ A \end{pmatrix} = \begin{pmatrix} \cos \theta_W & -\sin \theta_W \\ \sin \theta_W & \cos \theta_W \end{pmatrix} \begin{pmatrix} W^3 \\ B \end{pmatrix} \quad (2.21)$$

The vacuum expectation value, Equation 2.14, can be induced by the potential given in the renormalizable Lagrangian below in Equation 2.22 [10] [9]:

$$\mathcal{L}_{\text{Higgs}} = |D_\mu \phi|^2 + \mu^2 \phi^\dagger \phi - \lambda (\phi^\dagger \phi)^2 \quad (2.22)$$

When plugging the vacuum expectation value in, we see that the scalar field h is quantized with a particle with mass $m_h = v\sqrt{2\lambda}$. This particle is the physical Higgs boson, which does not have hypercharge or weak isospin (unlike the Higgs multiplet field). The $O(h)$ terms in Equation 2.15 describe the interactions between the Higgs boson and the W and Z gauge bosons.

It is not yet known if a similar symmetry breaking argument can be used to unify the electroweak force with the strong force, creating a “grand unified theory”. However, this is a topic of active research.

2.3.2 W and Z Bosons

The above writeup sketched out the arguments behind electroweak symmetry breaking, and explained how the weak boson masses and Higgs boson emerge. These particles also interact with the fermions that comprise the rest of the Standard Model, and so the remainder of this section focuses on describing those interactions. First, the W and Z bosons interact with other particles based on their weak isospin and weak hypercharge. The weak hypercharge Y and third component of weak isospin I_W can be related to the electric charge Q , since the photon is a linear combination of W^3 and B fields¹⁰:

$$Q = Y + I_W^{(3)} \quad (2.23)$$

It is important to note that left-handed and right-handed chiral states behave differently under the weak interaction. The $SU(2)$ gauge bosons *only* couple to left-handed fermions (and right-handed anti-fermions), and do not couple at all to right handed right fermions (and left-handed anti-fermions). As a consequence of this, left-handed fermions are written as a doublet, for instance the combination of an electron e and electron neutrino ν_e as $(\nu_e^e)_L$, or the combination of a strange and charm quark as $(s^c)_L$. Right-handed chiral states, on the other hand, are written as singlets, such as e_R , s_R , etc., and have weak isospin $I_W^{(3)} = 0$. Note that so far only left-handed neutrinos have been observed, but right-handed neutrinos are not forbidden in the Standard Model and are predicted by a number of theories. The W boson then couples to these doublets, which allows the W to pair produce a charged lepton and neutrino of the same generation, or a pair of up-type and down-type quarks from the same generation [9].

¹⁰Note that the linear combination can be written with a different scaling factor; the notation adopted here is used in Schwartz [9], but Thomson (for instance) writes $Q = 2(Y + I_W^{(3)})$ and adjusts the definition in Table 2.1 accordingly [11].

Particles	Q	Y_L (Left-Handed)	Y_R (Right-Handed)
Neutrinos (ν_e, ν_μ, ν_τ)	0	$-1/2$	0
Leptons (e, μ, τ)	$1/2$	$-1/2$	-1
Up-Type Quarks (u, c, t)	$2/3$	$1/6$	$2/3$
Down-Type Quarks (d, s, b)	$-1/3$	$1/6$	$-1/3$

Table 2.1: The charge and weak hypercharge for each type of fermion in the Standard Model: the neutrinos, the charged leptons, and the up- and down- type quarks. Because the weak hypercharge $Y = Q - I_W^{(3)}$, and because right-handed particles do not interact via the weak force, the weak hypercharge for the left- and right- handed versions of these particles will differ. They are therefore written as Y_L and Y_R , respectively.

On the other hand, all fermions can interact with the B gauge boson. The hypercharge of each type of fermion is given above in Table 2.1; note that it differs between left- and right- handed fermions but is the same for all three generations. That means both left- and right- handed states will interact with the Z , as it is also a linear combination of the W^3 and Z states, albeit with different coupling strengths (since the right-handed coupling will only depend on hypercharge, while the left-handed coupling will also depend on $I_W^{(3)}$). Also note that the Z will couple differently to charged leptons and neutrinos, because of the different hypercharges. Formally, the “weak neutral current” J_μ^Z can be written as follows in terms of the electromagnetic current, J_μ^{EM} , and the $SU(2)$ generator $T^3 = \frac{1}{2}\sigma^3$ [9].

$$J_\mu^Z = \frac{1}{\cos \theta_W} (J_\mu^3 - \sin^2 \theta_W J_\mu^{\text{EM}}) \quad (2.24)$$

$$J_\mu^3 = \sum_i \bar{\psi}^L \gamma^\mu T^3 \psi_i^L \quad (2.25)$$

$$J_\mu^{\text{EM}} = \sum_i Q_i (\bar{\psi}^L \gamma^\mu \psi_i^L + \bar{\psi}^R \gamma^\mu \psi_i^R) \quad (2.26)$$

These terms lead to the allowed Feynman diagrams for the Z boson, in which it can pair-produce any pair of fermions.

2.3.3 Higgs Boson

The Higgs field also couples to the fermionic sector. As h is a scalar, Yukawa theory, which governs interactions between scalars and fermions, can be used to describe this coupling. This links the left-handed doublet and right-handed singlets together, and will generate the same masses for both without violating symmetry. Equation 2.27 shows an example of this coupling for the electron, where $E_L = (\nu_e)_L$, and h.c. is the Hermitian conjugate [9].

$$\mathcal{L}_{\text{Yukawa}} = -\lambda_e \bar{E}_L H e_R + \text{h.c.} \quad (2.27)$$

Electroweak symmetry breaking will give the electron a mass, $m_e = \frac{1}{\sqrt{2}}\lambda_e v$, and the interaction between the electron and the scalar Higgs field will be proportional to this mass, as shown below in Equation 2.28 [10]. This is why, in addition to electroweak symmetry breaking, the Higgs boson is said to “give mass” to fundamental particles.

$$\mathcal{L}_e = -m_e \bar{e} e \left(1 + \frac{h}{v}\right) \quad (2.28)$$

The same interactions lead to mass terms for the quarks, however there is a complication. The Yukawa interactions between the Higgs and the up-type and down-type quarks are not diagonal. Diagonalizing them requires moving away from the flavour basis, through which the quarks interact via the weak force, and into a new mass basis. This diagonalization is done by means of a 3x3 matrix, $V = U_u^\dagger U_d$, called the [Cabibbo-Kobayashi-Maskawa \(CKM\)](#) matrix [29]. Then the up-type quarks translate from the mass basis into the flavour basis through $u_{\text{flavour}}^i = U_u^{ij} u_{\text{mass}}^j$, and similarly for the down-type quarks, $d_{\text{flavour}}^i = U_d^{ij} d_{\text{mass}}^j$ [10]. The quarks are generally referred to in the mass basis, with their couplings to the W and Z bosons modified accordingly through this mixing matrix. The elements of the CKM matrix need to be measured experimentally. Note that if the measured values of the 3x3 matrix were found to not be unitary, it would imply the existence of additional generations of quarks, though there is currently no evidence of this.

This problem does not arise with couplings to the charged leptons, where the mass terms can be written in a diagonal way. Therefore, there is no corresponding lepton mixing matrix. And finally, while neutrinos have been experimentally shown to have mass, in the Standard Model neutrinos are assumed to be massless, meaning they do not couple to the Higgs. Neutrinos are known to oscillate from one state to another, and another mixing matrix, the [Pontecorvo–Maki–Nakagawa–Sakata \(PMNS\)](#) matrix, governs these oscillations between flavour and mass states [30]. The non-Higgs mechanism that gives rise to neutrino masses is still unknown.

The Higgs boson then interacts with the vector bosons, through electroweak symmetry breaking, as well as all massive fermions, through diagrams of the form $h \rightarrow W^+ W^-$, $h \rightarrow Z Z$, and $h \rightarrow f \bar{f}$, as well as their crossing symmetry equivalents. A particle consistent with the Standard Model Higgs boson was discovered in 2012 by the ATLAS and CMS collaborations with $m_h = 125 \text{ GeV}$ [6] [7], and this large mass means that it is slightly difficult to produce the Higgs directly. The most common production mechanism is known as “gluon gluon fusion”, where a pair of very high energy gluons

creates a Higgs boson through a top quark loop. The analysis presented in this thesis is interested in the Higgs produced via “vector boson fusion”, via either the $ZZ \rightarrow h$ or $W^+W^- \rightarrow h$ processes [31]. More details on different Higgs production mechanisms and why one might be preferred over the other for experimental reasons can be found in Section 6.1.

2.4 Dark Matter

The Standard Model then contains the theory of QCD, which governs the strong interaction, electroweak theory as discussed above, which governs the electromagnetic and weak forces, and the Higgs field, which gives masses to the W and Z gauge bosons, the leptons, and the quarks. This theory is very successful, and calculations of decay rates and interaction cross sections performed using quantum field theory and renormalization have been shown to agree very well with experimental evidence. However, the theory is not perfect. While the Higgs mechanism explains how particles get their mass, the actual values of the masses are parameters determined experimentally. It is not known why the top quark, for example, should be so much heavier than the bottom quark. Additionally, the loop corrections to the Higgs mass are quite large, which means that some large cancellation mechanism is needed for the Higgs to have its observed mass of 125 GeV, something that is known as the hierarchy problem. These phenomena raise questions of “naturalness”: that is, is there a deeper reason why these masses and parameters take on the values that they do.

In addition to these sorts of questions, the Standard Model also does not explain a number of observed phenomena. It was noted above that neutrinos in the Standard Model were originally postulated as a massless particle, and yet they appear to have mass. Also, there is no theory of quantum gravity embedded in the Standard Model—there is no known graviton gauge boson that governs interactions due to gravity—as there are singularities that appear at the quantum scale when trying to reconcile general relativity with quantum mechanics. And there is also no explanation for “dark matter”, something for which there is a wide variety of experimental evidence from astrophysics [3]. Dark matter refers to matter which does not interact electromagnetically, meaning it is “invisible” (hence the title of this thesis) and cannot be seen, but does have mass and interact gravitationally, as opposed to “baryonic” or “luminous” matter comprised of protons, neutrons, and other Standard Model particles that does interact electromagnetically¹¹. The only Standard Model particle which fits this description is the neutrino, but astrophysical evidence suggests that

¹¹“Baryonic” in this astrophysical context refers not just to baryons in the sense of the Standard Model (a bound state of three quarks), but entire atoms— which, for instance, includes orbiting electrons, which notably are not actually baryons in a particle physics sense.

the neutrino alone would not explain the apparent amount of observed dark matter. That implies there might exist one or more new dark matter particles.

2.4.1 Astrophysical Evidence

One of the first sources of evidence for dark matter came from measurements of the rotation curves of galaxies. By observing hydrogen emission with a radio telescope at varying points along the radius of a galaxy, the speed at which the galaxy is rotating can be measured as a function of the radius¹². According to Newtonian mechanics, the speed a galaxy is rotating as a function of radius should go as $v(r) \propto \sqrt{M(r)}/r$. Measuring the rotation curve therefore allows the mass distribution to be measured as a function of the radius. In the 1970s, Vera Rubin measured the rotation curves of 21 galaxies and found that they did not appear to follow this functional form [2]. The mass distribution of a galaxy appeared to extend far beyond the visible part of the galaxy in the night sky. This implies that either there is something wrong with the Newtonian theory of gravity on a galactic scale or that a galaxy contains more mass beyond the edge of its visible structure that is invisible.

Subsequent evidence using gravitational lensing methods have found more evidence for dark matter [3]. Observations have made of galaxies or galaxy clusters that appear to be colliding. These observations can be performed by comparing visible light recorded by ordinary telescopes to measurements of X-rays emitted from the stars and gases that make up the luminous parts of the galaxy clusters. Mass distributions can be inferred due to gravitational lensing, and these studies have shown spatial separation between the mass of the luminous parts of the clusters recorded via X-ray astronomy and the mass of the cluster determined via ordinary telescopes. The “bullet cluster” [3] is the most famous such event, but there have been many others with similar structure, such as the “train-wreck cluster” [32]. One explanation for this phenomenon is that galaxies have dark matter “halos” surrounding them, and that the particles comprising these halos interact much more weakly with each other than the baryonic matter that makes up the visible parts of galaxies. Then, when galaxies collide, the dark matter halos mostly pass through each other, while the baryonic matter interacts, causing the visible parts of the cluster to slow down relative to the halo.

Additional evidence for the existence of dark matter comes from measurements of the [Cosmic Microwave Background \(CMB\)](#), through experiments like Planck [33]. By fitting cosmological models to the observed power spectrum of the CMB, it is possible to constrain the amount of dark matter in

¹²I did this experiment as an undergraduate in an advanced lab class at Johns Hopkins, using a simple radio telescope constructed on the roof of our physics building; it was quite exciting to be able to reproduce this result.

the universe, known as the relic density. These measurements suggest that only 5% of the universe’s energy density is comprised of ordinary or baryonic matter, made from the known Standard Model particles introduced in Section 2.1, while roughly 25% comes from dark matter. The remaining 70% appears to be due dark energy, the nature of which is also unknown. These measurements can also be used to set limits on the rates at which dark matter can self-interact and annihilate. Together with the relic density, this provides a constraint on any potential particle model for dark matter that needs to be taken into account.

More recently, there have been measurements of the rotation curves of galaxies that may lack dark matter halos altogether [34]. Any theory of modified Newtonian gravity would not just need to explain why galaxies have modified rotation curves, but why some don’t, which makes it trickier to build such a model. This isn’t necessarily impossible, but combined with the gravitational lensing measurements and evidence from the CMB, it seems to strengthen the case for dark matter existing.

2.4.2 Dark Matter Candidates

If dark matter does exist, the natural question is: what is it? Any proposed dark matter candidate or model needs to satisfy the relic density measurements mentioned above in order to fully explain the phenomenon. In fact, a huge variety of dark matter models have been proposed over an incredibly wide energy range. Wave-like dark matter models include proposed particles like axions, which were introduced by Peccei and Quinn [35] [36] to solve another problem with the Standard Model¹³ or “fuzzy” dark matter, an ultra-light particle with mass potentially as low as 10^{-22} eV [37]. On the other end of the spectrum, composite dark matter models have been proposed at the Planck scale, 10^{28} eV, or even higher, a range of more than 40 orders of magnitude. Both of these extremes are far beyond what could be searched for at a collider experiment like LHC. It might be possible to probe dark matter models with energies in the middle of this range, on the scale of weak interaction (1 GeV to 1 TeV). One of the most popular proposed dark matter models is that of [Weakly Interacting Massive Particles \(WIMPs\)](#), particles which would have masses roughly around this range and interact via the weak force as well as through gravity [38] [12].

A number of searches for WIMPs, as well as other dark matter candidates like axions, are underway. In general, these searches can be done in one of three ways. First, “direct” detection can look for evidence of Standard Model particles interacting directly with dark matter particles. For WIMPs, this could occur through the weak force by means of weak nuclear recoil, in which a dark

¹³Specifically, to explain why the strong interaction appears to preserve charge-parity symmetry, known as the “strong CP problem”.

matter particle might “bump” against a nucleus. Experiments like LUX [39] or PandaX [40] look for evidence of this WIMP-nuclear recoil in liquid xenon. Then, there are “indirect” experiments, which look for evidence of dark matter self-interaction or annihilation that might produce an excess of Standard Model particles in astrophysical processes [41]. Dark matter is theorized to self-interact quite weakly compared to the Standard Model, but it might still be possible to observe evidence of this at an experiment like Fermi [42]. And finally, if dark matter can produce Standard Model particles, it might also be possible to search for the production of dark matter from the Standard Model. This is the type of search that can be performed at a collider like the LHC.

2.4.3 Higgs Portal Dark Matter

One possible way in which dark matter might interact with the Standard Model is through the Higgs field, a class of models known as “Higgs portal” dark matter [43] [38]. Since most massive Standard Model particles, perhaps with the exception of the neutrino, get their mass from their Yukawa coupling with the Higgs field, it is reasonable to think the same might be true of a massive dark matter candidate. This idea is somewhat model-independent, in that while the Higgs portal dark matter candidate could be a WIMP, it does not have to be, it just has to couple with the Higgs. If it does, it is possible that a Higgs decaying “invisibly” to dark matter could be observed at the LHC. This is the class of models that we are searching for in the analysis presented in Chapter 6.

A variety of scenarios can be considered, but a simplified model can be written in which the Higgs field couples directly to one new Standard Model particle. The form of the Lagrangian describing this interaction depends on whether the new particle is a scalar S , Majorana fermion f (meaning that it is its own antiparticle, as opposed to a Dirac fermion), or vector V [44]:

$$\mathcal{L}_S = -\frac{1}{2}m_S^2 S^2 - \frac{1}{4}\lambda_S S^4 - \frac{1}{4}\lambda_{hSS} H^\dagger H S^2 \quad (2.29)$$

$$\mathcal{L}_V = \frac{1}{2}m_V^2 V_\mu V^\mu + \frac{1}{4}\lambda_V (V_\mu V^\mu)^2 + \frac{1}{4}\lambda_{hVV} H^\dagger H V_\mu V^\mu \quad (2.30)$$

$$\mathcal{L}_f = -\frac{1}{2}m_f \bar{\chi}\chi - \frac{1}{4}\frac{\lambda_{hff}}{\Lambda} H^\dagger H \bar{\chi}\chi \quad (2.31)$$

These particles will gain corrected masses after electroweak symmetry breaking in terms of v , from the Higgs potential’s vacuum expectation value [44]:

$$M_s^2 = m_s^2 + \frac{1}{2}\lambda_{hSS}v^2 \quad (2.32)$$

$$M_v^2 = m_v^2 + \frac{1}{2}\lambda_{hVV}v^2 \quad (2.33)$$

$$M_f^2 = m_f^2 + \frac{1}{2}\frac{\lambda_{hff}}{\Lambda}v^2 \quad (2.34)$$

Decay rates for the $h \rightarrow SS$, $h \rightarrow VV$, $h \rightarrow \chi\chi$ interactions involving the scalar Higgs boson can then be calculated, using the methods described earlier in this chapter, in terms of $\beta_x = \sqrt{1 - 4M_x^2/m_h^2}$ [44]. As long as these masses M_x are less than half the mass of the Higgs boson, this process should be able to occur.

$$\Gamma_{h \rightarrow SS} = \frac{\lambda_{hSS}^2 v^2 \beta_S}{64\pi m_h} \quad (2.35)$$

$$\Gamma_{h \rightarrow VV} = \frac{\lambda_{hVV}^2 v^2 m_h^2 \beta_V}{256\pi M_v^4} \left(1 - 4\frac{M_v^2}{m_h^2} + 12\frac{M_v^4}{m_h^4} \right) \quad (2.36)$$

$$\Gamma_{h \rightarrow \chi\chi} = \frac{\lambda_{hff}^2 v^2 m_h \beta_f^3}{32\pi \Lambda^2} \quad (2.37)$$

Limits on the invisible branching ratio, $\mathcal{B}_{h \rightarrow \text{inv.}} = \Gamma_{h \rightarrow \text{inv.}}/\Gamma$, can be set by performing a collider search at an experiment such as ATLAS. These limits can then be interpreted for each of the three models using the above expressions to set limits on the mass and interaction strength with the Higgs boson. This can then be used to set a limit on the spin-independent WIMP-nucleon cross section, from the interaction of this particle with a nucleus. The direct detection experiments report their limits in this form, and so this allows for a direct comparison [44].

It should be noted that the Lagrangians presented here are effective field theories, and so the vector and fermionic models are not UV-complete. In particular, the use of the vector model presented here in order to set realistic limits had been questioned [45]. It has been shown, however, that the vector model presented here can be taken as the effective field theory limit of UV-complete models with a more complex dark matter sector [46]. For instance, one possible interpretation is that the vector dark matter candidate is a “dark photon”, the gauge boson of a new $U(1)$ gauge group, which gets mass by means of a new “dark Higgs”. Limits on such a model will then depend on the mass of the dark Higgs, which is a free parameter. Because the simplified vector model has been shown to be a valid simplification of such a theory, it will be used along with the scalar and fermionic models when this interpretation is presented in Section 9.5 [47].

CHAPTER 3

LHC and the ATLAS Detector

To try and answer questions like the nature of dark matter, particle physics experiments are needed. The work presented in this thesis was done using the LHC [5], a proton-proton collider, and the ATLAS experiments [1], one of the four main detectors at the LHC. This chapter gives an introduction to both. Section 3.1 describes the operation of the LHC itself, and also introduces several concepts like pile-up and luminosity that are used to describe the rates of proton-proton interactions and the amount of data collected. Section 3.2 then describes the various components which make up the ATLAS detector itself, and explains the principles behind how they work. Finally, Sections 3.3 and 3.4 explain how data collected at the ATLAS detector is read out using a triggering system and processed using various reconstruction algorithms that are used to identify particles.

3.1 The Large Hadron Collider

The LHC [5] is a particle accelerator and collider at CERN, the European Organization for Nuclear Research. Located along the French-Swiss border near the Swiss canton of Geneva, the LHC is currently (as of 2021) the world's most powerful particle accelerator. Superconducting magnets and a series of resonant frequency cavities are used to guide two beams of hadrons along a circular 26.7 km tunnel and accelerate them to very high energies. These beams are then channeled into four interaction points spaced out along the ring, where bunches of particles are collided and specialized particle detectors observe the collisions [5].

The tunnels used by the LHC were first dug for the Large Electron-Positron Collider [48], which ran from 1989 until 2000 and reached a center-of-mass e^+e^- collision energy of 209 GeV [49]. The LHC was then constructed and installed, and first began operating in 2009. While designed to

be capable of accelerating proton beams to 7 TeV, and therefore collide protons at center-of-mass energies of $\sqrt{s} = 14$ TeV, during run 1 of the LHC technical issues limited the center-of-mass energy to only 8 TeV. Run 1 lasted from 2009 to 2013, and upgrades were then performed over the next two years to increase the center-of-mass collision energy. Run 2 lasted from 2015 to 2018 with proton-proton collisions at an energy of $\sqrt{s} = 13$ TeV. The LHC has also been used to perform runs with beams of heavy ions, like lead (Pb), but this chapter will focus on describing proton-proton operations.

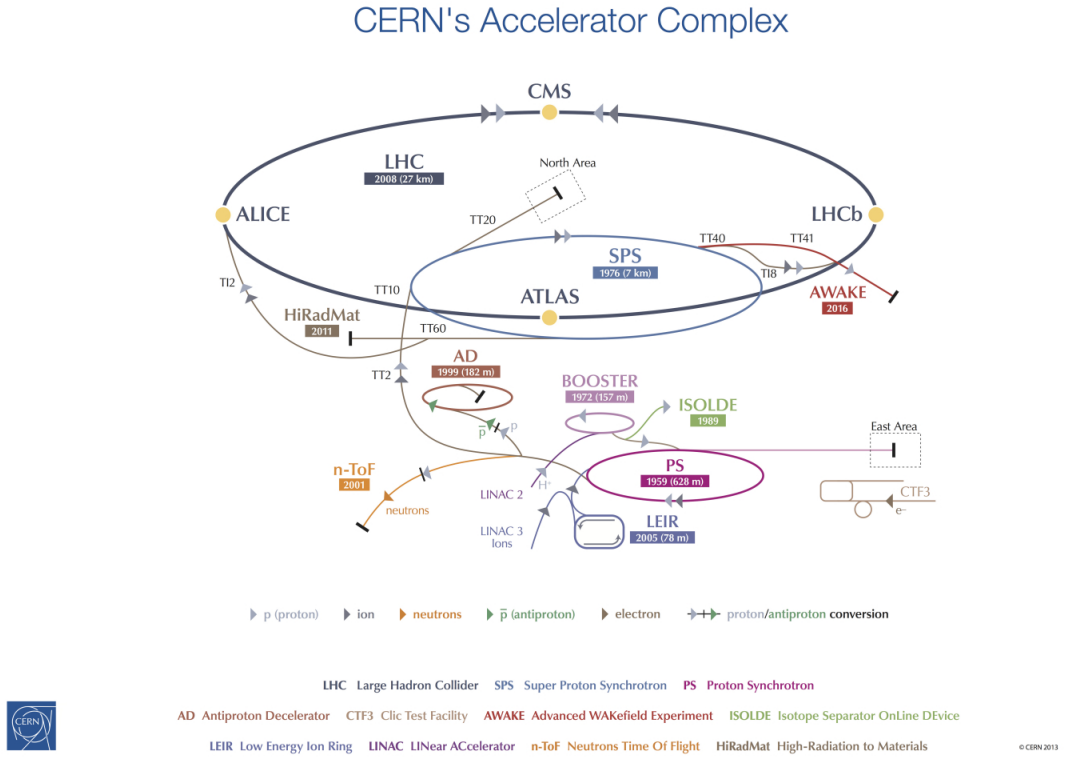


Figure 3.1: Diagram of the CERN accelerator complex, showing the flow of protons from LINAC2 through the PS and SPS into the Large Hadron Collider [50]. Other experiments and facilities are also labelled.

The protons used by the LHC are first extracted by applying an electric field to hydrogen gas to create free protons. Then, a chain of accelerators is used to boost these protons to relativistic speeds before injecting them into the LHC. Figure 3.1 shows a diagram of the different facilities and accelerators at CERN, and how they are connected. First, a linear accelerator known as LINAC2¹⁴ is

¹⁴For future LHC runs, starting with run 3, LINAC2 has been decommissioned and replaced with the new LINAC4

used to first boost protons to 50 MeV. Protons are then passed through a series of three synchrotrons to accelerate them to increasingly high energies: the Proton Synchrotron Booster, to reach 1.4 GeV; the Proton Synchrotron, to reach 25 GeV, and finally the Super Proton Synchrotron, to reach 450 GeV [5]. These synchrotrons were originally built in the 20th century and used for a previous generation of particle physics experiments; data collected from experiments on SPS, for instance, lead to the discovery of the Z and W bosons. They have since been upgraded for use as feeders to produce proton beams for the LHC.

Once the proton beams have been injected into the LHC, they are accelerated from 450 GeV to energies as high as 7 TeV. To accelerate the beams, a combination of resonant-frequency (RF) cavities and superconducting magnets are employed [52]. Both dipole and quadrupole magnets use niobium-titanium (NbTi) superconductors to steer and focus the beams, respectively. The LHC employs a “twin-bore” structure, in which two separate rings— in which two separate beams can orbit, in opposite directions— occupy the same physical tunnel, thus leading to a complex magnetic field structure. As the beams orbit around the ring, they pass through eight resonant field cavities, which operate at frequencies of 400 MHz. The beams are accelerated by the oscillation of the electric field inside these cavities. To keep the beams aligned, the magnetic field strength varies from 0.54 T at insertion to 8.33 T at 7 TeV. To reach this magnetic field strength, a cryogenic system based on superfluid helium is employed to keep the magnets at a maximum operating temperature of 1.9 K. The huge range over which the magnetic field must vary illustrates the need for the insertion chain, as it would be prohibitively difficult to boost the particles in a single step [5].

The beams inserted into the LHC do not contain single protons. Rather, a beam is constructed from a series of proton bunches, consisting of up to 1.7×10^{11} protons per bunch. These bunches are injected at intervals of 25 ns in a pattern known as a “bunch train”. Figure 3.2 shows an illustration of the original pattern the LHC was designed to use, but the bunch train has varied over time. More recently, alternate filling schemes have been tested in order to increase the rate at which bunches collide and thereby gather more data. With bunches every 25 ns, it would take 3564 bunches to “fill” the LHC ring, a period of time known as an “orbit” of the accelerator. However, technical limitations in the injection chain and in the LHC itself mean that about a fourth of these slots must be kept empty, so only 2808 slots contain protons per beam. Allowing for calibration time, using the original pattern it takes about 16 minutes to fill the LHC with one orbit, and then about 20 minutes to accelerate the bunches in that orbit to 7 TeV [5].

The two beams intersect at four interaction points along the ring, which contain the four main [51]. LINAC3 is used for heavy ion injection.

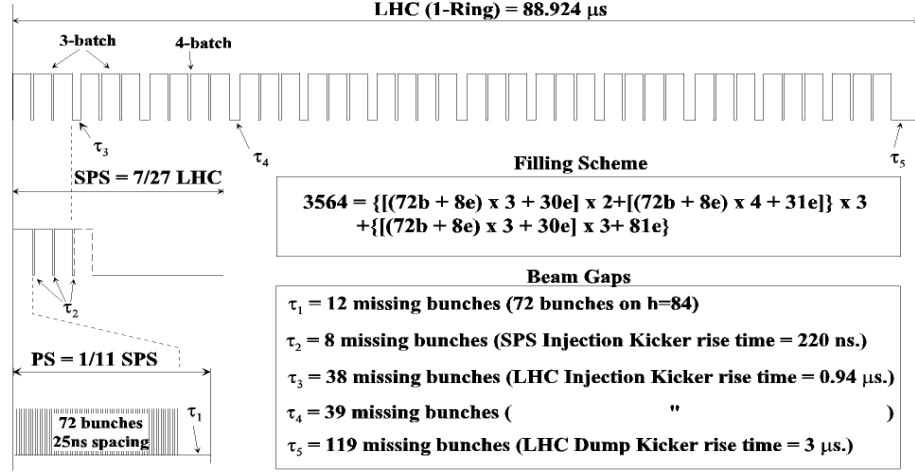


Figure 3.2: Diagram of the proton bunch structure used to fill the Large Hadron Collider [5]. The LHC has 3564 “slots” for bunches, but technical limitations mean that 756 of those slots are empty. Note that in the equation describing the filling scheme, $30e = \tau_3 - \tau_2$ and $31e = \tau_4 - \tau_2$, respectively, to prevent double-counting empty bunches. Similarly, the final 81 empties is due to the LHC dump kicker rise and is actually $\tau_5 - \tau_3 - \tau_2$.

LHC experiments: ATLAS (at point 1) [1], ALICE (at point 2) [53], CMS (at point 5¹⁵) [4], and LHCb (at point 8) [14]. Magnetic fields are used to bring bunches from the two beams together in an interaction known as a “bunch crossing”. Because the bunches are spaced out in 25 ns intervals, a bunch crossing can occur every 25 ns, or at a rate of 40 MHz. In these bunch crossings, multiple protons from the two bunches can interact with each other, meaning the actual number of proton-proton collisions per bunch crossing is variable. The number of interactions per crossing, μ , is also known as pile-up, and can be partially controlled by adjusting the magnetic fields to change the angle at which the beams intersect. As μ increases, the amount of activity recorded by the detectors also increases, meaning that it becomes harder to identify which specific interaction produced a given observed particle [54].

The numbers of bunch crossings, particles per bunch, and interactions per bunch can be used to calculate a quantity known as the *instantaneous luminosity*, $\mathcal{L}$. The instantaneous luminosity can be defined as the number of collisions, or events, N , that occur within the interaction cross section, σ [11]. As the experiment runs, the number of collisions will vary with time, and so $\mathcal{L}$ will not be constant. Therefore, the *integrated luminosity*, $L = \int \mathcal{L} dt$, is often used to describe

¹⁵There are eight total access points along the ring, but only four have beam crossings. Points 3, 4, 6, and 7 are used for other technical purposes. For instance, point 6 is the “beam dump”, where the beams are extracted and deposited into a shielded block of carbon, concrete, and steel [5].

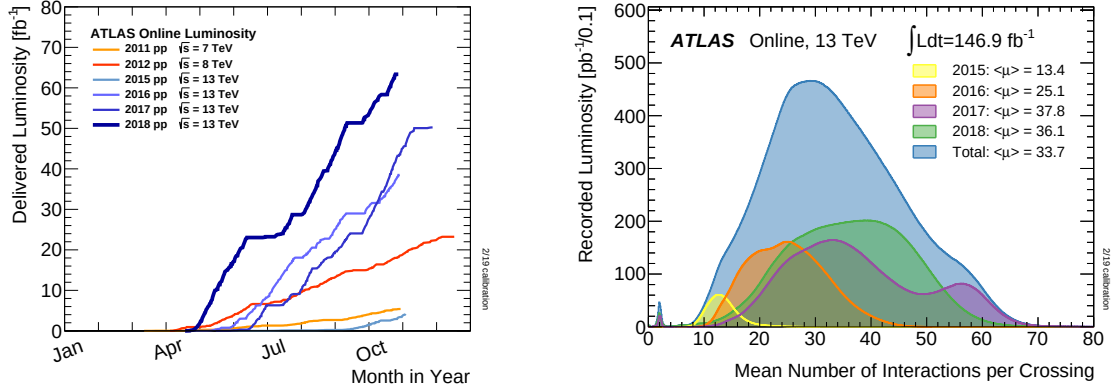


Figure 3.3: Plots showing the integrated luminosity of proton-proton collisions collected by the ATLAS detector [55]. On the left, recorded luminosity is shown as a function of time for both runs 1 and 2 (2011-2018). On the right, recorded luminosity is shown as a function of the number of interactions per bunch crossing (μ) for just run 2 (2015-2018).

the total amount of data collected by a given experiment. Luminosity can be determined either by using online monitoring, in which measurements of the profiles of each beam are recorded during data-taking, or offline, by performing an analysis to count the number of events detected for a process with known cross section. The ATLAS experiment, in particular, was designed to achieve an instantaneous luminosity of at least $10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ [1], which has been met and exceeded as of run 2. Figure 3.3 summarizes the integrated luminosity collected by the ATLAS experiment, both as a function of time and as a function of pile-up, recorded online using data from a dedicated luminosity detector. As of 2018, when run 2 data taking ended, ATLAS has collected 139 fb^{-1} of data [56]. The rest of this chapter focuses on ATLAS specifically in more detail, and describes how that data was collected and some of the steps done to process it for use in physics analyses.

3.2 The ATLAS Detector

ATLAS is a multi-purpose particle detector located at Point 1, one of the four LHC interaction points¹⁶ [1]. The detector was designed to observe particles produced by both high-energy proton-proton collisions as well as heavy ion collisions; as mentioned previously, this chapter focuses on proton-proton operations. ATLAS consists of a series of layered sub-detectors, which form a cylindrical shell around the interaction point. Figure 3.4 shows a cut-away diagram of the detector, with the many subsystems labelled; another view in the transverse plane can be seen in Figure 3.5. When

¹⁶Unlike the other three experiments, point 1 is conveniently located at CERN's main Meyrin site, on the French-Swiss border.

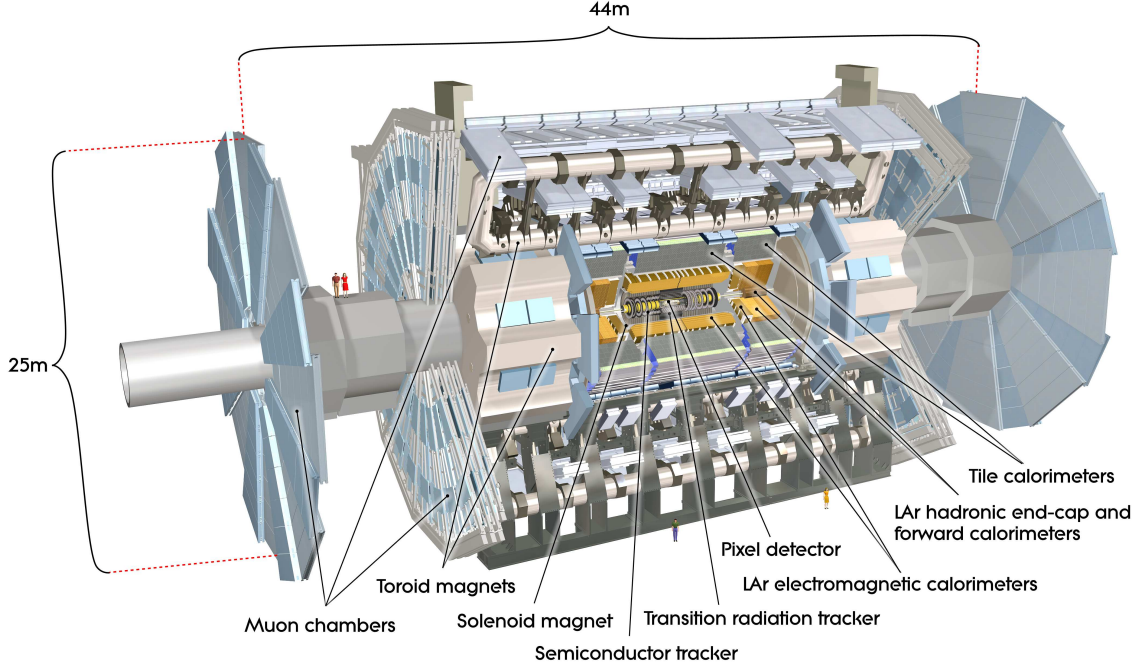


Figure 3.4: General cut-away view of the [ATLAS](#) detector, with major components labelled [1]. The Pixel Detector, Semiconductor Tracker, and Transition Radiation Tracker form the Inner Detector; the LAr and Tile calorimeters sit in the center of the detector, and then the muon chambers surround them in the outermost level.

particles are produced in a collision, they first pass through the **Inner Detector**, which sits closest to the beam. The Inner Detector records the tracks left by charged particles as they pass through this region. Then, in the middle of the detector, most particles decay and deposit their energy into the electromagnetic or hadronic **calorimeters**. However, some particles—most notably muons—do not, and so the outermost layers of ATLAS form the **Muon Spectrometer** in order to (as the name implies) measure the energy and trajectory of muons as well. These three systems are discussed in more detail below.

As ATLAS forms a cylinder, cylindrical coordinates are often used to describe its layout. The z axis is oriented along the beamline, with the transverse $r - \phi$ plane orthogonal to it. The origin of this coordinate system is the interaction point, so half of the detector has positive z and half negative z ; these halves are often known as “Side A” and “Side C” respectively. The radius r defines the distance from the beamline; in the $r - \phi$ plane, the x axis points to the center of the LHC ring and the y axis points upward into space. In general, ATLAS is designed to be symmetric both with respect to $\pm z$ and along the azimuthal angle ϕ . The full detector spans 44 m in the z direction, and

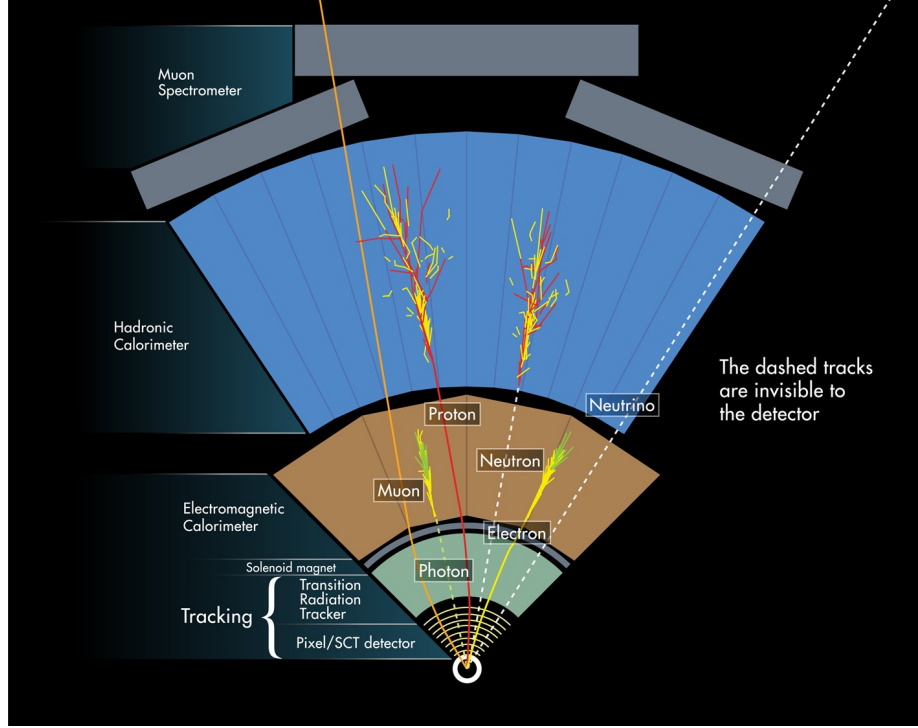


Figure 3.5: Illustration of different types of particles passing through the different layers of the ATLAS detector [57]. Charged particles leave tracks in the inner detector, while neutral particles do not. Electrons and photons then deposit all their energy in the electromagnetic calorimeter, while hadrons decay in the hadronic calorimeter. Muons make it through the calorimeters, and so the muon spectrometer is used to measure their presence. Neutrinos freely pass through the entire detector; their presence must be inferred by a transverse momentum imbalance.

has a radius of $r = 12.5 \text{ m}$ [1].

While these coordinates are useful to describe the physical layout of the detector, a spherical coordinate system is often used to describe objects measured within it. In this spherical system, the polar angle, θ , is generally not used directly. Instead, the pseudorapidity $\eta = -\ln \tan \frac{\theta}{2}$ is used as a second angular coordinate to measure how far “forward” an object is relative to the $z = 0$ axis [11]. In an environment where the energies of objects are much greater than their masses ($E \gg m$), the pseudorapidity serves as an approximation for the Lorentz-invariant rapidity. Since a combination of energy and trajectory is measured by the different components of the detector, objects measured by ATLAS are often then expressed as a four-momentum. This is usually written using η , ϕ , the total energy of the particle, and the component of the momentum in the transverse $x - y$ plane, generally written p_T . In this coordinate system, the distance between two objects can be written as $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ [1] [54].

In addition to the primary sub-detectors described here, ATLAS contains several other specialized systems, especially in the “forward” region [1]. For example, the LUCID detector sits 17 m from the interaction point [58]. LUCID is one of ATLAS’s luminosity monitors, and is used to produce the online luminosity plots showed above in Figure 3.3. Additionally, in addition to the four main experiments, a handful of special-purpose experiments make use of the interaction points as well. LHC-f, for instance, while not technically part of ATLAS, consists of two detectors that sit ± 140 m from Point 1 along the beamline, and is designed to measure extremely far forward neutral particles that might be produced in ATLAS collisions [59].

3.2.1 Inner Detector

The Inner Detector’s primary purpose is to measure the trajectories of charged particles [60]. When charged particles pass through a magnetic field, the Lorentz force $\vec{F} = \vec{v} \times \vec{B}$ causes them to move in a curve. The curvature of this trajectory can be used to determine a particle’s momentum and charge; additionally, by tracing the track back to the interaction point, it can be associated to a specific proton-proton vertex from the bunch crossing. Therefore, the Inner Detector is surrounded by a solenoid, which generates a 2 T magnetic field. The detector itself consists of a series of layers, which occupy a cylindrical volume with radius $R = 1.15$ m that spans from $z = \pm 3.512$ m on either side of the interaction point. In spherical coordinate, this provides tracking coverage out to $\eta < |2.5|$ [1]. When charged particles pass through the layers of the Inner Detector, they interact and cause “hits”. Tracks can then be found by matching hits from each layer together.

An illustration of the Inner Detector is shown in Figure 3.6. As this diagram shows, the Inner Detector is divided into three sub-systems: the Pixel Detector, the Semiconductor Tracker (SCT), and the Transition Radiation Tracker (TRT). These detectors make use of two separate tracking technologies. The Pixel and SCT detectors both use layers of silicon sensors, arranged in varying configurations. When particles pass through a layer of silicon, ionization causes electron-hole pairs in the semiconductor, which generates an electrical signal that can be digitized and captured as a hit [11]. On the other hand, the TRT makes use of straw drift tubes filled with a gaseous mixture of either xenon or argon and surrounded by polypropylene fibers with different indices of refraction. As charged particles pass through these fibers, they emit transition radiation photons; then the particles and transition radiation photons both pass through the tubes and ionize the gas [52]. Electrons produced from this ionization are then collected by the application of a voltage across each straw.

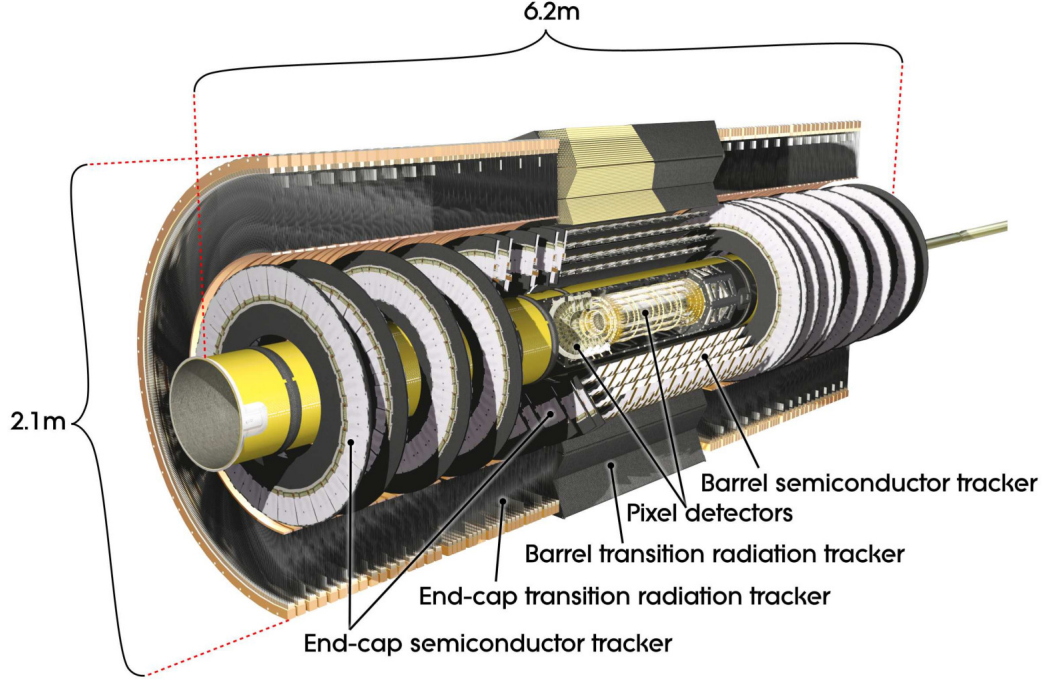


Figure 3.6: General cut-away view of the inner detector, and its three main sub-systems: the Pixel Detector, the Semiconductor Tracker, and the Transition Radiation Tracker [1].

As can be seen in Figure 3.6, the Inner Detector's sub-systems are divided into a central region, known as the *barrel*, and two forward regions, known as the *end-caps*¹⁷. In the barrel, which is closest to the interaction point, detector layers are generally oriented parallel to the beam-line and separated in R . In the end-caps, detector layers are generally orthogonal to the beam-line and separated in z . The detector is divided in this manner in order to maximize the track resolution in each region. Figure 3.7 shows another view of the Inner Detector's layout, in which the separation between the barrel and end-cap is clearly visible. More details about the layout and capabilities of each sub-system of the Inner Detector can be found below.

3.2.1.1 Pixel Detector

The pixel detector [61] is the closest part of the Inner Detector to the interaction point, extending from a radius of $R = 33$ mm to 149.6 mm from the beam-line. In the barrel region, which covers a length of $z = \pm 400.5$ mm, the pixel detector is divided into four concentric cylindrical layers at varying radii. In the end-caps, there are three disks on either side of the barrel between $495 < |z| <$

¹⁷The calorimeters and muon spectrometer are also divided into central/barrel and forward/end-cap layers, as described below.

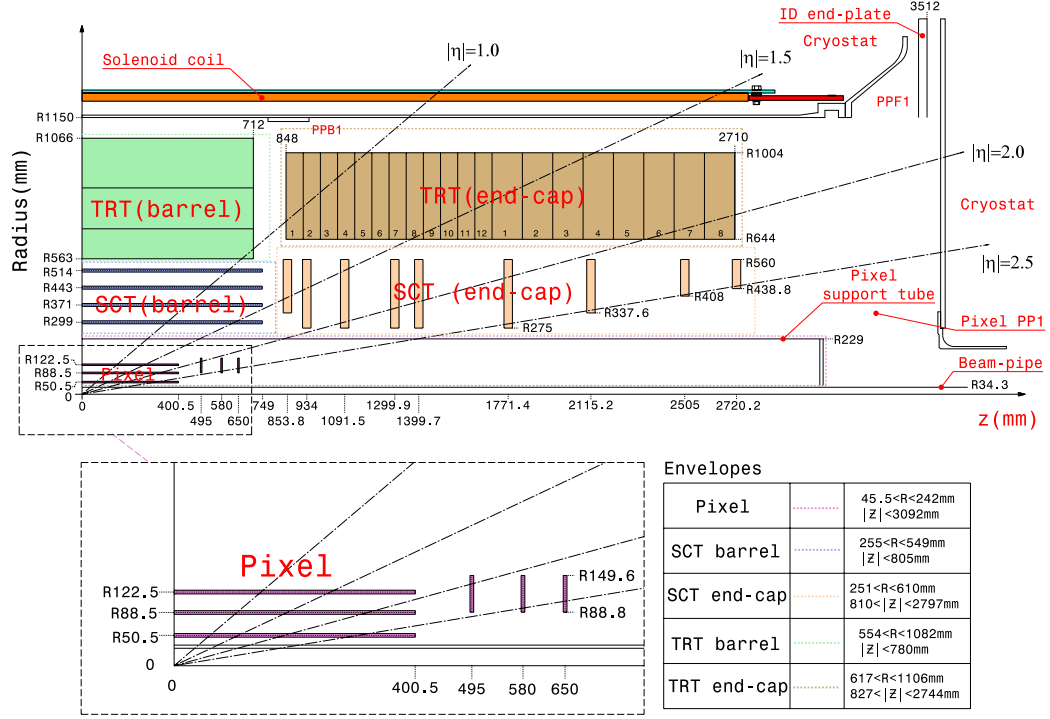


Figure 3.7: Layout plot of one ϕ slice of the Inner Detector, showing the locations of the Pixel, SCT, and TRT layers in cylindrical coordinates [1]. Lines show example tracks for certain values of η ; the Inner Detector provides coverage out to $\eta = 2.4$. The Inner Detector is symmetric about the interaction point $R = z = 0$, so only a single quadrant is shown here.

650 m, which sit at increasing values of z . The innermost layer of pixels in the barrel, called the Insertable B-Layer (IBL), was added to ATLAS as an upgrade between Run 1 and Run 2 [62]. Prior to the IBL's insertion, the lowest pixel layer sat at $R = 50.5$ mm, as shown in Figure 3.7; the exact locations all layers can be seen in this plot. The IBL was installed to provide additional tracking capability out to $|\eta| < 3.03$, and more precise vertex identification.

The pixel layers in both the barrel and end-cap are divided into modules. Modules contain the silicon sensors, as well as readout electronics which digitize the hits produced when a particle passes through the sensor and transmits that information off-detector. Because the pixel detector is so close to the interaction point, it is important to have a high spatial resolution when detecting hits. Therefore, the silicon sensors are divided into small areas known as “pixels”, which gives the detector its name. These pixels are $250\ \mu\text{m}$ thick, and have an area of either 50 by $250\ \mu\text{m}$ (in the IBL) or 50 by $400\ \mu\text{m}$ (in the other layers¹⁸), providing very high granularity in R , ϕ , and z . Across all the

¹⁸About 10% of these pixels are actually 50 by $600\ \mu\text{m}$, due to space constraints, in order to fit the front-end electronics used to read out the detector.

barrel and end-cap layers, there are approximately 92 million of these pixels [1].

3.2.1.2 Semiconductor Tracker

The Semiconductor Tracker [63] sits further from the beam-line than the pixels, occupying a radius of $R = 275$ mm to 560 mm. In the barrel region, the SCT comprises four cylindrical layers at varying radii from $0 < |z| < 749$ mm, while each end-cap contains nine disks between $839 < |z| < 2735$ mm. The locations of each layer and disk are plotted in Figure 3.7. Like the pixel detector, the SCT is divided into silicon sensor modules. However, as the SCT is further away from the interaction point, it must cover a larger surface area. Therefore, instead of dividing each sensor into pixels, in the SCT sensors are divided into thin, rectangular strips. This limits the amount of silicon and number of channels needed, at a slight cost to granularity and resolution in one dimension.

Strips are $285\text{ }\mu\text{m}$ thick, and their exact area varies, as sensors are rectangular in the barrel but may have different geometries in the circular disks in the end-caps. In the barrel, a strip is $80\text{ }\mu\text{m}$ wide and 63.960 mm long. In the end-caps, strips vary from around 56.9 to $90.4\text{ }\mu\text{m}$ wide and 54.4 mm to 65.1 mm long. Strip modules are oriented to provide higher granularity in the ϕ plane than in z (in the barrel) or R (in the end-caps). To improve resolution, the four barrel layers are double-sided, with a 40 mrad difference between the sensors on the top and bottom of each layer. Across the entire SCT system, there are 6.3 million strip channels [1].

3.2.1.3 Transition Radiation Tracker

The Transition Radiation Tracker [64] is the outermost component of the Inner Detector, and as noted above, makes use of a different, non-silicon technology for particle tracking. The TRT occupies a radius of $R = 563$ mm to 1066 mm, and is divided into polyimide straw drift tubes that each have a radius of 2 mm. In the barrel region, which in the TRT spans from $|z| < 720$ mm, the straw tubes run parallel to the beam line across this length. In the end-caps, $848 < |z| < 2710$ mm, the tubes are radially oriented, orthogonal to the beam. The drift tubes themselves are filled with a xenon or argon based gas mixture. A tungsten wire anode runs through each tube, creating a 1530 V voltage difference between the wire and the straw tube, which acts as a cathode. When particles passing through the tubes causes the gas to ionize, this electric potential will cause emitted electrons to “drift” towards one end of the tube, where they can be collected by readout electronics. The TRT contains approximately 351 thousand straws across both the barrel and end-cap region [1].

Transition radiation is produced when particles pass through the polypropylene fibers surround-

ing the straws, and can then be absorbed by the gas mixture. Hits from a charged particle passing through a straw tube produce much smaller signals than the absorption of transition radiation, so the two can be distinguished from each other using separate thresholds. The amount of transition radiation produced from a given particle depends on its Lorentz factor, γ . This means that the TRT can be used to aid in particle identification as well as tracking, and in particular is used in electron identification.

3.2.2 Calorimeters

A calorimeter is a detector designed to measure and absorb the energy of a particle. When particles pass through matter and interact, they radiate a fraction of their energy by emitting additional particles. This process continues, creating more and more particles until all of the energy has been emitted, and is known as a shower. If the particle in question is an electron or photon, it emits energy in the form of Bremsstrahlung photons and electron-positron pairs, and the shower is known as an electromagnetic shower. On the other hand, hadronic showers start when a produced quark or gluon hadronizes, creating additional $q\bar{q}$ pairs that then form mesons or baryons, which then interact with nuclei in matter via the strong force. These hadronic showers are commonly known as jets [11].

Beyond the Inner Detector, ATLAS contains two types of calorimeters for measuring both electromagnetic and hadronic showers. Both calorimeters contain multiple layers of absorbing media, where showers can occur, active media, in which showers can be measured, and electronics to record the amount of energy emitted in each layer. The electromagnetic calorimeter, which is the innermost of the two, uses liquid argon as an active medium, lead as an absorbing medium, and kapton electrodes to measure the size of electromagnetic showers. Hadronic particles will pass through this layer without minimal energy loss, and so the ECal is surrounded by a larger hadronic calorimeter to absorb jets. In the barrel region, the hadronic calorimeter uses steel absorbers, scintillating tiles, and photomultiplier tubes to measure the shower energy. This tile calorimeter is surrounded by end-cap and forward calorimeters, which also use liquid argon as an active material [1].

Figure 3.8 provides an illustration of both the electromagnetic and hadronic parts of the calorimeter system. Combined, the calorimeters provide coverage out to $|\eta| < 4.9$, and occupy a radius from the end of the Inner Detector out to $R = 4.25$ m.

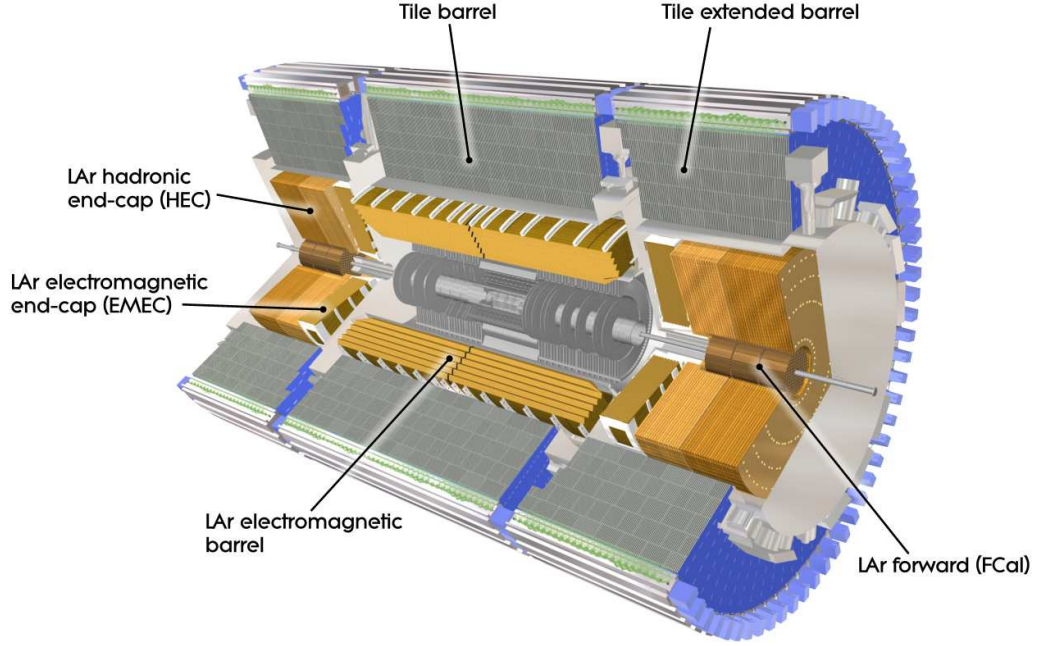


Figure 3.8: General cut-away view of the LAr and Tile calorimeters, in both the central barrel and end-cap regions [1]. The central cavity contains the Inner Detector, shown in Figure 3.6.

3.2.2.1 Electromagnetic Calorimeter

The electromagnetic calorimeter (ECal) [65] contains a barrel region that provides coverage out to $|\eta| < 1.475$ and two end-caps, which cover the region $1.375 < |\eta| < 3.2$. Both barrel and end-caps are broken down into an accordion geometry, which provides symmetry in ϕ across the detector volume without any discontinuities or “cracks”. The calorimeter consists of several layers of liquid argon, with kapton electrodes for readout, and lead absorbers. Figure 3.9 shows an example drawing of part of the ECal barrel region, in which the three separate barrel layers can clearly be seen. Each layer has a different resolution in η and ϕ ; for instance, the first layer provides very fine η granularity while the second consists of square cells of size $\Delta\eta = 0.025$ by $\Delta\phi = 0.0245$. In addition to the layers shown here, there is a presampler layer out to $|\eta| < 1.8$ in both barrel and end-cap which is used to measure the energy lost before particles enter the main calorimeter [1].

As can be seen in Figure 3.9, distance in an electromagnetic calorimeter can be described in terms of radiation lengths. One radiation length X_0 is defined as the average distance over which the energy of an electron is reduced by a factor of $\frac{1}{e}$. The radiation length depends on the atomic radius and number density of the nucleus; for lead, the radiation length $X_0(\text{Pb}) = 0.56 \text{ cm}$ [12]. To ensure most electromagnetic showers are stopped here and do not escape out into the rest of

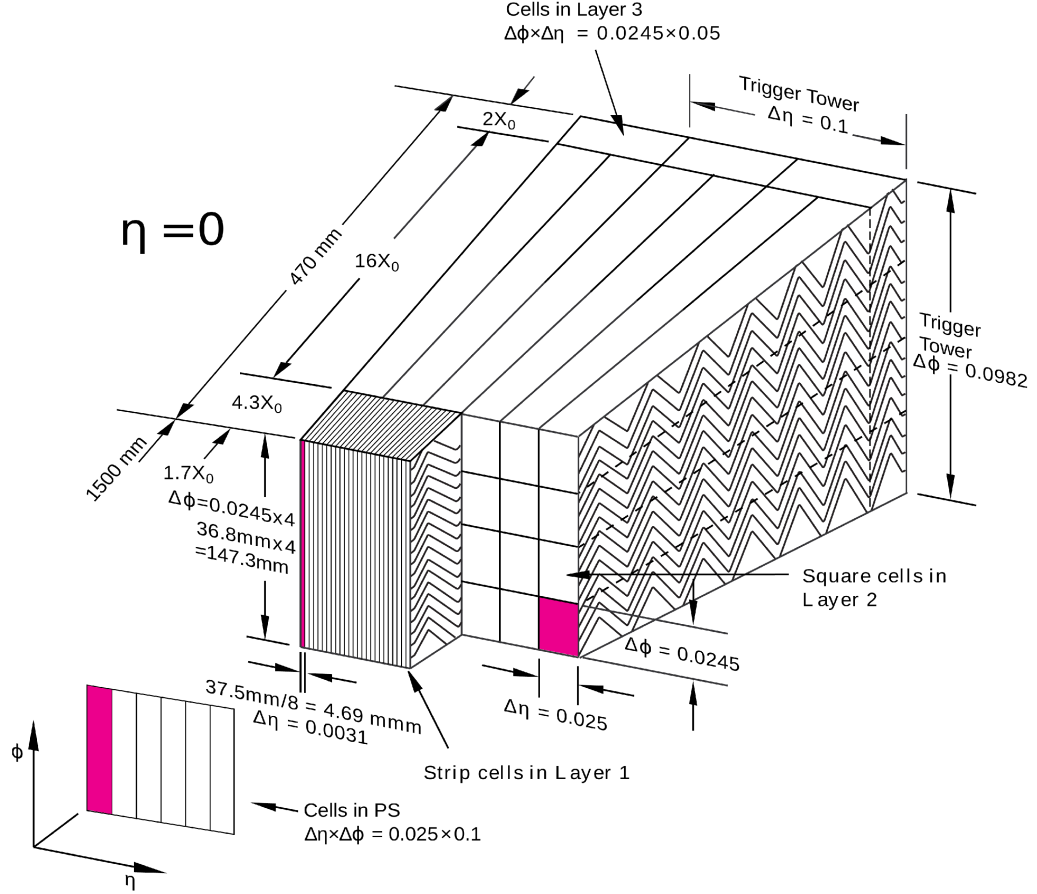


Figure 3.9: Diagram of a LAr barrel module, showing the different layers, the thickness (in radiation lengths), and physical resolution [1].

ATLAS, the total thickness of the ECal is at least $22X_0$ in the barrel and $24X_0$ in the end-caps. The amount of material a shower will pass through varies with $|\eta|$, and may be as high as $33X_0$ near $|\eta| = 1.3$.

3.2.2.2 Hadronic Calorimeter

The Hadronic Calorimeter (HCal) [66] uses both liquid argon and scintillating tiles as active media to capture jets. The tile calorimeter sits in the barrel region, $|\eta| < 1.0$, with extended barrels from $0.8 < |\eta| < 1.7$. The barrels are divided into steel absorbing layers and plastic scintillator tiles. When particles pass through the tiles, scintillating photons are emitted and recorded by photomultiplier tubes. The depth of the tile calorimeter is governed by the nuclear interaction length, λ , the average distance between hadronic interactions at relativistic energies. For iron (and steel), the

interaction length for pions (which comprise a large fraction of hadronic jets) is 20.42 cm [12], which is much larger than the radiation length of the lead used in the electromagnetic calorimeter. The tile calorimeter is therefore divided into three layers which collectively span approximately 9.7λ interaction lengths, or a region from $R = 2.28$ m to $R = 4.25$ m [1].

In the end-caps, a liquid argon hadronic calorimeter is used instead, and provides coverage over $1.5 < |\eta| < 3.2$. Unlike the electromagnetic calorimeter, copper is used as the absorbing material here instead of lead. Copper has a pion interaction length of $\lambda = 18.51$ cm, and as with the tile calorimeter, the hadronic end-cap calorimeter is designed so that hadronic jets pass travel a distance of about 10λ .

3.2.2.3 Forward Calorimeters

Both the electromagnetic and hadronic calorimeters only provide coverage out to around $\eta \approx |3.2|$. To provide coverage in the forward region, the appropriately named forward calorimeters sit on either side of the hadronic end-caps at a distance of about 4.7 m from the interaction point [67]. The FCal contains three layers, and uses liquid argon as an active medium. The first layer uses copper absorbers, is intended to capture far forward electromagnetic showers, and is $26.7X_0$ thick. The two rear layers use tungsten ($\lambda = 11.33$ cm) and are designed for hadronic showers. As with the hadronic calorimeter, these three layers provide a coverage of around 10λ for jets.

3.2.3 Muon Spectrometer

Most charged particles will fully deposit all their energy in either the electromagnetic or hadronic calorimeter, but there are exceptions. Most notably, high energy muons, due to their mass, are relativistic enough that are only minimally deflected in the Inner Detector and only lose about 3 GeV of energy as they pass through the calorimeters. The Muon Spectrometer [68], the outermost component of ATLAS, is therefore intended to measure the momenta of muons (and any other particles) in the region $|\eta| < 2.7$. Like the Inner Detector, the Muon Spectrometer is immersed in a magnetic field to cause muons to curve. A barrel toroid and two end-cap toroids generate fields of approximately 0.5 T and 1 T in each region, respectively [1]. The system then consists of several layers of drift-tube and wire chambers to detect the passage of muons; tracks are reconstructed by combining hit information from the different layers.

Figures 3.10 and 3.11 show a drawing of the Muon Spectrometer and a layout plot in the $r - z$ plane. There are four types of modules used in the detector, which can be divided into two sub-

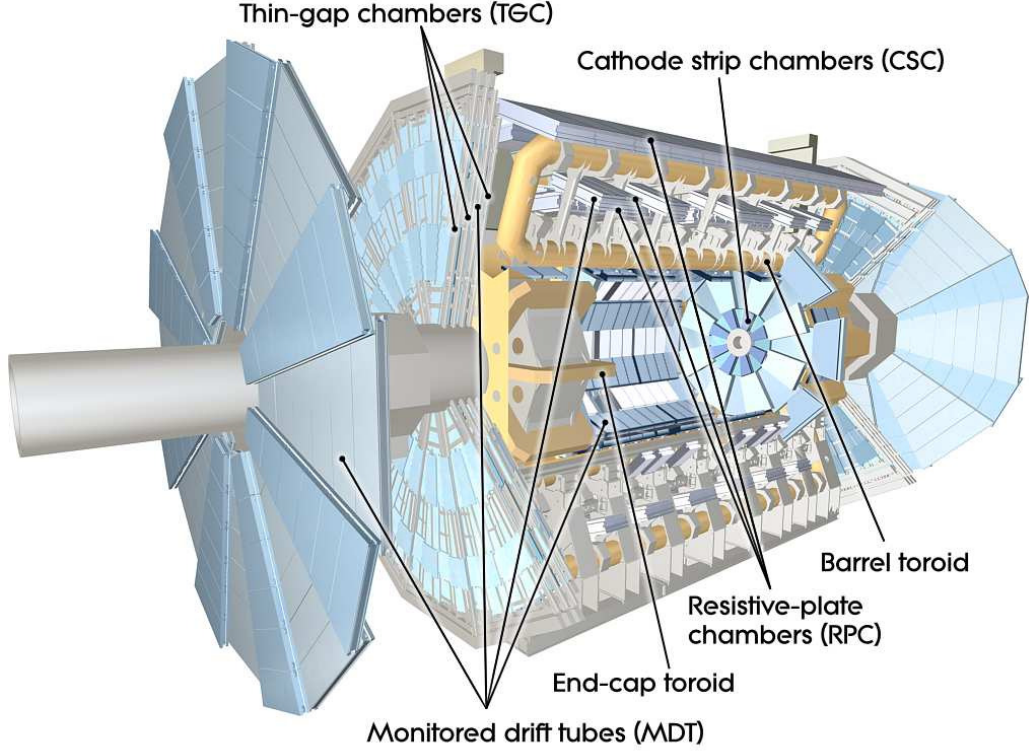


Figure 3.10: General cut-away view of the ATLAS muon system [1]. The central cavity contains the calorimeter system (as shown in Figure 3.8) and inner detector (Figure 3.6).

systems: two types of “precision-tracking” chambers and two types of “trigger” chambers. All four of these chambers use similar technology to the TRT: charged particles cause gas to ionize, and the ionized electrons are then collected and measured to record a hit. The precision tracking system consists of Monitored Drift Tubes (MDT) and Cathode Strip Chamber (CSCs). Each MDT contains a tungsten-rhenium wire and is filled with a mixture of argon and carbon dioxide gas. A potential difference of 3080 V causes electrons to drift when the gas ionizes. These MDTs are organized into three cylindrical barrel layers, and three end-cap wheels, as shown in Figure 3.11. Collectively, these layers provide tracking coverage out to $|\eta| < 2.7$. However, in the innermost end-cap layer, the $2.0 < |\eta| < 2.7$ region will see high rates of activity that are beyond the capabilities of the MDTs: therefore, CSCs are used in this area instead. CSCs are multi-wire proportional chambers: a single CSC module is divided into cathode strips, with multiple wires running across multiple strips. The cathode strips are read out instead of the anode wires, and this provides higher resolution than is available with the MDTs.

The other two types of modules, Resistive Plate Chambers (RPCs) and Thin Gap Chambers

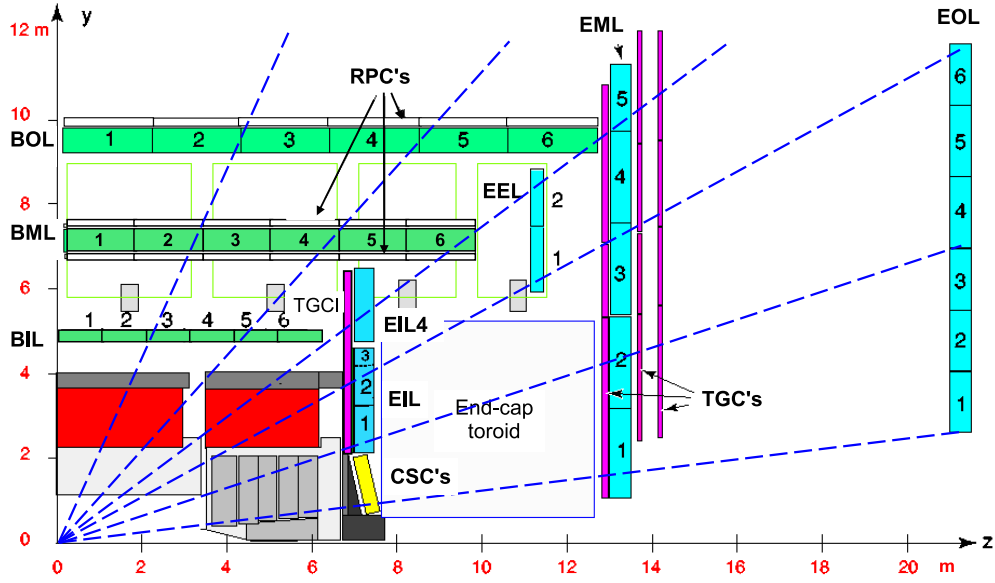


Figure 3.11: Layout plot of one ϕ slice of the muon spectrometer [1]. The three barrel and end-cap MDT layers are shown in green and blue; the CSCs can be seen as part of the innermost end-cap layer. The two sets of trigger modules, the RPCs and TGCs, are shown in white and pink, respectively, surrounding the MDT layers.

(TGCs) form the “trigger” system. RPCs can be found in the barrel, surrounding the MDT layers, and cover the $|\eta| < 1.05$ region, while TGCs are located around the second wheel in each end-cap and cover the $1.05 < |\eta| < 2.4$ region. RPCs use a plastic laminate electrode instead of a wire, while TGCs (like CSCs) are multi-wire proportional chambers. As the name suggests, these trigger modules are used to provide muon information to the ATLAS trigger system, to describe whether or not events should be read out. (More detail about the trigger system can be found in Section 3.3 below). In addition to triggering, data from these modules is also used to complement the information from the precision tracking system when measuring muons.

3.3 Trigger and Data Acquisition

The various components of ATLAS described above produce a large amount of information on every bunch crossing. A single physics event is on the order of 1 MB in size, and as bunch crossings occur every 25 ns, reading out every single event would require ATLAS to have a readout speed of at least 40 TB/s. Even if it were possible to read out the entire detector at that speed, it would only be

possible to store a fraction of this data¹⁹. In order to make this data rate more manageable, it is necessary to quickly determine whether or not a given bunch crossing is of interest before deciding to fully read it out. This process of filtering events is known as *triggering*, and is performed by the ATLAS [Trigger and Data Acquisition \(TDAQ\)](#) system, which is responsible for reading out the detector [70] [71]. A diagram of the components of this system can be seen in Figure 3.12.

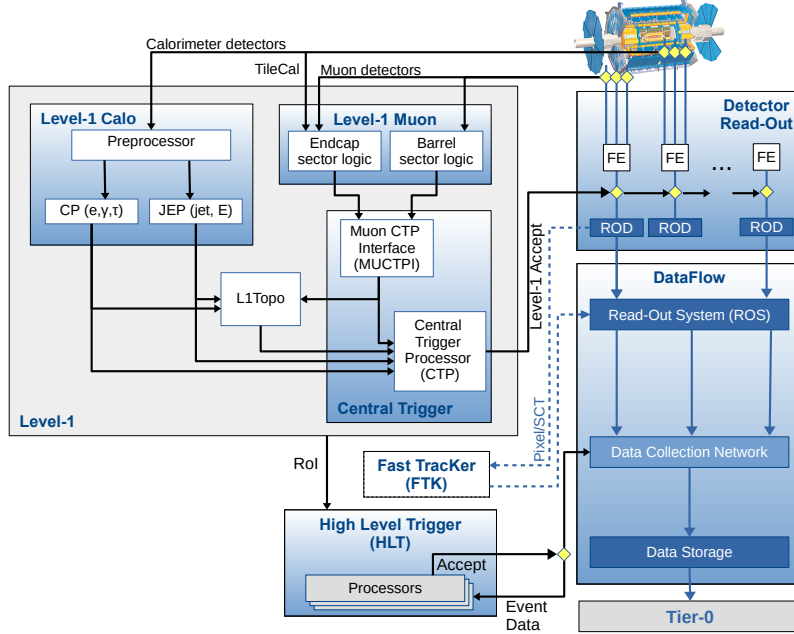


Figure 3.12: Diagram of the ATLAS Run 2 [TDAQ](#) system [70]. The Level 1 trigger receives data from the calorimeters and muon spectrometer, and uses this to make an initial decision. If the event is accepted, it is then fully read out, and the High Level Trigger (HLT) makes the final decision on whether or not to accept it. Note that the Fast Tracker (FTK), a proposed hardware-based track trigger, was ultimately never implemented [72].

In run 2, the ATLAS trigger system consists of a two-step process. First, a fast hardware-based “Level 1” trigger receives minimal information from some of the front-end detectors and makes an initial decision as to whether or not an event should be read out²⁰. This Level 1 process runs at a maximum rate of 100 KHz, meaning that only about 1 in 400 bunch crossings is accepted at this stage. As can be seen in Figure 3.12, the Level 1 trigger consists of “L1Calo” and “L1Muon” components, which receive information from the calorimeters and muon spectrometer, respectively.

¹⁹As of mid 2021, CERN’s tape-based storage systems contain 385.68 PB of data [69]. Even if this were *entirely* dedicated to ATLAS, if we were to read out the detector at 40 TB/s, this would be less than three hours worth of event data.

²⁰In run 1, a three-step decision was used, with a second “Level 2” trigger occurring after the Level 1 [71].

Information from the Inner Detector is not currently used as part of the Level 1 decision. These triggers are implemented in hardware and [field programmable gate array \(FPGA\)](#) firmware, and run simple algorithms with relatively coarse granularity. A Central Trigger Processor takes inputs from the L1Calo and L1Muon systems and makes the Level 1 decision. It also identifies regions of interest in the detector for the event in question that can be examined more closely in the next stage [\[70\]](#).

Once a Level 1 decision has been made, the entire event is read out. Information from all the subsystems is then sent to the second stage, the High-Level Trigger (HLT). Unlike the Level 1 trigger, the HLT is implemented in software, running on a CPU farm of around 40k processing units. Therefore, more sophisticated reconstruction algorithms can be used to identify particles and other objects in each event. A sequence of triggers known as a trigger menu is applied in order to decide which events to accept and to classify events based on the objects they contain. For instance, events which contain at least one electron with transverse momentum greater than $p_T = 24 \text{ GeV}$ will be identified as passing a “single electron” trigger, while other thresholds are used for other objects. The HLT runs at a rate of 1 KHz, or around 1 GB/s, meaning that only about 1 in 100 of the events accepted by the Level 1 decision are written out to disk [\[70\]](#). The events are then run through an even more sophisticated set of offline reconstruction algorithms, which identify specific particles and other objects for use in analyses. A brief summary of these algorithms is presented in the next section.

3.4 Object Reconstruction

Once the data is read out by the trigger system, it then needs to be processed. Raw hit information from the trackers and energy deposits in the calorimeters need to be identified as specific particles so that physics analyses can be performed. The combination of energy and trajectory information allows for the four-momentum of many different types of objects to be calculated, usually measured in the (p_T, η, ϕ, E) coordinate space introduced above in [Section 3.2](#). This section gives a brief overview of the ways in which tracks, electrons, photons, muons and hadronic jets are identified [\[54\]](#), as these are the objects that the analysis presented in this thesis is primarily concerned with. An overview of the way these objects in that analysis can be found later, in [Section 6.3](#). The main particle omitted from this section is the τ lepton, which is much harder to reconstruct than the other two charged leptons because of its short lifetime and the fact that it is heavy enough to decay hadronically [\[73\]](#).

The reconstruction algorithms described here are implemented as part of the ATLAS computing framework, Athena, which is implemented in C++ and Python, and was made open source under the Apache 2.0 license at the end of run 2 in 2018 [74]. Based on the CERN ROOT framework commonly used in particle physics [75], Athena is used to record data and run reconstruction, to generate simulated data (as described in Chapter 7), and to implement common algorithms for analyzing data. More details on the original model and architecture can be found in the computing Technical Design Report [76].

3.4.1 Tracking

Charged particle tracks are identified from hits in the Inner Detector’s subsystems [54]. A single particle might leave hits in adjacent sensors, so individual hits in the pixel, strip, and TRT detectors are combined together via a clustering algorithm. Timing information from the TRT is also included in this pre-processing step. A track will probably contain hits in most if not all of the layers of the Inner Detector, so tracking algorithms are then run across these layers to try and fit the helical track trajectory of a charged particle. Several algorithms are implemented and used, including a Kalman filtering approach [77] and a global χ^2 fitter developed by ATLAS [78]. These fit algorithms determine the five helix parameters which define the trajectory of the track from its perigee, the point closest to the interaction: z_0 , d_0 , η , ϕ , and q/p [79]. The longitudinal and transverse impact parameters, z_0 and d_0 , give the location of the perigee point in the z and $x - y$ planes, respectively. The angular parameters and q/p , the charge/momentum ratio, determine the curvature of the track and the direction it will move.

Tracks candidates are first identified just using information from the pixel and strip detectors, and then extended into the TRT and re-fit with information from the full Inner Detector. After resolving these track candidates, a second pass is performed in which TRT data not yet matched to tracks is used to find unused track segments. These unused segments are then extended back down into the pixel and strip layers if possible [80] [79]. Once tracks have been identified, vertexing algorithms are then run to try and identify tracks which appear to originate from the same point near the beam line [81]. This would mean that the charged particles that left these tracks were likely produced from the same interaction vertex. The identified vertex which has the largest scalar sum of the transverse momentum of its associated tracks is said to be the primary vertex of the event. Other vertices identified will be secondary vertices due to other interactions, such as pile-up.

3.4.2 Electrons and Photons

Both electrons and photons will shower in the electromagnetic calorimeter, and so are identified from energy deposits there [54]. The calorimeter is broken down into 0.25 by 0.25 cells in η - ϕ space in its most granular layers, as shown in Figure 3.9, and deposits in these cells are first clustered together. Historically, up until 2017, this was done using a “sliding window” algorithm, where fixed 3 by 5 cell areas called towers were constructed and the deposits in each cell summed together [82]. This window would “slide” across the detector, and overlapping deposits added together. In 2017, a new approach was introduced using dynamical-size topological or “topo” clusters [83]. Topo-clusters are defined using the cell significance, $\zeta = E/\sigma$, the ratio of energy in a cell to the average noise expected in that cell [84]. Each cell with significance $|\zeta| \geq S = 4$ is taken as a “seed” cell, and combined sequentially with neighbouring cells that have significance $|\zeta| \geq N = 2$ to form clusters. Cells that neighbour this set of merged cells are then also merged into the cluster if they have $|\zeta| \geq N = 2$, and so on, until no more combinations are possible²¹.

Topo-clusters are then matched to tracks from the Inner Detector, by extrapolating the track into the electromagnetic calorimeter. Clusters that match with a track become “supercluster” candidates provided they have energy at least $E_T > 1 \text{ GeV}$ (for electrons) or $E_T > 1.5 \text{ GeV}$ (for photons). Electron superclusters must additionally have matched a track with at least four hits in the strip or pixel detectors. A fixed window of $\Delta\eta \times \Delta\phi = 0.75 \times 0.125$ is defined around the center of the cluster and all lower energy satellite topo-clusters within that window are merged together to create the supercluster. For electrons specifically, a window of 0.25×0.30 is also used. This process runs sequentially starting from the highest energy topo-cluster, and any satellite topo-clusters identified will be skipped. Once superclusters are constructed, a likelihood-based discriminating algorithm is used to decide whether or not they represent an electron or photon. This algorithm is calibrated using data selected with lepton and photon triggers, such as $Z \rightarrow ee$ or $Z \rightarrow ll\gamma$ events. Several thresholds or “working points” are defined with varying identification efficiency for selecting events with electrons and photons [83].

3.4.3 Muons

Muons are reconstructed using a mix of information from the Inner Detector and the Muon Spectrometer [54]. Track segments are constructed in each of the MDT, CSC, RPC, and TGC layers

²¹The topo-cluster algorithm has a lower cutoff threshold, P , which is set to zero here. This is a minimum significance required for cells to be merged. This choice of parameters is known as the “4-2-0” threshold [84].

introduced above in Section 3.2.3 and then combined together with a global χ^2 fitter [85]. An attempt is then made to combine muon tracks with Inner Detector tracks, by performing another fit with both sets of layers included. Depending on whether or not this is successful, four types of muon candidates are identified by the reconstruction algorithms: combined, segment-tagged, calorimeter-tagged, and extrapolated muons. If a combined track can be identified across both systems, this is a combined muon. On the other hand, if the combination fails but an Inner Detector track can still be matched to individual muon segments from one or more module layers, the track is considered a segment-tagged muon. If the track cannot be matched to any muon segments, but instead matches a low-energy deposit in the calorimeter that might have been left from the passage of a muon, then it can be identified as a calorimeter-tagged muon provided it is within $|\eta| < 0.1$ and $15 < p_T < 100 \text{ GeV}$. And lastly, extrapolated muons are identified when a track in the muon spectrometer cannot be matched to an Inner Detector track, but can be extrapolated as having originated from somewhere near the interaction point. These muons have the lowest acceptance of all four types. [85].

Like the electron and photon identification algorithms, muon reconstruction and identification is calibrated using muon-triggered data (generally $Z \rightarrow \mu\mu$ events). Several working points have been defined which indicate the likelihood that a candidate is actually a muon [85].

3.4.4 Jet Finding

Jets refer to the hadronic showers caused by gluons and quarks created during an interaction, which deposit energy into the hadronic calorimeter [54]. Topological clusters, which were defined above for electrons, are first created from the calorimeter energy deposits [86]. A jet finding or jet clustering algorithm is then applied to determine which topo-clusters belong to the same hadronic showers and should be part of the same jets. ATLAS constructs jets using the anti- k_t algorithm, which is a special case of a method known as “ k_t ” [87]. In this family of methods, a modified distance measure d_{ij} between two potential jet constituents is defined as follows:

$$d_{ij} = \min(p_{T,i}^{2n}, p_{T,j}^{2n}) \frac{\Delta R_{ij}^2}{R^2} \quad (3.1)$$

In Equation 3.1, ΔR is the distance between the two objects in $\eta - \phi$ space, and p_T the transverse momentum of each object²², while R and n are external parameters. The algorithm runs over all

²²The algorithm is called “ k_t ”, because in the original paper(s) defining it, k was used instead of p to refer to the momentum of the constituent particles.

objects in the event and selects the pair i, j which give the smallest d_{ij} . This is then compared against $p_{T,i}^{2n}$, which is taken as a measure representing the distance between the object i and the interaction point [87]:

- If $d_{ij} < p_{T,i}^{2n}$, then the objects i and j are combined together by vectorially adding their four-momenta.
- If $d_{ij} > p_{T,i}^{2n}$, then the object i is identified as a complete jet, and removed from the list of objects.

The algorithm continues running until no objects are left to cluster. The variable R acts as a size parameter controlling the sizes of the jets that will be identified; if $\Delta R_{ij} > R$, it becomes harder to merge the objects together. The parameter n was set to 1 in the original k_t method. Unfortunately, k_t does not properly handle large numbers of soft, low- p_T particles. The anti- k_t algorithm was introduced to fix this issue by setting $n = -1$, which has the effect of causing soft particles to be clustered into high- p_T particles first, producing much more conical jets. The impact of this change is shown below in Figure 3.13, where the anti- k_t algorithm produces jets that appear much more natural than the k_t algorithm [87].

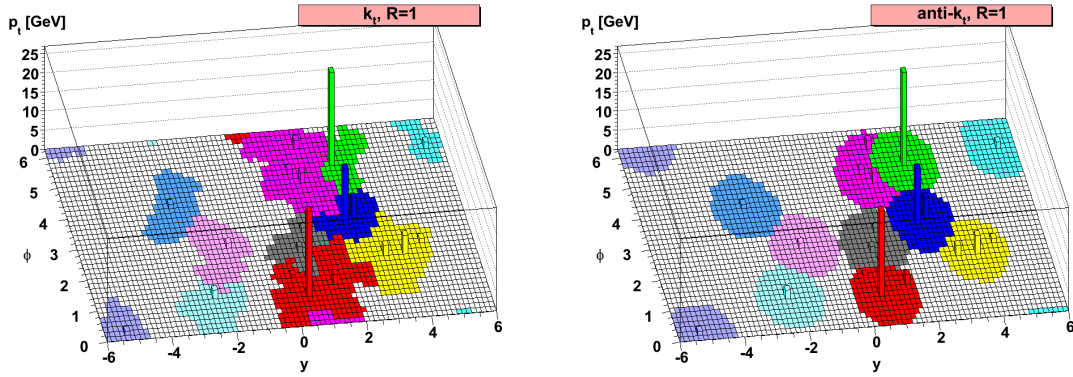


Figure 3.13: Sample event clustered using both the k_t and anti- k_t algorithms, for comparison purposes, from the anti- k_t paper [87]. Using anti- k_t produces noticeably more circular jets compared to k_t .

ATLAS produces jets using a size parameter of $R = 0.4$ by default, but other values are sometimes used for specific applications. Calibration of the jet energy scale is then done using jet-triggered data, such as QCD di-jet and multijet, and Z+jet and γ +jet events. Historically, ATLAS has only formed jets using topo-clusters. However, jet constituents can leave charged tracks in the Inner

Detector, so it would be good to have a way of including them in the jet-finding process. If tracks and topo-clusters are clustered together, some jet constituents will be double-counted, as they will have deposited energy in the calorimeter and created a track in the Inner Detector. During run 1 and run 2, this was dealt with by only using topo-clusters and matching jets to tracks afterward. A new “particle flow” algorithm has been developed and introduced during run 2 which first matches topo-clusters to tracks and then subtracts away any calorimeter deposits which overlap a track. Then this modified set of tracks and clusters is passed to anti- k_t to create particle flow jets, which have become the default for full run 2 analyses [86].

It is quite difficult in general to determine which quark or gluon initiated a jet. The easiest type of jet to identify is the b -jet, as bottom quarks will travel further away from an interaction vertex before beginning to hadronize. A multivariate machine learning algorithm, MV2c10, is used to determine the likelihood that a given jet originated from a b -quark as opposed to another flavour [88].

3.4.5 Missing Transverse Momentum

The objects listed above all interact in one way or another with the ATLAS detector, which is how they can be reconstructed. Neutrinos, however, are stable and electrically neutral, so they only interact through the weak force. They will pass through the detector without interacting with either the tracker or calorimeter, and so they cannot be detected directly. Instead, their presence is inferred through a variable known as the missing transverse momentum [54]. Momentum is conserved, so in the transverse plane, the vector sum of the momenta of all particles produced in a given event should be zero. If this vector sum is calculated from all reconstructed particles and is *not* zero, it indicates that a particle was produced which escaped the detector and was not identified. The missing transverse momentum p_T^{miss} is therefore defined as the negative vector sum of all jets, leptons, photons, and tracks not associated with any other object [89]. Often the magnitude of the missing transverse momentum, which is labelled E_T^{miss} , is used instead, and sometimes referred to as the “MET”.

3.4.6 Jet Vertex Tagging

When trying to classify an event, it is important to be able to distinguish jets created from the primary vertex from pile-up jets created by another vertex. Two methods developed do this are

known as [Jet Vertex Tagging \(JVT\)](#) [90] and [Forward Jet Vertex Tagging \(FJVT\)](#) [91]. A brief overview of both is presented here.

Central jets with $|\eta| < 2.4$ should have associated tracks from the Inner Detector, meaning that it should be possible to figure out if these tracks are associated with the primary vertex or not. Two variables have been introduced in order to try and quantify this, a corrected jet vertex fraction, and R_{pT} . The jet vertex fraction is defined as the ratio of the scalar sum of the p_T of all tracks in a jet that are associated with the primary vertex to the sum of the p_T of all tracks in the jet, whether they are associated with the pile-up. The corrected form, shown below in Equation 3.2, normalizes the denominator to the amount of pileup, to ensure that the JVF remains a useful variable as the number of collisions per bunch crossing increases [90].

$$\text{corrJVF} = \frac{\sum p_T^{\text{track}}(PV_0)}{\sum p_T^{\text{track}}(PV_0) + \frac{1}{n_{PU}} \sum p_T^{\text{track}}(PV_i \neq PV_0)} \quad (3.2)$$

The variable R_{pT} is similar; it is defined as the ratio of the sum of the p_T all tracks in the jet associated with the primary vertex to the p_T of the entire jet. The combination of these two variables is used to construct a multivariate discriminant known as the jet vertex tagger. A JVT score is assigned to a jet indicating whether or not it is consistent with the primary vertex or more likely to have originated from pileup [90].

The JVT method is quite effective, but will not work for forward jets located in $|\eta| > 2.4$. An alternate method known as forward jet vertex tagging has been developed for these jets instead [91] [92]. In forward jet vertex tagging, JVT is first used to remove any central pileup jets. Then, a missing transverse momentum $p_T^{\text{miss},i}$ is calculated for each pileup vertex i as the negative vector sum of all jets and tracks associated with that vertex. For a given forward jet and pileup vertex, a FJVT score is defined by projecting the missing transverse momentum of that vertex onto the momentum of the jet, as shown below in Equation 3.3. A large FJVT score indicates that the missing transverse momentum of this vertex is consistent with this forward jet, making it likely that the jet is pileup [91].

$$\text{FJVT}_i = \frac{p_T^{\text{miss},i} \cdot p_T^{\text{jet}}}{|p_T^{\text{jet}}|^2} \quad (3.3)$$

The FJVT_i is calculated for all pairs of pileup vertices and jets, and the maximum for a given jet taken as the discriminating variable [91]. Using this method, pileup identification can be performed in the forward region as well.

CHAPTER 4

ATLAS High Luminosity Tracker Upgrade

This thesis describes an analysis done primarily using data from the second run (or “run 2”) of the ATLAS detector at the Large Hadron Collider, a period of time ranging from 2015 to 2018. In that time, protons were collided at a center-of-mass collision energy of 13 TeV and 139 fb^{-1} of data was collected [56]. The previous chapter describes the ATLAS detector and its myriad subsystems as they existed during this period; the remainder of the thesis is dedicated to the analysis of that data. However, before beginning to discuss that analysis, it is worth taking a moment to discuss the future of the ATLAS detector. ATLAS, and the LHC, have already been through a major upgrade cycle between 2012 and 2014 in which the center-of-mass collision energy was raised from 8 to 13 TeV. The LHC program is scheduled to continue taking data until at least 2036, so it should not be too surprising that more upgrades are planned between now and then that will eventually render Chapter 3 outdated.

As of this writing, work is underway in preparation for both the LHC’s run 3, scheduled to begin by mid-2022, and the major [High Luminosity Large Hadron Collider \(HL-LHC\)](#) upgrade program, which is scheduled to begin once run 3 finishes (currently scheduled for late 2024). Unlike the upgrade from run 1 to run 2, which increased the *energy* of collisions, the HL-LHC will instead increase the collision *rate*, with the goal of colliding significantly more protons and thus substantially increasing both the instantaneous and integrated luminosity. By the end of run 3, ATLAS is projected to have gathered 300 fb^{-1} of data, but this should increase by an order of magnitude to 3000 fb^{-1} after a decade of HL-LHC operation [93].

As part of this program, a large number of upgrades to ATLAS’s subsystems and data collection are currently being prepared for installation in 2025 and 2026. One of those projects is the [ATLAS Inner Tracker \(ITK\)](#) upgrade, which will see the construction of two new strip and pixel tracking

detectors. This chapter, and Chapter 3, are devoted to work done on the design and verification of the front-end readout electronics for the new ITk strip detector. Along with Appendices A and B, this section of the thesis should be somewhat self-contained, though a few references are made back to Chapter 3. Readers not interested in the tracker upgrade can skip directly to Chapter 6.

This chapter is organized as follows: first, an introduction to the entire ATLAS HL-LHC upgrade program and an overview of the ITk strips detector are given in Sections 4.1 and 4.2. Functional descriptions of the ABCStar and HCCStar chips that comprise the strip detector’s readout electronics are given in Sections 4.3 and 4.4. Some information on the operation of these chips is given in Section 4.5, and finally, a description of the development process and timeline for the HCCStar is presented in Section 4.6.

4.1 ATLAS and the High Luminosity LHC

Many of the physics processes studied at or searched for at the LHC are potentially quite rare. Therefore, increasing the amount of data collected by the experiments, including ATLAS, will reduce statistical uncertainties on measurements and increase the chances of detecting a rare decay or interaction. By increasing the rate at which collisions occur, it becomes possible to collect more data in the same amount of time. For example, to reach an integrated luminosity of 3000 fb^{-1} without any upgrades, ATLAS would need to run for somewhere on the order of 50-60 years: the goal of the High Luminosity LHC is to achieve that goal in significantly less time.

Every 25 ns, a bunch crossing occurs inside the ATLAS detector, in which a certain number of protons from each bunch interact. The concept of pile-up, μ , was introduced in Section 3.1 as the number of proton-proton interactions that occur for every bunch crossing. ATLAS was originally designed to handle an average $\mu = 20$, and over the course of run 2 has been pushed to operate with pile-up rates as high as $\mu = 60$ [56]. In the High Luminosity LHC, the number of proton-proton interactions will be increased to at least an average of $\mu = 200$, a factor of ten times the original ATLAS design goal and a factor of 3.5 times higher than where ATLAS currently operates [93]. This will lead to significant increases in the amount of activity recorded during each bunch crossing in all areas of the detector.

The existing ATLAS detector electronics and readout systems, described in Chapter 3, will not be able to operate in this high luminosity environment and will need to be upgraded. This section outlines some of the key changes being made to both the detector itself and to the trigger system in preparation for HL-LHC running, which is expected to begin in around 2027 or 2028.

4.1.1 Inner Tracker

The ATLAS Inner Detector, described in Section 3.2.1, will be entirely replaced with a new Inner Tracker (ITk). Unlike the current Inner Detector, this will be an all-silicon detector: there will be no new version of the straw drift-tube based Transition Radiation Tracker. However, like the current Inner Detector, ITk will still be divided into pixel and strip subsystems, with the smaller pixel modules closer to the beam and the larger silicon strips further away [94] [95]. The new inner tracker will be able to provide extended tracking coverage in the forward region out to $|\eta| < 4.0$, well beyond the current limit of $|\eta| < 2.4$.

This complete replacement of the inner detector is necessary for several reasons. First, the current inner detector was designed to handle a collision rate of 23 proton-proton interactions per bunch crossing. The integrated luminosity has already been pushed higher than this in run 2, but still nowhere near the average rate of 200 interactions per bunch crossing at the HL-LHC. The tracker will need to read out events at an expected rate of at least 1 MHz, and the Inner Detector would not be able to keep up due to bandwidth and other performance constraints. In addition, there was interest in exploring the use of tracking information as part of the initial trigger decision, something not currently done in the current detector as part of the hardware trigger decision, something that is not currently done and would need support for an even higher readout rate. A new front-end readout infrastructure is necessary to achieve these goals.

Second, given that the tracker is the closest part of ATLAS to the actual proton-proton collisions, it is exposed to the most ionizing radiation from the decay of particles produced in those collisions. The increase in collision rate means a corresponding increase in the total radiation exposure. By the end of the HL-LHC's run, it is projected that the innermost pixel detector will be exposed to 1.7 gigarads, and the strips detector will receive 50 megarads. The current tracker—especially the pixel detector—is not designed to withstand this level of additional radiation exposure, and so will need replacement [94].

Both the strips and pixel detectors will be organized into a series of silicon sensor modules with varying geometries. When particles pass through these silicon sensors, they will generate an electrical signal. Both detectors will use (different) custom electronics to digitize, cluster, and read out that hit data in response to commands from the upgraded ATLAS trigger system. The remainder of this chapter focuses specifically on the ITk strips detector, but for the interested reader, additional detail on both the pixel and strips detector can be found in the technical design reports from both projects.

4.1.2 Other Detectors

The calorimeters and muon spectrometer, described in Sections 3.2.2 and 3.2.3, will not be completely replaced but will also be upgraded. In the calorimeters, new front-end and back-end readout electronics will be installed that are capable of handling both higher readout rates and more radiation due to the increase in luminosity [96] [97]. Readout electronics will also be upgraded in the muon spectrometer; in addition, some MDT, RPC, and TGT chambers will be replaced (and additional chambers added) in certain areas of the detector [98]. These upgrades follow on from the New Small Wheel project, which is currently ongoing in preparation for run 3, which will completely replace both innermost end-cap wheels (which currently contain the CSCs) of the muon spectrometer [98].

In addition to these upgrades, an entirely new subsystem, the High Granularity Timing Detector (HGTD) is in the process of being built. The HGTD will be installed in the gap between the barrel and end-cap calorimeters, providing timing measurements in the $2.4 < |\eta| < 4.0$ region. This timing information will be used to help with tracking and pileup identification in the forward region [99]. More details about all of these upgrade projects can be found in the referenced Technical Design Reports.

4.1.3 Trigger and Data Acquisition

Increasing the number of interactions per bunch crossing means that ATLAS will generate considerably more data for each event. Therefore, it will be necessary for the TDAQ system described in Section 3.3 to trigger on and read out events at higher rates than it is presently capable of supporting. To support the increase in luminosity, the HL-LHC ATLAS trigger will need to operate about 10 times faster; the current system has bandwidth and latency limitations and would not be able to achieve this. Additionally, increased pileup conditions means that existing algorithms will suffer from reduced performance. Therefore, an upgraded TDAQ system is currently being developed [100].

Like the current detector, the upgraded TDAQ system will consist of a two-level trigger. An initial hardware-based “Level 0” trigger will replace the current Level 1 system, and run at a rate of 1 MHz instead of 100 KHz. Like the current L1 trigger, the L0 system will receive data from the calorimeters and muon spectrometer and use this to decide which events to read out. Among other upgrades, the Level 0 system will contain a new Global Trigger component, which will receive full-granularity data from the calorimeters and allow for more sophisticated algorithms to be implemented directly in hardware. Then, the second stage consists of the Event Filter (EF), which will

process events at 10 KHz instead of the current HLT’s 1 KHz [100].

The possibility of using data from the Inner Tracker as input to the hardware trigger decision was explored as an “evolved” upgrade scenario. In the evolved scenario, the Level 0 trigger rate would be increased from 1 MHz to 4 MHz. For events accepted by the L0 decision, up to 10% of the Inner Tracker would be read out at 600 KHz. This regional tracking information would be sent to a hardware-based tracker, [Hardware Tracking for the Trigger \(HTT\)](#), where it would be used as input to a second “Level 1” trigger decision, which would operate at a rate of 400 KHz [100]. The front-end electronics for the ITk Strip detector, described below, were designed with support for this multi-level trigger decision. However, due to technical issues, the decision was taken to not proceed with the evolved scenario for the HL-LHC. More information about the impact this had on the ITk design can be found below; more details about the HTT proposal can be found in [Appendix B](#).

4.2 Inner Tracker Strips Detector

The last section presented a general overview of the ATLAS HL-LHC upgrade program. The remainder of this chapter focuses specifically on the new Inner Tracker’s strips detector, its overall layout, and its readout architecture.

4.2.1 Strips Layout

The strips detector will occupy an annular region inside ATLAS, from an inner radius of $R = 0.4$ m to an outer radius of $R = 1$ m. As is the case with the current Inner Detector, that space will be divided into three main regions: a central “barrel” region spanning $z = -1.4$ m to $z = +1.4$ m, and two forward “end-cap” regions which stretch from $z = \pm 1.4$ m to $z = \pm 3$ m, sitting on either side [94].

Both the barrel and end-caps will be divided into a series of layers of silicon sensors. In the barrel, there will be four cylindrical layers separated in R , while in the end-caps, there will be six disks separated in z . [Figure 4.1](#) shows a simulation of the strip detector’s geometry, with the different layers clearly visible [94].

In the barrel, these layers are further divided into staves, on which the silicon sensors are mounted. Each stave is 1.4 m long, and each barrel layer consists of pairs of staves arranged in a ring and running parallel to the beam line. Because the area enclosed by the outer barrel layers is larger than that of the inner layers, the number of staves per layer increases from layer to layer. There are 392 staves total in the barrel, across all layers. Further, to increase the granularity in the innermost

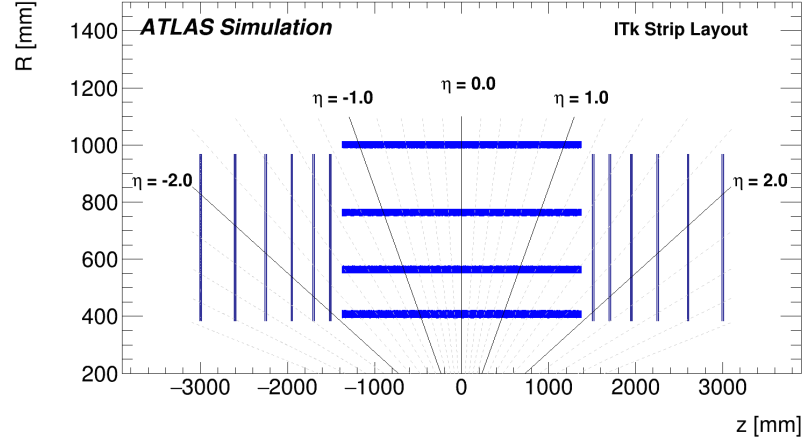


Figure 4.1: Plot from the ITK Strip technical design report showing a two dimensional slice of the ITk strip detector layout in R and z [94]. The strip detector forms an annular cylinder enclosing the beam line at $R = 0$ (as well as the ITk pixel detector, which occupies the space $0 < R < 0.4$ m). Each blue line represents a layer of silicon strip sensors; in the central barrel region, these rows of sensors run parallel to the beam line, while in the forward end-cap regions, they form orthogonal disks. Lines showing a few η values are superimposed on the plot; the strips detector will provide coverage out to $\eta = \pm 2.7$.

layers when constructing tracks, the sensors mounted on the staves in the inner layers each cover a smaller area than the sensors in the outer layers. As a result, the two inner layers are sometimes known as “short strips”, and the two outer layers “long strips” [94].

In the end-caps, disks are broken down into different wedge-shaped structures, known as petals. A disk contains 32 identical petals, and unlike the staves, the layout of the petals does not change from disk to disk. Instead, the sensor geometry of the petal itself varies as a function of the radius [94].

Both staves and petals are broken down into units known as *modules*, which contain the silicon strip sensors as well as the front-end electronics responsible for reading out hits from charged particles passing through them. There will be a variety of different module configurations across the entire system: long-strip and short-strip modules in the barrel and six different configurations in the end-cap. Figure 4.2 shows an example layout of a long-strip staff and a petal, with the different module configurations clearly visible. Note that, while it is not shown in the illustration, both staves and petals are double-sided, and have sensors mounted on both sides of the support structure [94].

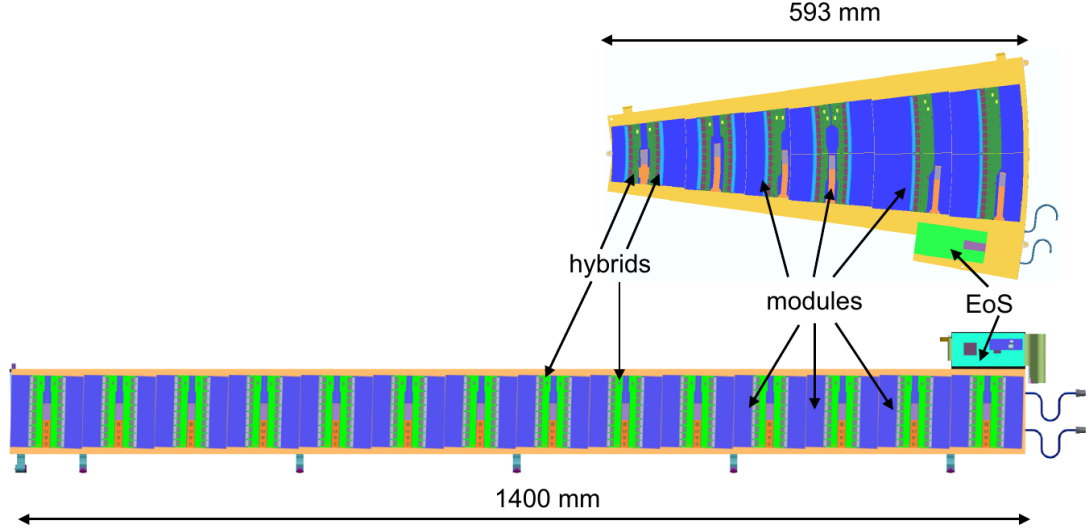


Figure 4.2: Renderings from the ITk Strip technical design report showing an example stave (bottom) and petal (top) for the ITk strip detector [94]. Both the stave and petal are divided into silicon strip sensor modules: on the petal, the sensor configurations vary with the petal’s radius, while on the stave, each module is identical. Each module consists of the silicon sensor, drawn in blue, as well as front-end electronics, drawn in green and referred to as the “hybrid”. Depending on the size and location of the module, there can be multiple hybrids: the example stave drawn here is a “short strip” stave, which has two hybrids per module. Staves and petals also contain an end of substructure (EoS) card, shown in the rendering, which connects the front-end, on-detector electronics to the off-detector systems.

4.2.2 Modules and Hybrids

As shown in Figure 4.2, modules contain a printed circuit board known as the *hybrid*, on which the front-end readout electronics are mounted. In addition to the hybrid, modules also contain a number of other components: the silicon sensors themselves, as well as a DC-DC converter [101] and power board that contain dedicated electronics for powering and monitoring the system. Among these electronics are three custom ASIC, chips designed for a specific purpose that cannot be reprogrammed: the HCCStar, ATLAS Binary Chip (ABCStar), and Autonomous Monitor and Control Chip (AMAC). These components are shown in more detail in Figure 4.3, a cutaway diagram of an example module in the barrel [94].

As the name suggests, the power board provides power to each module. The ITk strips detector will consume more power than the current strip tracker, and so a more efficient way of powering the overall system is needed. To that end, a DC-DC converter will be installed on each module to step

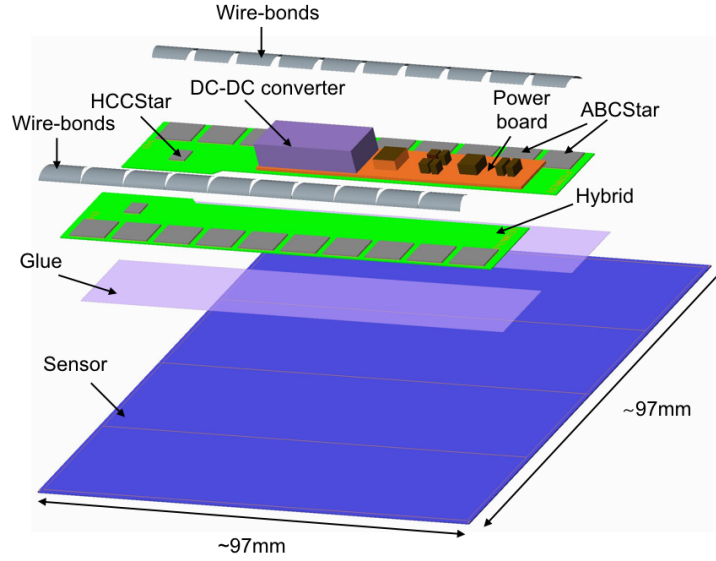


Figure 4.3: Cutaway diagram of an example barrel module from the ITK Strip technical design report [94]. Key components, such as the power board, DC-DC converter, silicon sensors, hybrid, HCCStar, and ABCStars are labeled. In the barrel, each hybrid contains ten ABCStars and one HCCStar. “Short strip” modules contain two hybrids, while “long strip” modules contain one, so this is a “short strip” module in one of the outer two barrel layers. The example stave shown in Figure 4.2 consists of 14 of these modules, sitting side by side.

the external voltage down to a lower voltage for the front-end electronics, thus reducing the overall power consumption and current draw of the system. The power board will also contain one of the three custom ASICs, the AMAC (not shown in Figure 4.3), which as the name implies will provide both monitoring and control functionality. The AMAC will be used to start up the module, as well as reset the other front-end electronics as necessary. It will also measure quantities like the power consumption and temperature of the module while the system is running [94].

The silicon sensors form the active area through which the passage of charged particles can be recorded. Each sensor is broken down into rows of long, thin silicon strips that are around $300\text{ }\mu\text{m}$ thick and less than $100\text{ }\mu\text{m}$ wide. In the barrel, these strips are arranged into $96.64\text{ by }96.64\text{ mm}$ squares, as shown in Figure 4.3. A barrel sensor consists of either four rows of 24.1 mm strips or two rows of 48.2 mm : hence the “short strip” and “long strip” terminology introduced previously. In the end cap, the strips form trapezoidal wedges, as seen in Figure 4.2, and are also broken down into either two or four rows. Each individual strip provides a single channel for the recording of hits [94].

Module Type	Strips/Module	Hybrids/Module	HCCs/ Hybrid	ABCs/Hybrid
Short Strips (Barrel)	2564	2	1	10
Long Strips (Barrel)	5128	1	1	10
Ring 0 (End-Cap)	4360	2	1	8 or 9
Ring 1 (End-Cap)	5640	2	1	10 or 11
Ring 2 (End-Cap)	3076	1	2	12
Ring 3 (End-Cap)	3592	1 (2/2)	2	14
Ring 4 (End-Cap)	2052	0.5 (1/2)	2	16
Ring 5 (End-Cap)	2308	0.5 (1/2)	2	18

Table 4.1: Table summarizing the different ITK strip module configurations, and the number of strip sensor channels and ASICs on each [94]. Strip sensors are organized into rows, with a single hybrid responsible for two rows. In the barrel, each hybrid has one HCC and ten ABCs; the only difference is that the long strips are organized into only two rows, while the shorter short strips modules have four. In the end-cap, configurations are more complex. The curvature of the innermost ring 0 and ring 1 modules is large enough that the number of ABCStars (and, therefore, the number of sensor channels) differs between hybrids on the same module. The outer four end-cap modules are wide enough to have *two* HCCStars on each hybrid, with each HCC connected to half of the ABCs. In fact, in Rings 3, 4, and 5, the sensor area is sufficiently wide that each hybrid is considered to cover two *separate* modules, with one HCC responsible for one module but both HCCs sitting on the same circuit board.

The actual readout of hit data is then done by the electronics on the hybrid. Each hybrid contains two additional custom ASICs, the HCCStar and ABCStar. The exact numbers of HCCs and ABCs vary with the module geometry, but in general, a single HCCStar controls a group of five to eleven ABCStars in a “star” network (hence the name), where the HCC sends and receives messages from the ABCs in parallel. Each ABCStar is connected to two rows of up to 128 strip channels, and is responsible for digitizing and clustering hits from them. Then, when the trigger system decides to read out an event, the readout command is first dispatched to the HCCStar, which then forwards it to the ABCs, which transmit any observed clusters to the HCC. The HCC receives these clusters, synchronizes and merges them together into a single packet for each event, and transmits that packet off-detector through the end-of-substructure card [94].

As has been mentioned, the exact module configuration varies considerably between the barrel and end-cap, and also within the end-cap as a function of the radius R . However, each module contains the fundamental objects shown in Figure 4.3. Full summaries of the different module types, how many hybrids, ASICs, and sensors they contain, and how many total modules can be found in the different layers of the detector are provided in Tables 4.1 and 4.2 [94].

The remainder of this chapter is primarily concerned with the front-end readout electronics, particularly the HCCStar. The next two sections discuss the ABCStar and HCCStar, as well as the flow of data through the system in more detail.

Region	Position (mm)	Staves/Petals	Modules	Module Types
Barrel Layer 0	$R = 405$	28	784	Short Strip
Barrel Layer 1	$R = 562$	40	1120	Short Strip
Barrel Layer 2	$R = 762$	56	1568	Long Strip
Barrel Layer 3	$R = 1000$	72	2016	Long Strip
Disk 0	$z = 1512$	32	576	Ring 0-5
Disk 1	$z = 1702$	32	576	Ring 0-5
Disk 2	$z = 1952$	32	576	Ring 0-5
Disk 3	$z = 2252$	32	576	Ring 0-5
Disk 4	$z = 2602$	32	576	Ring 0-5
Disk 5	$z = 3000$	32	576	Ring 0-5

Table 4.2: Table summarizing the different layers and disks of the ITK strips detector [94]. For the barrel, numbers quoted here are for the entire barrel on both sides of $z = 0$, as shown in Figure 4.1, and for $0 < \phi < 2\pi$. Similarly, for the disk layers, numbers are only quoted for a single end-cap. Note that due to the complex hybrid/module layout described in Table 4.1, each side of a petal has nine modules: one each of R0, R1, and R2, and two each of R3, R4, and R5.

4.3 ATLAS Binary Chip

To discuss how the strips detector will process and read out hit data, it is necessary to discuss the two hybrid ASICs, the ABCStar and the HCCStar. The ABCStar is responsible for digitizing and clustering hits from the passage of charged particles through the sensors [102] [94]. This section describes how this process works and how data flows through the ABCStar. While this part of the thesis is primarily focused on the HCCStar, the two chips are tightly coupled. It is not possible to fully understand one without understanding at least a little about the other.

In fact, some aspects of the two chips are similar, if not identical. Both use the same serial communications protocol, and the implementation of that protocol is mostly shared between the two chips. Further, because the HCC must be able to receive packets generated by the ABC, it needs to be able to interpret and decode the ABCStar output packet format as well. Finally, the chips use a similar register model for configuration and monitoring, and registers can be programmed and read using very similar commands. A description of how to communicate with the ABC, how the HCC and ABC communicate with each other, and how to program both chips can be found in Sections 4.4.1 and 4.4.2 below.

This section focuses on describing how data flows through the ABCStar itself²³. Figure 4.4 shows a high-level diagram of the ABC’s logic, and how data passes through it. Each ABC can be attached to two rows of 128 silicon strip channels on the sensor, or 256 total. On each bunch crossing, the

²³Readers with access to internal ATLAS documents interested in more technical details on the ABCStar beyond what is included here may be able to consult the ABCStar specification document.

analog front-end digitizes these 256 channels and translates each into a single bit: one if a hit was detected, and zero if not. These hits then pass through a series of pipeline-delayed buffers, until they are clustered and transmitted to the HCCStar.

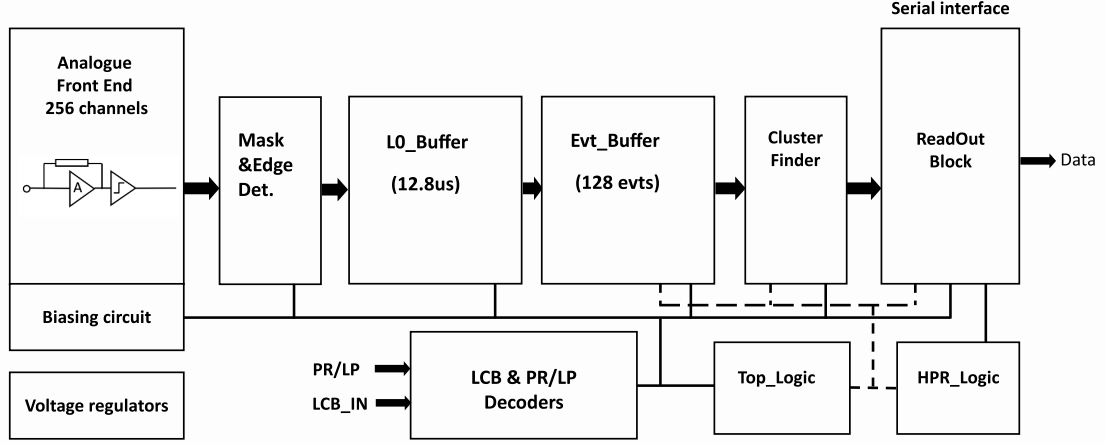


Figure 4.4: Top-level block diagram of the ABCStar, showing how physics data flows through the chip in response to commands from the trigger system [102]. On each bunch crossing, the analog front-end reads in hits from 256 input channels. The bunch crossing is then stored in a pipeline-delayed buffer until a trigger accept command arrives from upstream, at which point the hits are transferred into a smaller event buffer and uniquely identifiable by a 7-bit Level 0 trigger tag. Then, when a readout command arrives to read out the data corresponding to that tag, the hits are clustered and transmitted serially upstream by way of the HCCStar. Details on the PR/LP and LCB communications protocols which are used to transmit these commands can be found in Section 4.4.1.

4.3.1 ABCStar Pipeline

When the ABCStar reads in hits from the analog front-end, it must store them until commands arrive from the trigger system. To that end, as shown in Figure 4.4, hits are loaded into a pipelined buffer, where they can remain for up to $12.8\ \mu\text{s}$. In this buffer, the hits are only uniquely identified by a bunch crossing count, or BCID, which is incremented every 25 ns on every bunch crossing. Every 25 ns, hit data from the current bunch crossing is copied into this pipeline, and the oldest bunch crossing is removed: either copied into the next buffer, if the event has been accepted, or simply deleted if not. Therefore, the depth of the pipeline, which is configurable, determines how long events will be stored, and must be set based on how long it will take the trigger system to

accept an event.

To accommodate the “evolved” hardware track trigger scenario described in Section 4.1.3, the ABCStar supports a two-level trigger decision: first, events can be marked to indicate that the trigger system *might* want to read them out, in what is referred to as the “Level 0” decision. Next, the event would actually be read out, with the ABCs transmitting the hits for that event upstream through the HCC. This enables the potential use of information from the strips tracker in the final trigger decision: the system could mark an event, then sample a subset of the strips modules to help decide if the event is interesting, and only then fully read out hit data from all the modules. More on this can be found in other parts of the thesis.

From the ABC’s point of view, then, it expects to receive two commands: a level 0 accept, and then one (or more) readout requests for the event. The level 0 accept consists of a seven-bit “L0 tag”, which will uniquely identify an event in the front-end, as well as four bits which will define which bunch crossing to tag. It takes 100 ns to transmit this command to the ABCStar, and since a bunch crossing is 25 ns, it is necessary to identify which of the four BC that overlap with the command should be tagged and read out. Multiple bunch crossings can be tagged by setting more than one of the bits to one, in which case the specified tag will be incremented for each successive bunch crossing to tag.

When a L0 accept arrives, the ABC applies the L0 tag to the oldest bunch crossing(s) in the pipelined L0 buffer. The accepted bunch crossings are then copied into the “event buffer”, which can store 128 events at a time. In this buffer, the 7-bit L0 tag is used as the address of an event, so data will remain here until the tag is reused in another L0 accept command. From this point onward, the L0 tag will be the primary way to identify events in both the ABCStar and HCCStar. The 8-bit BCID is stored in the buffer as well, however, and will still be used as an extra piece of metadata to identify the event.

In addition to their primary function, L0 accepts also contain a twelfth bit, which is known as the “bunch counter reset”. As the name suggests, this bunch counter reset will reset the BCID counter on the ABCStar back to zero. This reset can be used to resynchronize the counters across the entire detector, and will be performed automatically in the strips detector every 891 bunch crossings (a period of time known as an *orbit*). The BCR can be sent with or without any of the BC selector bits set: if it is sent with a nonzero BC selector, the reset will be applied after tagging events.

The ABCStar keeps hit data in the event buffer until a second command, known as a readout request, arrives from the HCC. This command contains the seven-bit L0 tag of an event that should be read out. When the ABC receives a readout request for a given tag, it uses that tag as a memory

address in the event buffer and selects the event currently stored at that address. The hit data is then fed into a cluster finder block, which converts the hits into clusters that are then serially transmitted to the HCCStar.

4.3.2 Clustering Algorithm

In normal HL-LHC operating conditions, most ABCStars will have a very low occupancy per event: that is, they will see far fewer than 256 hits. Therefore, transmitting 256 mostly-zero bits to the HCC for every event would waste bandwidth. Instead, the ABC's cluster finder is used to first cluster this 256-bit array of hits down to a list of 12-bit clusters. If there is only one hit, there would only be one cluster, so this has the potential to significantly speed up transmission times.

The ABCStar's cluster finder proceeds as follows. First, the 256-bit hit array is split into two separate 128-bit arrays by separating the even-indexed channels from the odd-indexed channels. Because of the way the sensor channels are wire-bonded to the ABCStar, the odd-indexed channels are all from the first row, and the even-indexed channels from the second row. Thus, these two arrays are processed separately.

Then, for each row of 128 channels, hits are combined into 12-bit clusters using the following algorithm:

1. The cluster finder steps through each 128-bit array until it finds a hit at index x (which can range from 0 to 128).
2. The next three entries in the array ($x + 1$, $x + 2$, and $x + 3$) are checked and combined into a three-bit "next hit" map. If there are less than three entries remaining (because $x > 125$), then zeroes are used as padding.
3. To distinguish clusters from the even array from the odd array, the cluster address is either taken to be x (if this is the even array) or $x + 128$ (if this is the odd array).
4. The algorithm continues to step through the array, skipping the next three entries and proceeding to $x + 4$, until the entire array has been fully processed.
5. A single bit called the "last cluster bit" is appended to the cluster with the highest address, to indicate that this is the final cluster for a given event. For all clusters, this twelfth bit is set to zero.

Let's consider an example. Suppose only two neighboring channels in the “odd” row, with indices 47 and 49, had hits for a given bunch crossing. The cluster finder would output a cluster with address 175 ($47 + 128$), and with a next-hit map of “100”. Since this is the only cluster in this event, the last cluster bit would be set to 1. The total sequence in binary would then be 0b101011111001.

Because a single cluster contains information about four channels, the maximum number of clusters for any given event is 64. Were an event to have 64 clusters, this algorithm would translate 256 bits of information into 768, which is clearly inefficient. A simple calculation shows that 21 clusters (or 252 bits) is the transition point: more than 21 clusters per ABC will not lead to any reduction in bandwidth²⁴. Fortunately, simulations have shown in normal HL-LHC running conditions, the average number of clusters per each ABCStar will be about 2, which is well below this threshold.

Once the cluster finder has evaluated an event, the clusters are then transmitted to the HCCStar. A detailed description of the transmission format can be found in Section 4.4.2 below.

4.4 Hybrid Controller Chip

The other ASIC which sits on the hybrid is the HCCStar, which controls a group of ABCs [94]. The HCCStar dispatches the trigger accept and readout request commands to its star network of ABCs, and receives back clustered hit data. It then takes that data, combines the responses from each ABC together, and transmits it off detector through the end-of-substructure card. All communication with the ABCs, including resets and configuration, occurs through the HCC. Therefore, a large amount of logic is required on the HCC in order to keep the entire hybrid synchronized.

A high-level block diagram of the HCCStar's digital logic is shown in Figure 4.5. This breaks the chip down into three major components: the control path, which receives commands from upstream and passes them to the ABCs; the input channels, which receive back messages from the ABCs, and the packet builder, which synchronizes and merges the messages from the ABCs together into output packets. These three blocks are discussed in considerable detail in the next few sections below, which outline how commands and data flow through the hybrid²⁵.

As of this writing, three iterations of the Hybrid Controller Chip have been developed. A very early prototype, known as the “HCC130”, did not use the star architecture for communicating with

²⁴This does not take into account any event metadata which is sent along with the clusters.

²⁵As with the ABCStar: for additional technical detail on the HCCStar, readers with access should consult the internal HCCStar specification document.

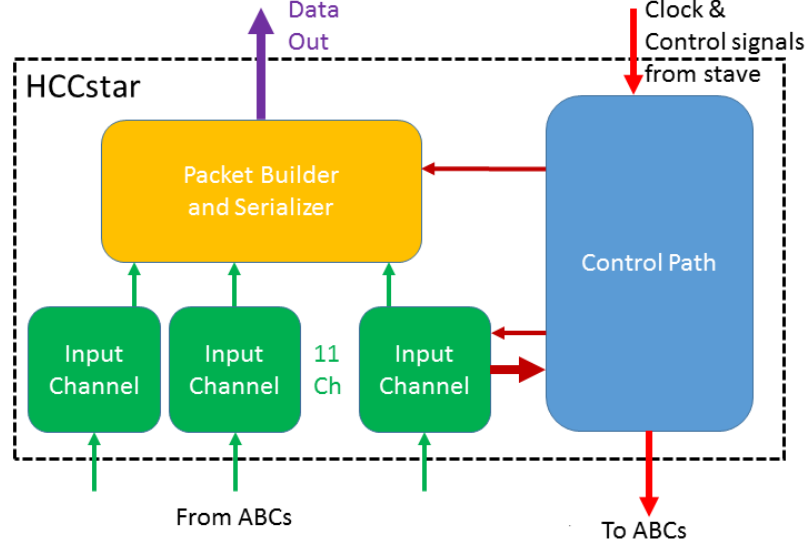


Figure 4.5: Top-level block diagram of the HCCStar. The three main components of the chip are shown here: the **control path** (see Section 4.4.1), which receives commands from the stave and dispatches them to the connected ABCStars; the eleven **input channels** (see Section 4.4.2), which receive packets from each ABCStar in parallel; and the **packet builder** (see Section 4.4.3), which merges the parallel input streams together and produces output packets that are sent back out to the stave.

the attached ABCs²⁶. Experience testing the HCC130 resulted in a redesign that led to the star architecture, and the creation of the HCCStar as discussed in this section. An initial prototype, HCCStar version 0 (“v0”), was developed over the course of 2017 and 2018. Testing of HCCStar v0 identified several issues which would need to be addressed before building the final version, which would actually be installed in the ATLAS detector. A hopefully final Version 1 (“v1”) is currently in the final stages of development. More details on the changes between version 0 and version 1 can be found below in Section 4.6.

The HCCStar was primarily developed by the University of Pennsylvania’s high energy physics instrumentation group, with assistance from other members of the Penn ATLAS group (myself included), the rest of the ATLAS collaboration, and hardware developers at CERN. The work discussed in this section was primarily done by Paul Keener, Nandor Dressnandt, Bill Ashmanskas, and Mitch Newcomer (Penn), with contributions from Matt Warren (University College London), Jaya John John (Oxford), and Pedro Leitao (CERN).

²⁶The strips ASICs use 130 nm transistors, hence the name. “Star” was added to the project names once the star architecture was adopted.

4.4.1 Control Path

When reading out an event, the trigger system will first send commands to the HCCStar’s **Control Path** (the blue block in Figure 4.5). The Control Path contains decoders for the serial protocols used by the HCCStar, as well as a programming interface for configuring the chip and logic for forwarding commands to the ABCStar.

Most communication with the HCC occurs via a custom serial protocol known as **L0A/CMD/BCR (LCB)**, so named because it supports three broad types of messages: **level 0 accept (L0A)** triggers sent to tag and read out an event, commands to program either the HCC or ABCs, and **bunch counter resets (BCRs)** to clear the bunch crossing counter used to assign an ID to events when tagged by a level 0 accept. However, the name LCB is actually a slight misnomer, as BCRs are sent as an optional part of the trigger accept rather than as their own type of message, and “commands” include both “fast” and “slow” commands with different formats: fast commands are shorter than slow commands, and can be sent in a shorter amount of time. Commands can be sent both to the HCC and to the attached ABCs, as the LCB stream is “passed through” to the ABCs from the HCC and all messages can be seen by both chips.

The protocol uses a line encoding known as 6b/8b, where 6 bits of data are encoded as 8 bit words in such a way that each 8 bit word contains an equal number of ones and zeroes [103]. The difference between the number of ones and zeroes in an encoding is called the *disparity*, and so in this encoding each eight-bit word has a disparity of zero. Encoding messages in this way has several advantages. First, by transmitting an even number of ones and zeroes on the serial line, on average the voltage will not be biased low or high, reducing the risk of a glitch or bit error. This state lack of bias is commonly referred to as a “DC balanced” signal. Second, adding two extra bits adds parity to each message: any single bit error in an 8-bit encoded word will produce an invalid sequence. This allows single bit errors, as might be caused by operations in a high radiation environment, to be detected, and makes it harder for a command to be corrupted into a different command. And finally, the extra two bits also allows for the use of “control codes”: valid 8-bit patterns that are also protected against single bit errors, but that do not decode to a 6-bit sequence. These control codes can be used to add additional structure to the protocol, marking the beginning and end of messages.

LCB commands are transmitted as a “frame”, a pair of two eight-bit words. Frames are transmitted at 160 MHz (with a period of 6.25 ns), and a 160 MHz system clock is supplied from the stave along with the LCB serial stream. Since a single frame consists of 16 encoded bits, and a single bit

is sent on each pulse of the clock, it takes 100 ns to transmit a single frame. The LCB line is always active even if no commands are being sent: when there is no other activity, two of the control codes (labelled K0 and K1) are transmitted as an “IDLE” frame. Continually transmitting IDLE frames helps to keep the HCC synchronized: at startup, the LCB decoder will ensure it observes 16 IDLE frames before “locking” to the stream and allowing commands to be parsed, and can unlock again after seeing another 16 decoder errors. LCB lock is also used to generate an internal 40 MHz clock that is in phase with the LCB framing. This clock is passed to the ABCs, as well as used in some parts of the HCC.

As noted above, there are three types of messages that can be sent on the LCB protocol: the L0 accept triggers described in Section 4.3.1, as well as fast commands and slow commands. L0 accepts and fast commands can both be sent as a single frame, while slow commands take multiple frames to transmit, hence their name. To distinguish a fast command from a level 0 accept, fast commands are prefixed with another control code, labelled K3. Slow commands consist of a series of payload frames sandwiched between a “start” and “end” frame. These start and end frames are identified with a fourth control code, labelled K2. Fast commands and L0 accepts can be inserted inside a slow command as well. All slow command payload frames begin with five ‘0’ bits where a L0 accept would have the BC selector and BC reset bits, thus allowing the decoder to distinguish the two.

Command	ID (4 Bits)	Description
Logic Reset	2	Soft reset; resets logic, but does not empty buffers or restore configuration registers to their default states. Both the HCC and ABC will respond to this command.
HCC PRLP Toggle	11	Toggles the sending of physics readout requests from the control path to the ABCStars. Issuing this will pause or unpaue transmission.
HCC Register Reset	12	“Harder” reset for the HCCStar. Performs a logic reset, but additionally restores configuration registers to their default values and empties all buffers.
HCC SEU Reset	13	Resets read-only status registers which toggle when the HCC detects a radiation-induced bitflip in the configuration registers. See Section 5.3 for an extensive discussion of these phenomena.
HCC PLL Reset	14	Only resets the block which generates the HCC-Star’s internal clock(s) from the 160 MHz stave clock; see Section 4.5.1.

Table 4.3: Table listing the five LCB fast commands which the HCCStar will respond to, with a brief description. Fast commands just for the ABCStar are not listed.

A fast command consists of six bits sent immediately after a K3 control code: a 4-bit command

ID and a 2-bit bunch crossing selector. Table 4.3 lists the available commands, most of which are used to reset different parts of the hybrid. Because all LCB commands are seen by both the HCCStar and ABCStar, it is not possible to indicate that a fast command should only be performed by one of the chips. Therefore, the command address space is shared, with some of the commands only used on the HCC, some only used on the ABC, and one (the “logic reset” command) used by both. Because it takes 100 ns, or four 25 ns-bunch crossings, to transmit a single frame, the 2-bit bunch crossing selector is used to define which of the four bunch crossings on which the command should execute.

Unlike fast commands, slow commands do contain metadata indicating whether they are for a (particular) HCC or ABC. Slow command start and end frames, which follow the K2 control code, contain six bits: an ABC/HCC bit, a start/end bit, and a HCC ID. The start/end bit is used to distinguish “start” frames (if set to one) from “end” frames (if set to zero). The ABC/HCC bit distinguishes commands intended for the ABCs (if set to one) from commands intended for the HCC (if set to zero). Finally, the HCC address allows the slow command to be targeted to a particular HCC rather than all of the hybrids on a stave. HCCs have a four-bit address, which can be configured dynamically, but only addresses 2 through 11 are considered valid. This means that a group of up to 10 HCCs on a stave can be connected to the same LCB command stream without losing the ability to distinguish individual chips from one another. The special address of 15 (0b1111) is interpreted as a “broadcast” address, so all chips will respond to any commands sent with this ID regardless of what address was programmed into them. Additionally, since 14 (0b1110) is always an “invalid” address, the LCB decoder will rewrite all HCC slow commands to have a HCC ID of 14 to ensure that they will be ignored by the attached ABCs.

After a K2 start frame, all slow commands will contain at least two payload frames and seven bits of metadata. Slow commands contain a read/write bit, which indicates whether the command is a register read (if set to one) or a register write (if set to zero). All registers in the HCC and ABC are 32 bits wide, and so all register writes will have an additional five payload frames with 32 bits of content to be written into the register. The remaining metadata consists of a 4-bit ABC ID, which like the HCC ID is used to uniquely identify attached ABCs; also like the HCC ID, an ABC ID of “15” is interpreted as a broadcast address. Then an eight-bit register address indicates which register should be written to or read out, depending on the type of command. The HCCStar contains a variety of writable configuration registers, as well as some read-only status registers. Reading a register will produce a register read packet in response, as described in Section 4.4.3 below.

The third type of command, the level 0 accept, is perhaps the most important. If upstream

wants to tag and read out the hit data for a given bunch crossing, it begins by sending a L0A, which will be seen by the ABCStars. Section 4.3.1 already introduced these commands: a L0A consists of a seven-bit L0 tag, as well as a 4-bit bunch crossing selector to identify which of the four 25 ns BC to tag, and a single bunch crossing reset bit. Note that at least one of the five BC selector plus BC reset bits must be nonzero for the L0 accept to be considered a valid command (as noted above, this is used to distinguish slow command payload frames from L0 accepts). The L0 accept is passed through to the ABCStars, where the tag is applied to hit data in the L0 pipeline as described in Section 4.3.1. The event is then transmitted to the HCC once a readout request command is sent.

The mechanism by which the HCC sends readout requests to the ABCs depends on whether the HCC is running in its default “single-level” triggering mode (the baseline scenario for the ATLAS trigger upgrade) or the “multi-level” mode (the evolved trigger upgrade scenario; see Section 4.1.3). For single-level triggering mode, the HCCStar will automatically generate the command to read out an event when a L0 accept is seen. This command is transmitted to the ABC along a separate serial protocol, referred to as “PRLP”. A PRLP readout command consists of eleven bits: a three bit prefix, followed by the seven bit L0 tag of the event to read out, followed by at least one zero bit to space out commands. The three bit prefix nominally distinguishes higher priority (“PR”) commands beginning with “101” from lower priority (“LP”) commands beginning with “110”, but as discussed in Section 4.6, in the current HCCStar design all commands are transmitted to the ABCStar with the same “lower priority”. These commands are sent at 40 Mbps, using the 40 MHz clock generated by the LCB decoder²⁷.

In the multi-level trigger scheme, L0 accepts do *not* automatically send readout commands to the ABCs. Instead, commands are transmitted to the HCC over a separate serial line, R3L1, using a variant of the LCB protocol. Like LCB, the same 6b/8b encoding used, and IDLE frames are sent when there is no other activity. Unlike LCB, the R3L1 protocol only supports two types of commands; [regional readout request \(R3\)](#) and [level 1 accept \(L1\)](#) triggers. In the evolved trigger scenario, R3s would be used by a hardware tracking system to read out 10% of the strip detector modules shortly after tagging an event. This would allow information from the strip tracker to be used to make a level 1 triggering decision, and if the trigger decided to accept the event, the L1 could be issued later to read out all the modules. Both commands contain the seven bit L0 tag of the event to read out. These tags will be stored in separate buffers in the control path, and any pending

²⁷In the prototype HCCStar, both LP and PR commands could be transmitted in parallel. PRs would be sent on the falling/negative edge of the 40 Mhz clock, while LPs would be sent on the rising/positive edge, thus achieving a 80 Mbps transmission rate.

R3s will be transmitted to the ABCs before any pending L1s in order to give them a slightly higher priority. Both will be translated into 11-bit LP messages and transmitted as discussed above²⁸. R3s are distinguished from L1s by a nonzero, five-bit module mask, which will be matched against the HCC ID to decide if a given HCC should respond to the command. The HCC ID x is matched to the module mask by checking the $(\lfloor x/2 \rfloor - 1)$ th bit of the module mask, so if the least significant bit is set to one, then HCCs with ID 2 and 3 would respond to the command.

In both operating modes, the control path will not continuously send readout requests to the ABCs if certain operating conditions are not met. The HCC’s input channels will receive the packets produced in response to the readout request, and must have enough space, otherwise the HCC will not transmit requests and will hold the LP commands in the control path’s buffers. Specifically, each input channel must be able to store $n + 1$ additional events of the (configurable) maximum size, where n is the number of already outstanding requests; that is, the number of requests that have been sent to the ABCStar and for which the input channels have not received packets in response. The transmission of readout requests can also be manually paused by issuing the “sending toggle” fast command; sending the fast command again will resume transmission. The control path’s LP buffers are implemented as a queue, or [First-In-First-Out \(FIFO\)](#), of size 128, so they can store up to 128 readout requests. On overflow, the oldest readout requests will be overwritten, as they may no longer be valid.

The ABCStars will respond to a readout request for a given tag by transmitting any clusters that were recorded during the corresponding bunch crossing. More detail on this response can be found in the next section, but it is worth noting here that the clusters will be identified with both the L0 tag and a shortened form of the bunch crossing ID. In the event of a glitch or radiation-induced bit error, it is possible that the metadata from one or more ABCs will become corrupted. To guard against this, the HCC contains an emulated 8-bit bunch crossing counter, which serves as a source of truth for all readout requests. When the HCC decodes a L0 accept trigger, the current value of the BCID counter will be retrieved and stored temporarily alongside the L0 tag; if the BC reset bit is set, the counter will also be reset²⁹. Once a readout request is issued for the tag in question, the BCID and L0 tag will be inserted into a dispatch queue of size 16, another FIFO which stores all outstanding readout requests which have been transmitted to the ABC. The packet builder, when

²⁸In the prototype HCCStar, the “PR” transmission mechanism mentioned in the previous paragraph was used to separate R3s and L1s, so that R3s would be sent as PRs and L1s would be sent as LPs. However, in the final version of the chip, this is no longer the case, and both external trigger commands are sent as LPs.

²⁹The counter emulates the ABCStar’s pipeline delay, so the reset will only take effect after a very large number of BC have passed. This emulated delay can be changed on the HCC by writing a configuration register.

parsing physics data, will later retrieve the tag and BCID from the dispatch queue and check this against the metadata transmitted by each ABC; errors will be generated if there is a mismatch. (More detail on this mechanism can be found in Section 4.4.3). The dispatch queue’s occupancy is also an input into the PRLP policy decision: if it is full, readout requests will not be sent to the ABCs.

4.4.2 Input Channel

Once a readout request has been sent to the ABCs for a given event, if everything is going well, the ABCs will produce packets in response containing any clustered hits they observed for the event in question. These packets will be sent in parallel to the **Input Channels** (the green block in Figure 4.5). The HCC can have up to eleven connected ABCs, and accordingly has eleven input channels. Each input channel can be enabled or disabled during runtime by writing a configuration register.

The input channel’s main task is to store packets produced by its connected ABCStar until the HCC’s packet builder can act on them. To that end, it contains three main stages: a deserializer, which decodes the serial transmission of packets from the ABCs; several buffers, which store packets of different type; and an interface to the packet builder.

An ABCStar packet is 68 bits long, and packets are transmitted at 160 Mbps. Each packet begins with a 3-bit prefix, “111”, and ends with a single “0” bit, so there must be a short pause between packets. Unlike the 6b/8b-encoded serial protocols, nothing is transmitted if the line is idle. Within this framing is a 64-bit packet, beginning with a 16-bit header and 48 bits of payload. The exact structure of the header and payload is determined by the first four bits of the header, which contain the packet’s type code³⁰ identifying whether the packet is a physics packet, a register read packet, or a high priority register (HPR) packet.

Register read packets are produced by the ABC in response to a LCB register read command, and contain the current value of a given 32-bit register. The register read packet’s header contains the 8-bit address of the register being read, with the other four header bits unused. The first 32 bits of the payload then contain the 32 bit value of the register in question. The remaining 16 bits are then used by the ABC to transmit information about the current status of the chip. HPR packets are identical in structure, except they have a different type code and are only ever used to read out the “HPR register”, a read-only register storing 32 bits of debugging information. Both the HCC and ABC have a HPR register (although the exact information stored in each differs), and

³⁰The “type” code is often written internally as a “TYP” code—presumably to save an extra letter. This is mainly done in the source code, but can also be found in specification documents.

both chips will automatically transmit a packet with its current contents every 1 ms unless this is explicitly disabled. HPR register packets are also generated in response to changes of state on the chip, such as the loss of LCB lock, and can also be generated explicitly by writing a push register. The logic for deciding when to generate a HPR is shared between the two chips.

The other main type of packet is a LP physics packet. A physics packet header, as already mentioned in Section 4.4.1, contains both the 7-bit L0 tag as well as a truncated form of the bunch crossing ID in addition to the type code³¹. The three least significant bits of the 8-bit BCID are stored, along with a parity bit computed by taking the exclusive or (XOR) of all eight bits. (As an example, the BC count of “0101011” has a parity of 0, so the value “0110” would be stored as the 4-bit “BCID”). The twelfth header bit is used as an error flag. The exact meaning of this flag can be determined by writing a configuration register on the ABCStar: it will be the logical AND of a number of possible status bits.

The 48-bit physics packet payload is divided into four 12-bit clusters. These clusters are generated by the ABC’s cluster finder block using the algorithm described in Section 4.3.2. The HCC pays attention to the “last cluster bit”, the twelfth bit in each cluster that is set to 1 if the event is complete, and if that cluster is the last cluster for the event. ABC packets are a fixed size, but as a given event can have more, or less, than four clusters, it is possible that an event’s clusters will need to be distributed across multiple packets. If the last cluster in a packet does not have the last cluster bit set, this tells the HCC to expect at least one more packet for that event with more clusters. A special twelve-bit sequence, 0x7FF, which is not a valid output from the clustering algorithm, is used to fill up the empty space in a packet if the number of clusters is not a multiple of four. If an ABC observed *no* clusters for a given event, another special sequence, 0xBFEE, is sent as the first cluster, followed by three copies of the empty cluster. Transmitting a packet in this case ensures that the HCC knows the ABC is functioning and responding to commands, even though no hits were seen.

Packets are read in by the input channel’s deserializer, which looks for the the framing bits and checks that the packet has a valid type code. What happens next depends on the currently configured HCC operating mode. In normal operations, also known as “physics mode” the packets will be transferred into buffers depending on their type code: a separate buffer for LP, HPR, and

³¹As “LP physics packet” may imply, the prototype HCCStar also supported “PR” physics packets as well, which would be produced by a PR readout request and have a type code of 0001 instead of 0010. PRs and LPs would be stored in separate buffers in the input channel, and treated separately by the packet builder (with PRs being given more priority than LPs). While the final ABCStar still has the logic to produce a PR packet, the final HCCStar no longer contains any logic to either request or store them.

RR packets. The packets are then passed separately to the packet builder, which will (among other things) combine physics packets from the same event across multiple channels into a single output packet. However, the HCC also supports two “transparent” debugging modes³². In the first, packet transparent mode, all packets are copied into the register read buffer regardless of type. The packet builder will not attempt to merge physics packets for a single event together; instead each 64-bit packet will be transmitted through the output path, unchanged. In the second, full transparent mode, the entire input channel and packet builder will be bypassed, and a single input channel’s input connected directly to the HCC’s serial data output. Which input channel is connected can be specified by writing a HCC configuration register. These debugging modes will enable the user to monitor the ABC-to-HCC connection without having to somehow add hardware between the chips to eavesdrop on that line.

The LP, RR, and HPR buffers are all implemented as FIFOs³³. The RR and HPR buffers can store up to 4 packets at a time, while the LP buffer can hold 16. These buffers are prevented from overflowing by the deserializer, and should excess packets arrive while the buffers are full, those new packets will be dropped. However, as noted in Section 4.4.1, the control path will not send additional readout requests if any input channel’s LP buffer is at risk of filling up. Each input channel counts the number of outstanding readout requests it has not yet seen LP packets for, and multiplies this by the maximum packet size for an event. If receiving another event of the maximum size, plus the current number of potential outstanding packets, would cause the LP buffer to fill up, the input channel is considered “full” for the purposes of the LP sending decision in the control path. The maximum size is a configurable value, and can be set to one, two, four, eight, or sixteen packets. If the ABC were to produce extra packets over the maximum packet size for an event, then those excess packets will be ignored.

Depending on how busy the HCC is overall, and how many clusters (and packets) are being generated by each ABC, it is possible that physics packets may need to be stored in the LP buffer for a relatively long period of time. Therefore, an effort is made to protect the physics packet metadata against radiation-induced memory corruption by using a Hamming encoding [104]. A Hamming encoding is a type of encoding which adds enough parity bits to allow single bitflips to be

³²The original HCCStar design proposed a fourth mode, “register read special mode”, which would see register read packets copied into the *physics* buffer, which is larger than the register read buffer. In this mode, other packet types would be ignored. This would, in theory, have enabled the faster reading of a large number of ABC registers than is normally possible in physics mode. This mode was left unimplemented due to a lack of time and resources.

³³In the prototype HCCStar, physics packets were stored in a RAM, rather than FIFO, of size 128x64. The RAM was divided into two segments, with PR and LP physics packets stored separately, allowing 64 packets of each type to be kept.

detected and corrected, as well as double bitflips to be detected (but not corrected). For the physics packets, the seven-bit LO tag, the 4-bit form of the BCID, the four last cluster bits, and the ABC error flag are grouped together, and encoded with six Etta parity bits. Four of these parity bits are stored by overwriting the physics packet’s type code, as it is redundant at this point since only LP packets will be stored in the LP buffer. The remaining two parity bits are stored separately. Due to resource limitations, it is not possible to protect the entire packet, but protecting the metadata ensures that the packets can still be uniquely matched to their L0 tag and that the last cluster can still be marked.

Once the system is ready, packets are then transferred out of the buffers and passed to the packet builder interface. For register read and HPR packets³⁴ the packet builder will operate on a single sixteen-bit chunk of each packet at a time. Therefore, RR and HPR packets are split into four segments, and each segment loaded into a second “output” FIFO which the packet builder will pull from. The RR output FIFO can store up to four packets (in four chunks each, with a total depth of 16), while the HPR FIFO can only store a single packet at a time. This double FIFO structure means that the input channel can actually store eight ABC RR packets and five ABC HPR packets at a time. For physics packets, the packet builder instead operates on twelve-bit clusters, including the header, which also consists of 12 bits if the header is ignored. At this stage, the “empty” cluster will be dropped and not passed onto the packet builder. Clusters will be loaded into an LP FIFO of depth 8, which is passed to the packet builder. The input channel will also check the parity bits for the packet metadata to see if any detectable bit errors occurred. In the event of an uncorrectable double bitshift, the packet will be deleted, as the metadata cannot be relied upon when merging this packet in the packet builder.

4.4.3 Packet Builder

The **Packet Builder** (the yellow block in Figure 4.5) is one of the most complex parts of the HCCStar. When handling physics data, it is the packet builder which is responsible for combining the packets produced by each ABC, and recorded by each input channel, into a single output packet that is transmitted to the end-of-substructure card and off-detector. The packet builder has to handle error conditions, such as missing packets, metadata mismatches, synchronization issues, unexpected events, and so on. It also needs to handle register read and HPR packets, both those coming from the ABCs, as well as those generated by register operations on the HCC itself.

³⁴As well as physics packets, if running in full transparent mode.

The packet builder is implemented as one large finite state machine. A finite state machine contains code broken down into different sections, or “states”, as well as rules for transitioning between those states under certain conditions. In the packet builder, the state machine continuously polls the output FIFOs in enabled input channels, looking for packets from the ABCStars. It also polls two separate FIFOs which store up to eight HCC register read packets and four HCC HPR packets. HCC register read packets are produced in response to LCB slow commands, as described in Section 4.4.1. HCC HPRs are generated automatically much like ABC HPRs, every 1 ms, as well as in response to a change in the LCB lock or if requested explicitly from upstream.

If the packet builder finds pending data in any of these FIFOs, it will transition to the appropriate state and begin constructing a packet of the appropriate type. If two different types of input are available, the packet builder will use a prioritization scheme to decide which packet type to build. This is the stage at which “high priority” register reads are given the highest priority: HCC HPRs will always be selected first, followed by ABC HPRs. Then, any pending LP physics packets will be built³⁵, followed by HCC register reads, and finally by ABC register reads.

When the packet builder handles physics data, it attempts to combine all the physics packets produced by all the ABCStars into a single LP packet for each event. The packet builder will try to wait until all the configured input channels have pending LP physics packets before it begins processing any one channel. To prevent the system from stalling in the event of missing packets, a timeout counter starts once at least one input channel reports that it is non-empty. If $6.4\ \mu\text{s}$ (or 1024 clock pulses) pass, and some channels are still missing packets, it will mark those channels as having timed out and proceed. The timeout status will be reported in an error block inserted at the end of the physics packet.

As noted above, the packet builder reads physics packets from the input channels in 12-bit chunks. The first chunk contains the L0 tag and BCID, as well as a 1-bit ABC error flag. That flag will simply be written into the error block as an ABC error bit for the relevant channel. As for the L0 tag and BCID, the packet builder will check to see if this metadata is correct by comparing it against the dispatch queue, which stores the list of outstanding readout requests dispatched by the control path. If the dispatch queue is empty, this means the HCC did not request any data from the ABCs, and so any present physics packets must be erroneous. The packet builder will enter a “flush” state, where it deletes any physics packets found in the input channels, until either all the input channels are emptied or until the dispatch queue becomes non-empty again. If the dispatch

³⁵In the prototype HCCStar, the priority “PR” physics packets would, as the names suggest, be given a higher priority here than LPs but a lower priority than the HPRs.

queue does become non-empty, the packet builder can continue flushing physics packets until 400 ns (or 64 clock pulses) pass, as it would take at least this long to transmit the requested event data from the ABCs, meaning that any packets that arrive before then are still erroneous.

If the dispatch queue is not empty, then the packet builder will not automatically flush the input channels and may attempt to build a physics packet. Instead, it will check the oldest L0 tag and BCID stored in the dispatch queue and compare them to the metadata in each input channel packet. If the L0 tag does not match, then this is considered a serious issue: we do not know if the clusters on this input channel really belong to the event that was tagged. The packet builder will not include them in the output packet, and set a L0 tag mismatch bit for that channel in the error block. We will then check to see if this L0 tag matches the next-oldest entry in the dispatch queue, as that would imply these clusters are valid data and should be preserved for the next event. If so, then the clusters are preserved and the channel in question marked as deleted. If not, the packet is just deleted, as we have no way of knowing whether the clusters are valid or not.

If the BCID does not match, but the L0 tag does, then this is considered a much less serious error. It may imply a synchronization issue with the emulated BCID counter on the HCCStar, or it could imply that the BCID was corrupted as the ABC packet passed through the input channel. Since the event can still be identified with the 7-bit L0 tag, and since the clusters may still be valid, we just set a BCID mismatch bit for that channel in the error block but do not delete the clusters. The HCCStar can be programmed by writing a configuration register to silence or “squench” the generation of these BCID mismatch bits for each individual input channel, if it turns out that many spurious BCID mismatches are generated in the real system.

Assuming that the metadata for a given input channel matches, then the packet builder begins reading clusters from that channel. The packet builder will continue reading until it either sees a last cluster bit, in which case the channel is done, or until it runs out of pending clusters, in which case the channel is not done but temporarily stalled while we wait for another packet to arrive. The packet builder will switch to another channel that does have pending data, and allow more packets to arrive for this event until the timeout counter is exceeded. At that point, if no last cluster bit has been seen, a timeout error is generated. Note that in this process, the special “empty cluster” and “no cluster” words that can be included in the ABCStar packets are dropped by the packet builder, so they will not appear in the output merged physics packets.

The merged physics packets themselves consist of a 16-bit header, followed by a list of 16-bit clusters. The header contains the LP type code, the L0ID, and BCID, as before, with one spare bit. The 12-bit clusters in the ABCStar packets are prepended with the address of the ABC that

produced them, so all clusters read in from input channel 5 will be prefixed with “0101”. In addition, the last cluster bit is dropped, and an extra “zero” added as padding instead. The list of clusters is terminated either with an “end-of-clusters” word— a 16-bit pattern (0x6FED) that does not represent a valid cluster— or with an error block. The error block is only generated if errors were observed when processing this packet, and contains the timeout, ABC error, BCID mismatch, and L0ID mismatch status bits described earlier. If present, the error block will begin with a different 16-bit pattern (0x77F4) that also does not represent a valid cluster, then contain the error bits, followed by the end-of-clusters word.

Compared to this, the non-physics packets are far more straightforward. There is no attempt to combine register read or HPR packets from multiple ABCs, as is done for physics packets. The 64-bit ABC packet is modified slightly by inserting the 4-bit ID of the ABCStar which produced the packet (numbered 0 through 10) immediately after the type code. Because the packets will be transmitted in chunks of eight bits at a time, 4 zero bits of padding are added to the end, as well, thus increasing the total size to 72 bits. If the HCC is instead in packet transparent mode, all 64-bit packets originating from the ABCStars are prepended with a 4-bit “ABC transparent” type code and a 4-bit ABC ID. Note that the packet builder does assume that the ABC RR and HPR buffers contain complete packets; that is, four sixteen-bit words. To ensure that this is the case, the packet builder will always empty these buffers immediately after a reset to ensure any incomplete packet fragments are removed.

Packet	Type Code (4 Bits)	Length
PR Physics	0001	Variable
LP Physics	0010	Variable
ABC Register Read	0100	72
ABC Transparent	0111	72
HCC Register Read	0111	44
ABC HPR	1101	72
HCC HPR	1110	44

Table 4.4: Full list of packet types that can be produced by the HCCStar, along with their 4-bit type codes and their lengths. Type codes for the PR, LP, ABC RR, and ABC HPR packets are the same as those used on the ABCStar. Note that HCC v1 can no longer produce PR physics packets (but the ABC still can).

HCC register read and HPR packets are slightly different from their ABCStar counterparts. They use different type codes, and unlike the ABC versions, do not contain any status flags. Instead, a HCC register read or HPR packet just contains the 4-bit type code, the 8-bit address of the packet, and the 32-bit value of the register in question, and four bits of padding for a total of 40 bits. Table

4.4 summarizes the type codes and lengths of the different packets the HCC (and ABC) can produce. Note that if an attempt is made to read out a HCC register which does not exist, the HCC will still produce a register read packet in response with the requested (illegal) address and with 32 '0' bits as the register contents.

All the packets produced in the packet builder are passed to the output path, which is the part of the HCC which encodes and transmits them upstream. Output packets are transmitted at either 320 MHz or 640 MHz, depending on a configuration setting. By default, output packets are 8b/10b encoded, so eight bits of data are encoded into 10-bit words [105] [106]. This uses a similar technique as the 6b/8b encoding used by the LCB and R3L1 protocols that was discussed in Section 4.4.1, and is done for similar purposes. However, there are some key differences beyond the presence of two extra bits.

For one, while every encoded 8-bit word in 6b/8b is balanced, this is not the case in the 8b/10b scheme used here. Some 10-bit words have a disparity of ± 2 , rather than zero, meaning that they have four zeros and six ones, or vice versa. Further, every 8-bit word which encodes with nonzero disparity actually has *two* unique encodings: one with a disparity 2, and one with disparity -2. Thus, to achieve an overall DC balance, an 8b/10b encoder must keep track of the *running* disparity, and select which encoding to use accordingly. If the current running disparity is -1, then the next word that is sent should either have a disparity of zero, in which case the running disparity remains unchanged, or a disparity of +2, in which case the running disparity becomes +1. Then the next word should be sent with an encoding of -2, and so on, so that the average disparity is zero.

Like 6b/8b, there are some valid 10-bit patterns that do not map to an 8-bit word and can be used as control codes. In the HCC, three of these, labelled K28.1, K28.5, and K28.6, are used for the output packet protocol. Like the LCB protocol, “idle” words (K28.1) are transmitted continuously when there is nothing pending to send. The other two control codes are used to represent the start (K28.1) and end (K28.6) of packets. Packets produced by the HCC will begin with a start-of-packet, consist of a series of 8-bit words encoded to 10 bits, and then end with an end-of-packet. Unlike the ABC packets described in Section 4.4.2, HCC packets can be of variable length, and so the packet framing clearly indicates that a packet is complete. Idle words can also appear inside a packet if there are delays in either the packet builder and output path, and can be ignored (though they may indicate problems occurring somewhere in the system).

Packets can also be transmitted in “unencoded” mode, where the 8b/10b encoding is skipped. In this mode, 8-bit words are still padded to 10-bits. A ninth “control” bit is added; if set to 1, this indicates the subsequent 8-bits should be interpreted as either a start-of-packet or end-of-packet

control code, while if set to 0, this indicates this is a normal frame. A tenth bit, that is always zero, is also added as additional padding. In unencoded mode, the output serial line will always be set to 1 if there is nothing to transmit. Unencoded mode is not intended to be used in actual operations, but rather as a debugging tool to enable packets to be decoded by eye in the laboratory without needing to consult an 8b/10b decoder (or lookup table).

4.5 Hybrid Startup and Resets

The previous two sections primarily discuss the digital logic of the hybrid ASICs. By digital logic, we mean the algorithms which have been implemented on the HCCStar and ABCStar that describe how the chips will communicate and how physics data will pass between them. One topic that has been glossed over so far is how the system will start up, and how off-detector operators will power on and reset a strip sensor module. This section describes the expected module startup sequence and some other details related to the operation of the chips.

4.5.1 Clocking

The digital logic described above is primarily synchronized to clock signals supplied to the chips with varying frequencies. As data propagates through the circuit, flip-flops and logic gates will only update on the rising edge of the clock, or when the clock transitions from a 0 to a 1, which occurs at a fixed frequency. It was mentioned in Section 4.4.1 that the HCCStar receives a 160 MHz clock from the stave alongside the LCB signal. This clock, along with the LCB and R3L1 serial lines, must be supplied for the HCC to function. It is fed into a [phase-locked loop \(PLL\)](#), a circuit which can produce a variety of clocks with varying phases from a single input signal. The HCC's internal system clock is then drawn from the PLL, and a phase offset from the stave clock can be specified by writing a configuration register.

Most of the HCCStar's logic runs off of this 160 MHz system clock, which has a period of 6.25 ns. However, bunch crossings are 25 ns, and so most of the ABCStar runs at a slower 40 MHz. The LCB decoder generates a 40 MHz clock that is synchronized to the LCB framing, and this clock signal is used in some parts of the control path that communicate with the ABC. The 40 MHz and 160 MHz clocks are then both passed to the ABCStar, since the ABC will need to generate packets at 160 Mbps. HCC output packets, meanwhile, are generated at either 320 Mbps or 640 Mbps, and clocks of this frequency are also generated by the PLL.

4.5.2 Resets

When the strips ASICs are powered on, they will first undergo a power-on reset before they are ready for configuration and data-taking. All three ASICs begin in a power-on reset state, which lasts for 6.5 ms. During this time, output signals from the chips are disabled and it will not be possible to communicate with any of the ASICs. Internal control signals are set to their default values and all internal state machines held in their initial states. This ensures that once the system comes out of the power-on reset, the hybrid will be in a known, default configuration, ready for programming.

After the initial power-on reset, it may be necessary to reset a chip, a hybrid, a module, or even an entire stave or petal in the event of problems. There are a variety of ways to reset the hybrid ASICs. Some reset commands can be transmitted to the HCC and ABC as fast commands via the LCB protocol; these resets were listed above in Table 4.3. The HCCStar, in particular, can be given a “logic” reset, which will reset logic and state machines, as well as “register” reset, which will additionally clear all memories and restore registers to their default states. However, for these resets to work, LCB communication with the chips must be possible. If the LCB decoder is unlocked or stuck in a bad state, it would not be possible to reset the chips via this method.

The AMAC, which sits on the power board, is connected to the HCC. The AMAC has its own custom serial protocol, known as Endeavour³⁶. A command can be sent to the AMAC to pull the HCC’s “hard reset” pad, an input on the HCC wired directly to AMAC. The hard reset, if invoked, behaves much like the power-on reset, and will fully clear all memories, restore registers to default states, and hold state machines in their default states until the hard reset is lifted. The HCC, in turn, is connected to the hard reset pad of the attached ABCStars. Once the HCC has been reset by the AMAC, a LCB command can be sent to write a configuration register which controls the ABCStar reset.

An AMAC on a given module is also responsible for powering up the HCCs and ABCs sitting on that module. It can put the chips into, or bring them out of, “low power” mode. In low power mode, all the inputs and outputs are disabled, including clock generation. With the clocks disabled, the chips will more or less be fully inactive, thus considerably reducing power consumption.

The AMAC itself can be reset by sending serial commands over the Endeavour protocol. If for some reason the AMAC is not responding to serial commands, the HCC is capable of resetting the AMAC’s decoder and command interpreter. This is done by writing a configuration register

³⁶The AMAC serial protocol was developed by Bill Ashmanskas, and is based on Morse code. It’s named “Endeavour” after the character of Endeavour Morse, from a series of detective novels by the British author Colin Dexter. A full description of the AMAC serial protocol is beyond the scope of this thesis; as with the other chips, interested readers with access should consult the AMAC specification document.

on the HCC, which toggles an output pad connected directly to the AMAC that will reset the communications block. Modules may contain more than one HCC per AMAC, and so in modules with two hybrids this reset can be issued from either HCC. In extreme circumstances, a module's AMAC can be reset by a neighbouring module's AMAC as well.

4.6 Strips ASIC Development Process

Because ASICs are designed for a specific purpose and cannot be reprogrammed, producing an ASIC like the HCCStar can be an involved, lengthy process. The digital logic first needs to be written in a [HDL](#), a programming language specifically intended to describe circuitry in a hardware project like an ASIC. For the strips ASICs, the digital logic described above was primarily written in the [Verilog](#) language. Turning this Verilog source code into a physical object which will ultimately be installed in the ATLAS detector involves several key steps:

1. First, the Verilog source code will be compiled, translating a high-level description of the logic into a lower-level list of connections between logic gates and other primitive circuit elements. This process is known as *synthesis*, and the output list of logic gates is referred to as a *netlist*. For the HCCStar, we use the Genus synthesis compiler developed by Cadence Design Systems, a hardware design company.
2. Next, the output from the synthesis undergoes a process known as *place and route*, in which the logic gates are laid out and fit into the available space. A single HCCStar will ultimately occupy a space of 4266 by 5200 μm , meaning that the transistors, logic gates, and circuit elements from the synthesis process must be fit into this space. Analog components such as the PLL, or the driver and receiver circuits connected to the chip's inputs and outputs, are added to the design at this stage as well. For the HCCStar, this is done using the Cadence Innovus Implementation System.
3. Final testing of the placed-and-routed design is performed. Both digital and analog logic must be checked before the chip can be submitted.
4. The ASIC design is submitted for fabrication. The three ITk Strips ASICs (the HCC, ABC, and AMAC) will all be produced by Global Foundries using their 130 nm CMOS8RF transistors.

5. In the fabrication process, the placed-and-routed layout is turned into lithographic *masks*, which will be applied to many large silicon *wafers* to create the ASICs themselves. A single wafer will contain more than one ASIC, and will need to be *diced* before the chips can be used.
6. Once the chips are produced, they will be tested. If problems are found in the testing, another version of the chip may need to be produced, restarting the entire process from the beginning. If not (and if other components, such as the other ASICs and the sensors are ready), then more wafers can be produced and this version of the chip used in the assembly of strip sensor modules.

For the strips ASICs, the ABCs are fabricated separately, while the HCCs and AMACs will be submitted together. As of this writing, prototype HCC, ABC, and AMAC designs were produced in 2018 and tested over the course of 2019. Second versions of all three chips are in various stages of development: a second iteration of the ABCStar was fabricated in 2020 and is currently undergoing physical testing. The second iterations of the AMAC and (especially) the HCCStar took longer, with the designs finalized on August 16th, 2021 and submitted for fabrication shortly afterward. Some of the changes to HCCStar version 1 have already been discussed above, but to summarize them again here, there were three significant changes to the digital logic:

- The priority (PR) physics path was mostly removed. While the ABCStar still retains support for producing physics packets of type PR, the HCCStar’s control path will no longer generate PR requests. Instead, the control path will translate R3 commands to LP readout requests, but preferentially transmit pending R3/LPs to the ABCs over L1/LPs. This was primarily done for two reasons. First, performance studies demonstrated that full support for PR readout requests was not necessary to satisfy timing requirements for a regional readout-based hardware tracker anyway. Second, as mentioned in Section 4.1.3, the ATLAS collaboration made the decision to not proceed with a regional readout-based hardware track trigger, at least for the initial runs of the HL-LHC³⁷. Therefore, removing support for PR packets from the input channels and packet builder reduced the overall amount of space taken up by the logic, increasing flexibility when attempting to lay out and route the chip. R3/LP translation was implemented in the control path to preserve some support for regional readout in the future, should this decision be reconsidered.

³⁷Additional discussion of this decision and the proposed hardware track trigger system can be found in Appendix B.

- The physics packet buffer in the input channel, which was implemented as a RAM in the prototype, was replaced with a smaller FIFO. In the prototype, 64 LP packets could be stored (with another 64 PR packets); in version 1, this was reduced to only 16. Given that, as noted in Section 4.3.2, the expected occupancy per ABCStar is expected to be on the order of 2 clusters per event, it is very unlikely that most events will ever have more than one packet. Performance studies demonstrated that the input channel buffers would almost never reach even a quarter of their original capacity, and that it would be safe to reduce the buffer size to 16 without a risk of data loss. Accordingly, the default maximum number of packets per event was reduced from 16 to 4. In addition, in the prototype, two of the parity bits computed for physics packet metadata were stored in a separate FIFO; in version 1, this FIFO has been removed and the LP buffer’s width increased from 64 to 66. These changes further reduced the amount of space taken up by the digital logic.
- To better protect the chip against the effects of ionizing radiation, redundant copies of much of the digital logic were added in a process known as triplication. Section 5.3.3 describes this technique and the work done to protect the HCC in considerable detail in the next chapter. Even with the space saved from the first two changes, including redundant copies of the logic considerably increased the amount of space taken up by the design. Therefore, in order to place and route the logic, it was necessary to make version 1 of the HCC wider by 766 μm compared to the prototype.

As the above might suggest, producing an ASIC is an expensive and time-consuming process. Writing digital logic in a hardware description language is not too dissimilar from writing software, or even writing firmware for a [FPGA](#), except that it costs hundreds of thousands of dollars and at least several months³⁸ to “compile” the source code. This high cost to development means that it is critical to do everything possible to reduce the risk of an ASIC not working during development, including simulation and virtual testing of the digital logic. The HCCStar, in particular, has quite complex digital logic, and so an extensive simulation and validation campaign was performed to verify the correctness of that logic before submission. This simulation campaign is described in detail in the next chapter.

³⁸Several months assuming the foundry is not busy with other work. The final version of the HCCStar was being prepared during the global “chip shortage”, partially induced by the COVID-19 pandemic, in which global demand for electronics and limited foundry availability has led to severe delays and shortages for all ASIC production across the world. The exact effect of this shortage on the ATLAS upgrade and HL-LHC schedule is unknown, as of this writing, but could have serious consequences if further revisions to the HCCStar are needed.

CHAPTER 5

HCCStar Verification

To ensure that the [HCCStar](#) will function as described, the digital logic discussed in Chapter 4 has been simulated in a process known as verification. The Verilog source code that describes the chip is loaded into a simulator, which creates a virtual, version of the ASIC that can be controlled via dedicated test software commonly known as a “testbench”. The testbench is responsible for sending input stimulus to the simulated HCC and monitoring the output produced in response in order to determine whether the digital logic is working as designed. Simulations are performed both by running directly over the source code, at what’s known as the [register-transfer level \(RTL\)](#), or by running over the outputs from the synthesis and place-and-route process, at what’s known as the gate-level.

The verification of the HCCStar was performed using two main tools. First, the Incisive and Xcelium simulators from Cadence Design Systems³⁹ were used to simulate the digital logic itself. Then, the testbench software were written in the Python programming language using the open-source library cocotb to control the Cadence simulator [107]. By using cocotb and writing the testbench in a general-purpose programming language like Python, the verification and modelling of data flow through the system could be done at a higher level than the implementation of the design. This was very useful in enabling physicists with limited hardware experience, such as myself, to work on the verification [108].

This chapter focuses on describing how cocotb was used as part of the testing of the HCC, and not on cocotb itself. For more details on cocotb, including the philosophy behind its approach to verification and some examples of its usage (including actual source code), please consult Appendix

³⁹Cadence Xcelium was launched in 2017 to replace the older Incisive simulator. The prototype HCCStar was designed in 2017 and 2018 using Incisive, and we upgraded to Xcelium for the next iteration of the HCC in 2020 and 2021.

A. This appendix can be read either as background material for this chapter, or as a standalone document for readers interested in the use of cocotb for other, non-ITk projects.

Using cocotb, three main sets of simulations were developed to verify the HCCStar. First, Section 5.1 outlines the “standalone” functional verification framework built to test the HCC. Then, Section 5.2 describes the “hybrid-level” and “module-level” simulations, in which the HCCStar is simulated together with the source code for the ABCStar (hybrid-level) and AMAC (module-level). And thirdly, Sections 5.3 and 5.4 outlines the effects of ionizing radiation on the HCC’s digital logic and the work to simulate and mitigate these effects.

5.1 Functional Verification

Functional verification of the HCCStar focuses on testing the features of the chip using a series of “unit” tests. Each unit test is targeted to a single, specific feature of the digital logic or scenario that the chip might encounter. If a new feature is added to the chip, or a new edge case discovered that needs investigation, a new unit test can be written. Then the full suite of unit tests can be ran to fully verify the HCCStar’s functionality.

Verification work on the prototype HCCStar began by building a dedicated testbench for the control path (described in Section 4.4.1), with particular focus on the LCB protocol. Based on this work, a full “standalone” testbench for the entire HCCStar was then developed, which consists of many unit tests. This standalone testbench was kept up to date as the HCC’s design changed, and used to test both the prototype (v0) and final (v1) versions.

5.1.1 LCB Testbench

Verification of the prototype HCCStar was done in parallel with work on the design. To help build experience using cocotb, the first major verification effort in early 2017 was to simulate the control path, which at the time was the most complete block of the chip. To that end, a Python testbench for the control path’s serial protocols was developed.

The focus of this work was on creating a complete simulation for the LCB and R3L1 serial protocols, described in Section 4.4.1, which are used to send 6b/8b-encoded commands to the HCCStar. Python implementations of the LCB and R3L1 protocols were developed that could translate triggers, readout requests, slow commands, and fast commands into 8-bit encoded words. These 8-bit encoded words would then be serialized into a 160 Mbps stream and transmitted bit-by-bit to the control path block through the cocotb interface, where they would be interpreted by the chip.

In the LCB simulations, the initial focus was on verifying that the output LCB passthrough stream appropriately matched the input generated by the testbench⁴⁰ and that the 40 MHz clock generated from the LCB decoder was properly in phase with the LCB framing. Once this was proven to work, additional monitoring was added to the interfaces between the control path and the (unsimulated) rest of the chip. For instance, once the LCB decoder decodes a slow command, it tells the register block the address of the register to read or write, and– if writing– the contents to write. The testbench was made to explicitly check that the correct address and contents were decoded from the slow command. Finally, simulations of various error conditions were done, focused on the decoder’s handling of invalid frames, its automatic unlocking after seeing a sufficient number of decoding errors, and its ability to relock and resynchronize to the stream after again observing sixteen good frames.

Like the LCB simulations, the R3L1 simulations focused on verifying that the readout requests produced from the control path matched the R3 and L1 commands sent to the HCC. This involved writing a Python monitor for the PRLP serial protocol, which would be able to capture the PR and LP commands being transmitted to the ABCs⁴¹ and match the L0 tag to the R3 or L1 command which produced it. Tests focused on understanding the various flow control mechanisms in the control path, and ensuring that readout requests would not be sent if the (unsimulated) input channels claimed to be full. Tests were also done to ensure the PR and LP buffers would behave properly on overflow, deleting the oldest readout requests.

Using this testbench, long combined LCB/R3L1 simulations were performed in both single-level and multi-level trigger mode configurations. These simulations were intended to approximate realistic HL-LHC running conditions, with triggers and readout commands sent to the chip at random intervals at the planned ATLAS trigger rates. In the single-level simulation, L0 accepts were sent at an average rate of 1 MHz, and in the multi-level simulation, at 4 MHz. In the multi-level simulation, R3s and L1s were also transmitted at average rates of 400 KHz. For both simulations, the testbench monitored that the L0 accepts appeared in the LCB passthrough stream and that the expected PRs and LPs were generated and sent on the PRLP line, either automatically from the L0A (in single-level mode) or from a R3 or L1 (in multi-level mode). To complement the flow of triggers, bunch counter resets were inserted every orbit, or 891 frames, with fast commands, slow

⁴⁰The LCB decoder will rewrite the start and end frames of HCC slow commands, replacing the HCC/hybrid ID with an invalid address to make it harder for the ABCs to mistakenly act on a HCC slow command. Aside from this rewriting, the LCB input and LCB passthrough are expected to exactly match.

⁴¹At the time this work was done, the HCCStar *could* produce both PR and LP readout requests, and so the verification needed to be able to handle both cases.

commands, and superfluous idle frames inserted randomly as well. The simulations would also insert random single event errors, or bitflips, in order to generate decoder errors by randomly toggling a bit from 0 to 1 or from 1 to 0. This was done with a 1% probability per frame in both the R3L1 and LCB streams.

The LCB testbench, and these long simulations, were used extensively during the early development of the prototype HCCStar in order to find problems in the control path. These simulations were also useful in validating changes made to the LCB and R3L1 decoders, and in catching regressions as they occurred. More broadly, the experience using cocotb for these control path simulations was a very positive one. Complex tests could be written rapidly and in a reusable way; the LCB and R3L1 parts of the simulation were originally developed separately before being merged into the same testbench. Using Python, it was also straightforward to make the tests self-checking by modelling the expected behavior of the control path in response to its inputs, and automatically detecting errors in the outputs. This experience convinced us to continue using cocotb for the verification of the full chip, which is discussed below. It also led to the use of cocotb for the verification of the [AMAC](#), which was also designed by the Penn instrumentation group.

5.1.2 Standalone Testbench

Drawing upon the work done to build the control path testbench, a full testbench for the HCCStar was developed using cocotb. This verification suite is commonly referred to as the “standalone” or “HCC-only” testbench, because it runs over the full HCC and does not require any other components (as opposed to the hybrid- and module- level simulations discussed later in this section). The standalone testbench has become the primary tool used to verify the chip’s functionality, as the control path testbench was not kept up to date with changes made in HCC version 1.

The standalone verification suite is broken down into 104 dedicated tests. Each test instantiates a reusable HCCStar testbench class, and then uses this object to test specific areas of the chip’s logic. Like the control path testbench, the standalone testbench class contains Python implementations of the HCC’s serial input protocols as well as code to monitor and parse the HCC’s outputs. The two 6b/8b encoded inputs, LCB and R3L1, have Python drivers which can encode and serialize commands. The testbench also contains LCB passthrough and PRLP monitors, to record the transmission of commands to the ABCStars sitting on the hybrid. The LCB monitor attempts to lock to the LCB passthrough stream by observing sixteen valid idle frames, just as the real ABCStar would, and will unlock (and generate an error) if enough bad frames are seen. Then, when a test

sends a LCB command or readout request, these monitors will automatically ensure the expected command is seen, and fail the test if this does not occur.

Unlike the control path verification, the standalone path also needs to be concerned with the HCC’s “data” path: the input channel and packet builder, as described in Sections 4.4.2 and 4.4.3. To test these areas of the chip, it is necessary to be able to produce ABCStar packets and load them into the input channels. One way to do this would be to load the ABCStar’s source code into the simulation alongside the HCC, attach the real ABCStar to the LCB and PRLP outputs, and use it to produce these packets. The standalone testbench is so named because it does not take this approach. Instead, a Python model of the ABCStar data flow was written that can be attached to the LCB passthrough and PRLP monitors. This model, referred to internally as the Python or “fake” ABCStar, will produce appropriately formatted physics, register read, and HPR packets in response to commands.

Physics packets are generated in response to PR and LP readout requests⁴². By default, packets are produced with a randomized number of random clusters, but a given test can choose to send a fixed number of clusters (random or predetermined) for a given L0 tag. By default, the random number of clusters is determined by sampling a Poisson distribution with $\lambda = 2$, the expected average occupancy per ABCStar per event. The fake ABC can produce more than one packet in response to a readout request if the number of clusters are high enough, in which case the last cluster bits will be set accordingly. The Python model will store the clusters generated for each L0 tag so that the testbench can compare them with any clusters observed in the output physics packet produced by the packet builder, and complain about any mismatches.

The Python ABC model also contains a bunch crossing counter, so that physics packets can be produced with “correct” BCIDs. The model does not attempt to emulate the ABCStar pipeline. Instead, when a L0 accept is seen by the testbench, the Python model will record the current value of the BCID counter so that this value is available when a readout request arrives for the corresponding L0 tag. Like the real implementation, this Python counter counts up every 25 ns, and will respond to BCRs if that bit is set in a L0 accept. The lack of a pipeline implementation means that to use this model, the HCCStar’s configurable pipeline delay must be set to nearly zero, as the only delay in applying the L0 accept will be transmission time from HCC to (Python) ABC. Note that if a L0 accept does not arrive before a readout request for a given tag, the model will generate a random BCID and print a warning that this will likely result in a mismatch in the output packet.

⁴²Like the real ABCStar, the Python model still contains logic for responding to a PR readout request even if version 1 of the HCC is no longer capable of producing them.

The ABC model can also produce register read and HPR packets. Register read packets are generated in response to ABC register read commands sent via the LCB protocol. Produced packets will contain the requested address in their metadata, but the actual register contents will be randomized. (The model does not implement a register model of the ABCStar, and it does not react to ABC register *writes*). Because we can compare the randomized value transmitted to the input channel with the value seen in the output HCC packet, this is sufficient to ensure that the HCC does not corrupt the contents of ABC register read packets. A similar approach is used for ABC HPRs. The Python model can be configured to produce ABC HPRs periodically (with random contents) $500\mu\text{s}$ after starting the simulation, and 1 ms thereafter, as the real chip would. ABC HPRs can also be manually sent by calling a Python method from test code, should a test need to explicitly request them.

In addition to a packet generator, a new monitor is needed to parse and record output packets produced from the packet builder. To synchronize to the serial output stream, the testbench makes use of the HCCStar’s data clock debugging output, which will emit either a 320 MHz or 640 MHz clock depending on the configured readout speed. The HCCStar output packet monitor will then use an 8b/10b decoder to parse encoded frames, and will find the the start-of-packet and end-of-packet framing to build Python packet object. Like the LCB monitor, the output packet monitor will attempt to “lock” to the 8b/10b frame, and can unlock and relock over the course of the test if parsing errors are encountered. Idle frames will be disregarded, although the number of idle frames that appear within a packet will be recorded, as this may be a sign that the system is overloaded or experiencing difficulties. The packet monitor also supports running in unencoded mode, in which case it will wait until it sees the leading “0” bit to indicate activity and then parse the unencoded 8-bit words into packets.

Test code can tell the testbench to expect specific types of packets, and fail the simulation if they do not arrive; a test which performs a readout request will then wait until it sees a LP physics packet, for instance. In addition to any checking done explicitly as part of the test, the output packet monitor can attempt to perform a large number of automatic checks on the packets it receives. First, it will attempt to parse each packet by inspecting the type code, and then print out a parsed version to the simulation log. Since register read and HPR packets (as well as the transparent debugging packets) all have fixed sizes, this is a way of checking that the received packets are properly formatted. If we receive a packet claiming to be a HCC register read packet, but it’s twice or half the length of an HCC register read, then we can tell immediately something has gone wrong. Similarly, physics packets have some internal structure despite being of variable length, so we can check to make sure

that the “end of clusters” word is in the right place, or that the error block is correctly formatted if present.

Next, assuming the packet’s structure has been successfully parsed, the checker can attempt to check its contents. As has already been mentioned, for physics packets, ABC register read packets, and ABC HPR packets, this can be done by simply inspecting what was transmitted by the Python ABCs and comparing with what was observed. For register read and HPR packets, this is as simple as ensuring that the register address, register contents, and status bits matched what was transmitted from the relevant Python ABC model.

For physics packets, we use the observed L0 tag to look up all the physics packets produced on all input channels with that tag. We expect that the clusters and metadata (BCID and ABC error flag) in the input packets will exactly match the metadata in the output packet, unless the output packet contains an error block. So, for example, if a BCID error bit is set on input channel 5, then the checker expects that there will be a mismatch between the BCID used on input channel 5 and the BCID in the output packet. If the BCIDs agree, this is considered an error, because it means that either the BCID was corrupted inside the HCC or the BCID mismatch bit was improperly sent. The same approach is adopted for other error block information, except for L0 tag mismatches: these are considered errors by default, since L0 tags are used to look up truth information in the ABC model.

An attempt is also made to automatically verify HCC register read packets. HCC register reads are only produced in response to a LCB register read command, so the testbench keeps a record of outstanding register reads that have been sent to the chip. If it receives a register read packet, it checks to see that the address of the register in question was indeed requested by the testbench. In addition, the contents of some HCC register read packets are automatically verified for read/write registers. At the beginning of simulation, the testbench loads the default values of every read/write register into memory. Then, as HCC register write commands are issued over the course of the simulation, the testbench keeps track internally of the expected values of those registers. When a register read is issued, the testbench will check to see if the observed value in the output packet matches the value it thinks has been programmed into the chip, and will log an error if they do not agree. The testbench does not attempt to predict the value of read-only registers, or of the HCC HPR register.

These drivers and monitors are enabled at the start of each test by instantiating the HCCStar testbench class. The testbench class contains a number of high-level Python functions for interacting with the chip, transmitting commands through the drivers, or retrieving data from the monitors.

Once the testbench class has been set up, the test then uses it to fully reset the chip, first waiting for the power-up reset to clear (if this is the first test to run), followed by a full hard reset, a register reset, and a logic reset. By fully resetting the chip before each test, we ensure that the chip is in an expected, “clean” state before beginning each test. Once the chip comes out of reset, the test can then program configuration registers via the LCB protocol, turning on the hybrid-side drivers that generate the LCB passthrough and PRLP outputs, as well as enabling as many input channels (and Python ABCs) as needed. The exact configuration varies from test to test, depending on what is needed for the test in question. A test focusing on LCB passthrough or on HCC register operations might not need any input channels, while a test focusing on physics data would.

The first test in the test suite is known as the regression test. This test performs the startup sequence as just described, and then proceeds in attempting to verify key functionality with one input channel connected. The test sends a L1 readout request, a R3 readout request (both with corresponding L0 accepts), a HCC register read, an ABC register read, and an ABC HPR. It then checks that the expected output packets are produced from the HCC, and contain the expected contents. This test only takes a few seconds to run, and can be used to quickly see whether the chip is functioning properly.

The remaining 103 tests each focus on testing specific parts of the digital logic, as well as looking at specific edge cases where we are interested in understanding what the HCC will do under certain circumstances. 83 of these tests were written during the verification of the prototype HCCStar, with an additional 20 added in the development of HCCStar version 1. These tests can be broken down into several key categories:

- 27 dedicated packet builder tests. These tests focus on different aspects of the data path, testing different parts of packet builder functionality and different edge cases. The first few tests look at simple scenarios, like multiple physics packets per event, the combination of packets from multiple input channels, and so on. Other tests cover error conditions, looking at the creation of the error block, the flushing of bad data, the handling of timeouts, and so on. These tests all run with the packet builder in normal “physics” mode.
- 7 tests focused specifically on the various alternate readout modes and other alternate operating conditions. There are dedicated tests for the packet transparent and full transparent debugging modes, as well as tests of the “unencoded” transmission of output packets from the output path.

- 12 dedicated control path tests. Arguably, the entire test suite consists of control path tests, as nearly every test sends at least one LCB command to the chip. These 12 tests focus on covering additional, untested cases, like looking at how the LCB and R3L1 decoders handles various errors, like improper slow command framing.
- 17 tests of radiation-induced bit errors across different parts of the HCCStar. Radiation can cause registers and other bits inside an ASIC to change state. To protect an area of the chip against this, it is necessary to add redundancy, so that a single bit flip can be detected and corrected. These tests focus on verifying that this protection works in key areas of the chip by simulating bit flips and checking to see if they are automatically corrected by the chip⁴³. Much more information on this type of error, the work done to protect against the chip against it, and the testing of that protection can be found in Section 5.3.1.
- 11 tests that send random, erroneous data to the HCCStar’s serial inputs. Some of these tests randomly inject glitches or other pulses into otherwise valid commands. Others just generate random bit patterns that may or may not bear any resemblance to a valid LCB/R3L1 command or ABCStar packet. These tests focus on ensuring that the HCC can recover properly once random injection finishes by seeing if it possible to send valid serial commands again without having to first reset the chip.
- The remaining 30 tests cover a wide variety of other behavior. Some tests explicitly cover HCCStar register operations, or the HPR status register. Some test other functionality like the HCCStar’s AMAC communications reset, by checking to see that the AMAC reset output toggles in response to a register command. Still others focus on exploring various edge cases and attempting to stress the system by filling up various buffers in the input channel and control path.

The testbench can run both RTL simulations and gate-level simulations. Gate-level simulations can be performed both immediately after synthesis and before place-and-route, or on the post-place-and-route (post-PNR) design. At the RTL level, the full suite of unit tests takes about 1.5 hours to fully run; post-synthesis and post-PNR simulations can take up to another hour.

⁴³The tests here only cover parts of the design that were protected against ionizing radiation in the prototype HCCStar v0. This includes the configuration registers, physics packet metadata in the input channels, and some key state machines.

5.1.3 Code Coverage

To measure the completeness of the standalone testbench, we can look at its code coverage. In software engineering, coverage is a way of evaluating how much of a project's source code is executed by its test suite. An analogous concept exists for digital logic verification, where the simulator can report what percentage of the logic in the design being tested is exercised when running the verification. We used this information to guide the development of the HCCStar testbench and to evaluate its relative completeness.

The Cadence reports report three coverage scores: block, expression, and toggle coverage. To understand the different meaning of each score, let's consider an example. Suppose we wanted to evaluate the coverage of an OR gate, a logical circuit which takes two inputs and evaluates to one if either the first *and* second input are one; otherwise it is zero. Source Code 5.1 shows a potential implementation of such a circuit in the Verilog language, written using an if-else statement.

```
input wire a, b;
output wire x;
if (a == 1 && b == 1) begin
    x = 1;
end else begin
    x = 0;
end
```

Source Code 5.1: Verilog circuit that implements an AND gate using a conditional statement. Note that an AND gate could be written in a more compact way as `x = a && b`, but writing it with a conditional is useful for the purposes of understanding different types of code coverage.

This circuit has two blocks of code: the true branch of the if statement (when both *a* and *b* are equal to one), and the false branch (when they are not). To measure block coverage, we would track whether or not both blocks are executed over the course of the simulation. If the true branch is never executed, but the false branch is, then the block coverage would only be 50%. To get full block coverage, the simulation would need to have executed both the true and false branches of the conditional at some point.

We can learn more about the coverage of this conditional statement by looking at the expression coverage, which reports what combination of states the expression `(a == 1 && b == 1)` sees over the run of the simulation. This expression has four possible states, as *a* and *b* can either be zero or one independently of each other. Therefore, for this expression to have full coverage, we would need all four of these states to occur. Note that we could have full block coverage without having full expression coverage, if *a* is always true but *b* varies.

Finally, toggle coverage simply reports whether or not every variable bit in the design changes value. To have full toggle coverage, each signal must change both from a 0 to a 1 and from a 1 to a 0. In the source code above, to get full toggle coverage, we would need a , b , and x to all fully toggle. There are various ways this could not occur while still maximizing block and expression coverage, for instance by changing the inputs in such a way that x toggles from a 1 to a 0 but never back to 1. It is therefore important to consider all three scores when understanding the coverage of the design.

It is important to note that this is an imperfect metric for evaluating a testbench. The coverage merely reports whether the simulation executes all the source code. It cannot tell us whether the testbench’s checking is complete enough to catch any errors as a result of fully covering the chip. For example, consider a test which fills the physics packet buffers in the HCCStar’s input channel. The coverage report can tell us whether or not those buffers ever fill up. But of course, it cannot tell us whether or not the chip behaves reasonably if this occurs, or whether physics data will be lost.

Also, it may be the case that it is not necessary to fully cover all the logic. There may be source code in the design that is uncoverable for various reasons. In some cases, this could be because there are bits or logic in the design that are actually unused (perhaps a register has unnecessary bits) that will be optimized away by the synthesis compiler when translating the design into logic gates. In other cases, it could be because of limitations in the coverage tools themselves. One example of the latter can be found in instances of the Verilog case/switch statement, often used to implement a state machine. As shown in Source Code 5.2, it is possible for this to lead to uncoverable blocks of code.

```
reg [1:0] state_register;
case (state_register)
    2'b00:      // State 0
    2'b01:      // State 1
    2'b10:      // State 2
    2'b11:      // State 3
    default:    // There is no fifth state!
endcase
```

Source Code 5.2: Example of a Verilog switch statement used to implement a state machine. The first line declares a 2-bit state register. Then depending on the value of the state register, a different state is executed. Case/switch blocks always have a “default” state, which is either implicit or explicit. But here, it is impossible for a two-bit state register to have more than four possible values, meaning the fifth default state can never be reached. The coverage tools, however, do not know this and will report that this circuit would only have a block coverage of 80%.

Name	Overall (%)	Block (%)	Expression (%)	Toggle (%)
v0 (Average)	94.99	97.55	83.02	94.53
v0 (Cumulative)	87.29	90.52	74.42	87.33
v1 (Average)	95.64	99.08	93.99	94.14
v1 (Cumulative)	96.08	97.19	93.03	96.02

Table 5.1: Coverage scores for the HCCStar v0 and v1 designs, listing the overall, block, expression, and toggle scores separately. “Average” and “cumulative” scores are reported for both v0 and v1 designs. Average scores are computed by averaging the coverage of all submodules regardless of how much logic each submodule contains. On the other hand, cumulative scores are computed without regard to internal module boundaries. Both values are useful metrics, but the average scores are biased upward by a number of very small Verilog blocks that are fully covered. Therefore, the cumulative scores are taken to reflect the true coverage of the design.

With those caveats in mind, let’s consider the coverage of the HCCStar. This coverage is evaluated through a RTL simulation, from running the standalone testbench directly over the Verilog source code⁴⁴. At the time of the prototype HCCStar’s submission in 2018, the standalone testbench achieved a total coverage of 84%. As part of preparations for the final version of the HCCStar, a thorough review of the code coverage report was performed to determine what parts of the chip were not being tested and what parts needed additional testing. As a result of that review, some unreachable code was removed and additional tests were written, with the result that the overall coverage has increased to 96.08%. Table 5.1 shows a comparison between the block, expression, and toggle scores for both the v0 and v1 verification.

The coverage review identified several areas of the design which needed further work, one key area of which was the LCB decoder. Because initial verification work on the HCCStar had started by writing a dedicated control path testbench, and because both the standalone and control path testbenches were used during the development of the prototype, a full set of LCB tests (focused on error conditions, like invalid slow command framing) had never been added to the standalone testbench. This was rectified for HCCStar v1, which helped increase the control path coverage significantly. The coverage review also identified uncovered states in the packet builder and HPR state machines, which new tests were written to cover.

Additionally, the coverage review showed that several of the input channel buffers, as well as the dispatch queue, were never being filled by the functional verification. New tests were written to explicitly explore this phase space, fill these buffers, and verify what happens should they overflow. Having solved these issues, we are reasonably confident that all major parts of the design are exercised

⁴⁴The coverage analysis is technically computed from a “non-triplicated” RTL simulation. That is, code which is installed to protect the HCCStar against radiation-induced single event effects is not included when the coverage is computed. See 5.3.3 below for a discussion of the triplication process and how it hardens the HCC against radiation.

by the standalone testbench.

5.1.4 Continuous Integration

Having written a testbench with reasonably high code coverage, it is important to regularly run it as changes to the design are made. It is regrettably very common to unintentionally introduce bugs when making apparently unrelated changes to a project, and digital logic design is no exception. Regularly running the entire simulation ensures that changes to the design can be immediately validated, and any new errors quickly identified. In addition, because some of the tests use random inputs, running the tests regularly can help reveal problems that might not occur every time the tests are run due to a different initialization of the random number generator. In software engineering, the practice of regularly running a test suite for a project is known as continuous integration, and the same approach has been adopted for the HCCStar verification.

Two levels of continuous integration have been deployed. The HCC's source code, as well as the source code for the testbench, are stored in a repository using the `git` version control system. When a developer makes a change to the project and checks their change into the repository, a set of continuous integration tests are performed on that change. The regression test is run on the RTL simulation, and then the synthesis compiler is invoked. If the synthesis is successful, the regression test is run again on the post-synthesis simulation. This CI check helps validate both changes to the Verilog design and to the verification environment, ensuring that changes to either have not prevented the simulation from starting up successfully and that the chip is still minimally functional.

Second, a collection of scripts were written to run the entire testbench every night, in a number of different configurations. Each evening, the latest version of the project's source code is checked out from version control. RTL, post-synthesis, and post-PNR simulations are then launched using the Unix utility `cron`. The full set of tests in the standalone testbench is run for each, and a summary of the results emailed to the development team with a list of any test failures. Code coverage reports are generated automatically from the RTL simulations. For the gate-level simulations, the coverage report also contains a list of any timing violations logged during the run. A timing violations can occur when a signal's state changes state in such a way that would be physically impossible: for example, in reality, it will take a certain amount of time (perhaps 10 to 100 ps) for a flip-flop's state to change, but in a simulation, it is possible to force the value of a flip-flop to toggle faster than this. This is often represented in the simulator as an 'x'; something unphysical indicating a metastable

state that could be interpreted as either a one or a zero. The simulator will log that this occurs, and the CI scripts will report this information as part of the nightly report.

The nightly CI scripts are designed to be flexible, and take a configuration file as input that specifies what simulation to run, what arguments to pass to the simulator and cocotb, whether the synthesis compiler needs to be ran first, what log files should be kept, what additional information should be included in the email report, and so on. In addition to just running the standalone testbench, the CI system is used to run a variety of other jobs, such as the hybrid and module level simulations discussed below, as well as the single event error simulations covered in Section 5.3. The same scripts have also been used to run the AMAC testbench, which, as noted above, was also created using cocotb.

5.2 Hybrid and Module-Level

The standalone testbench is so named because a Python model of the ABCStar is used to test the HCC. Instead of using a model of the ABCStar, it is possible to load the real ABCStar source code into the Verilog simulator alongside the HCCStar source. Then, the ABC design can be attached to the HCC's clock, LCB, and PRLP outputs, as well as being connected to an input channel. Commands sent by the testbench to the HCC would then be forwarded to an actual instance of the ABCStar, which would produce packets in response. This approach allows a simulation of an entire hybrid, and is therefore referred to as a *hybrid-level* simulation.

Similarly, this approach can be further generalized to the *module-level* through the addition of the AMAC as well. While the AMAC only occasionally communicates with the hybrid ASICs, it plays a vital role in starting them up when a module is powered on and resetting them if something goes wrong. This can be verified before submission by loading an AMAC into the simulator as well and connecting it to one or more hybrids. The HCCStar testbench is modular enough that it can support running standalone, hybrid-level, and module-level simulations all mostly using the same codebase⁴⁵.

5.2.1 Hybrid-Level Simulations

In the hybrid-level verification, the simulation can be run with anywhere from one to eleven ABCStars attached to the HCC. Much of the hybrid-level testbench structure is shared with the stan-

⁴⁵The simulations discussed in this section (and the SEE versions described in Section 5.3) were primarily developed by Jeff Dandoy (Penn), and were initially built in parallel with the standalone framework during the verification of HCC v0.

dalone testbench. Just as in the standalone simulation, Python monitors are installed on the LCB and PRLP lines to allow the testbench to monitor communication between the HCC and ABC. Since the Python ABC model is no longer present, a new ABC packet monitor was written to eavesdrop on communication going the other way, from the ABCS to the HCC input channels. The HCC output packet monitor is still used to collect and parse packets produced from the HCC, but the handling and checking of physics packets is more complex.

Unlike the standalone verification, clusters are not randomly generated. Instead, when the hybrid-level tests wish to send a readout request to the full hybrid, the Python code first needs to load hits into the ABCStars themselves. For each event, the testbench will determine how many clusters to place on each ABC, and then use a Python implementation of the clustering algorithm to produce 256-bit unclustered hit data for every ABC. The hit data is then loaded and allowed to pass through the ABC pipeline before the testbench issues a L0 accept (and readout request). The testbench also contains a bunch crossing counter, and careful synchronization is required to ensure the testbench's BCID will match the BCIDs seen on the ASICs when tagging an event, especially when issuing bunch counter resets. The observed clusters are then compared directly against the hit data that was loaded onto the ABCS, again using the Python implementation of the clustering algorithm.

The hybrid-level tests serve two main purposes. First, they allow communications between the HCC and ABC to be rigorously tested before fabricating either chip. Each hybrid-level test begins by first resetting and configuring the HCCStar, and then proceeds to reset and configure the connected ABCStars. This requires setting various phases and delays on the HCCStar's outputs so that the ABC can see and lock to the serial protocols. In general, the Python monitors are more forgiving of the output phase alignment than the real ABCs, so it is important to ensure that it is possible for the two chips to communicate. Tests will send register writes to the ABCStars, to configure the hybrid for data-taking, and then issue register reads to ensure that both LCB communication from the HCC to the ABC and packet transmission from the ABC to the HCC are working as expected. Running these simulations during the course of the development process were helpful in uncovering real communications issues between the chips. For instance, this process exposed real differences in the ways that the PRLP protocol had been implemented between the HCC and ABC that could be caught and identified before producing the physical ASICs.

The other main purpose of the hybrid-level simulations is to test data flow through the hybrid under realistic operating conditions. As was done in the initial control path verification, in the hybrid-level simulations L0 triggers and readout requests are sent randomly at realistic HL-LHC

ATLAS rates. In single-level trigger mode, L0 accepts are sent at a rate of 1 MHz, and at a rate of 4 MHz in multi-level trigger mode. R3s and L1s are then sent at rates of 600 KHz and 400 KHz in multi-level mode as well. The number of attached ABCs in these simulations is a configurable parameter, which defaults to 11 but can be set to test any particular module. During these runs, the hybrid-level simulations use the approach described above to place hits on the attached ABCs. The number of clusters per event is configurable, but by default is drawn from a Poisson distribution with $\lambda = 18.9$. This is the highest expected average number of clusters in the strips detector: specifically, it corresponds to the expected occupancy for the Ring 0 end-cap modules (see Table 4.1). The testbench then randomly places clusters across each configured ABC.

A variety of different configurations are tested in the hybrid-level simulations, each of which has a dedicated test. Tests look at increasing the trigger rate(s), increasing the cluster occupancy, biasing the cluster distribution so a single ABC has most of the clusters, simulating back-to-back bursts of triggers, inserting other LCB commands like ABC register reads into the trigger stream, and so on. The full hybrid-level simulation suite contains 34 tests which are run in the nightly continuous integration system alongside the standalone verification with eleven attached ABCStars. They are run with RTL, post-synthesis, and post-PNR versions of the HCCStar design.

The amount of triggers performed in each test is also a configurable parameter. By default, the standard single-level trigger mode test sends 5000 L0 accepts, and most of the other configurations run with a similar number of triggers. We run a much longer version of the single-level trigger mode test with 75000 L0 accepts separately on a weekly basis. Running for a long period of time, with random transmission of triggers, allows us to potentially discover very rare failure modes or edge cases in the packet builder’s handling of physics data that would not have been found simply from running the standalone functional tests. Once a rare problem is identified, and the cause understood, a standalone test could be written to try and reproduce the problem and also make sure that similar issues were not reintroduced later in the development process. This approach proved very valuable in verifying the prototype HCCStar.

The realistic simulations can also be used to do performance studies of the hybrid using the real chips. These performance studies can be used to evaluate the performance of the hybrid and confirm that it meets the requirements set for the upgraded strips detector. The testbench can store information like the occupancies of various buffers and FIFOs on the chips, as well as the time it takes for readout requests to be fully acted upon. Because the verification is done with Python, this data can be immediately analyzed and plotted with tools like numpy and matplotlib, and also stored for additional offline analysis [109] [110]. Figure 5.1 shows some example performance

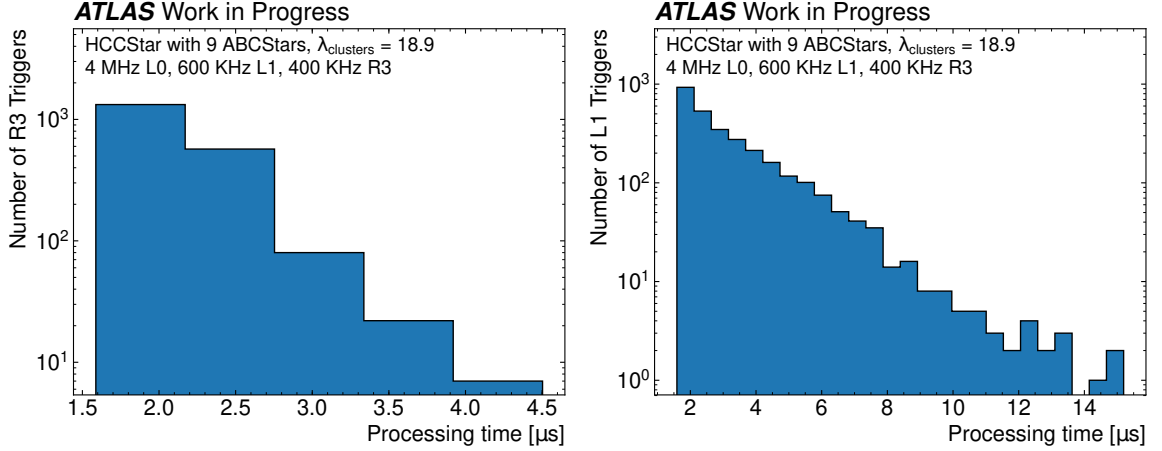


Figure 5.1: Example performance plots produced from the hybrid simulation testbench during the verification of the prototype HCCStar v0. These plots show a realistic simulation with nine ABCStars on the hybrid running in multi-level trigger mode with expected trigger rates. Each plot shows the amount of time that it took for the HCC to produce a physics packet in response to the transmission of a R3 (left) or L1 (right) trigger. As can be seen from the plot, it took much less average time for the HCC to respond to the higher-priority R3 triggers than the lower-priority L1s. Note that the exact performance has changed since these plots were made, due to changes in the prioritization mechanism for HCCStar v1.

plots from the verification of the prototype HCCStar, looking at the time it took the hybrid to produce physics packets in response to R3 and L1 readout requests in multi-level trigger mode. These simulations were also used to assess the effect on performance from changes made between HCC v0 and v1, such as the partial removal of the PR path or the change to the input channel packet buffer.

5.2.2 Module-Level Simulations

The hybrid-level simulations test a full hybrid; that is, they run with both the HCCStar and ABCStar designs loaded into the simulator at the same time. For the development of HCCStar version 1, the hybrid-level simulations were generalized into full module-level simulations. As Section 4.2.2 describes, a module contains one or two hybrids and (as part of the power board) the AMAC, the third strips ASIC. The AMAC is used to start up and reset the hybrids, and monitors them while the module is running. The module-level simulations therefore take the hybrid-level simulation environment and load the source code for the AMAC in addition to that of the HCC and ABC.

The AMAC was also developed by the Penn instrumentation group, and its verification also performed using cocotb. Thus, it was possible to load the Python libraries developed to control

the AMAC into the existing hybrid-level testbench. The AMAC verification could then be used to start up the AMAC, and read and write registers using the Endeavour protocol to configure the chip. Then, with the AMAC integrated into the testbench, simulations of an entire module can be performed.

The module-level simulations run with two hybrids, rather than just one, each containing one HCC and up to eleven ABCs. The reset sequence described in Section 4.5 can then be performed to bring the entire module online, with the hybrids being reset by sending a command to the AMAC. This verifies that the designs of the three ASICs are fully compatible with each other, and that it will be possible to turn on real modules once they have been assembled. The module-level simulations also contain a second instance of the AMAC as well, as in the real system, an AMAC will be able to reset another AMAC on a neighbouring module in the event of an emergency. This second AMAC is used to test this mechanism.

Once the system is enabled, the module-level simulations use the same algorithm from the hybrid-level simulations to test taking physics data under realistic operating conditions. In the module-level simulations, the AMAC will periodically request status information from the other chips while recording triggers. It will also occasionally reset a hybrid and ensure that the HCC and ABCs are able to recover and resume data-taking. This is done both by performing a hard reset and by putting the HCC into low-power mode.

These simulations are ran nightly using the continuous integration system, alongside both the standalone and hybrid-level simulations. Like the other tests, they are performed with RTL, post-synthesis, and post-PNR versions of the HCC.

5.3 Single Event Errors

One of the main challenges for developing front-end electronics for a particle physics detector is that not only do they need to function properly, they also need to function properly in a high radiation environment. Some of the unstable particles produced from proton-proton collisions will decay inside the electronics and deposit energy, which can interfere with their operation. This can lead to a variety of adverse effects on a chip like the HCCStar, from increased power consumption⁴⁶ to glitches or changes of state in the digital logic. This last class of phenomena, where the passage

⁴⁶An effect known as the total ionizing dose (TID) current “bump”, where the current draw can increase under irradiation [111]. The TID bump is so named because, after the initial increase, the current consumption will gradually decline back towards the initial pre-irradiation demand.

of a charged particle through the circuit causes a state change, are commonly called [single event effects \(SEE\)](#) (or sometimes “single event errors”).

Single event effects can cause major operational challenges for an ASIC. In principle, at any time, radiation could cause a 1 to toggle to a 0, or vice versa, in any part of the logic. For the HCCStar, the consequences of this could range from mildly annoying to extremely severe depending on what area is affected. Suppose a SEE caused a change of state in the configuration register defining which input channels are enabled, switching off input channel 6. All physics packets from the HCC would suddenly no longer contain any clusters from the sixth ABCStar. And because the input channel is now disabled, the packets would not contain error blocks or any indication that anything had gone wrong, aside of course from the absence of clusters from one input channel. This would also persist indefinitely, until some upstream monitoring (either automated or manual) noticed that something had gone wrong and reprogrammed the HCC.

Even more serious issues could potentially render the HCCStar unresponsive, and require a full reset and reprogramming of the entire hybrid. It is therefore important that, as part of the development of the digital logic, we attempt to understand potential places where SEEs can cause problems and harden the design accordingly. The prototype HCCStar contained some protection against single event effects in some key areas of the design. As briefly described in [Section 5.1.2](#), a handful of tests were written when verifying the prototype to prove that this protection worked. Unfortunately, when the prototype HCCStar was produced and tested in a high radiation environment, this SEE mitigation was found to be woefully inadequate. To correct this, a considerable amount of work has gone into adding additional SEE protection for HCC version 1 and into verifying its correctness.

Using the cocotb verification framework, simulations have been written to observe the effects of SEEs across the entire design, rather than just in key areas. These simulations are focused on both ensuring the additional SEE mitigation works, as well as understanding the potential consequences of SEEs in parts of the design that are not mitigated. It is hoped that running these simulations before submission of HCCStar version 1 will substantially reduce the risk that this version of the chip will also fail to function properly in a high radiation environment.

5.3.1 Upsets and Transients

Radiation can cause several different types of single event effects in digital logic [\[112\]](#) [\[113\]](#) [\[114\]](#). When charged particles pass through the transistors that make up the circuit, they can deposit energy into the semiconductor. Depending on exactly where the particle passes, how much energy is

deposited, and the type of transistor, the exact effects on the circuit can vary. This can range from a correctable state change or bitflip to very rare, destructive effects in some types of transistors. For instance, it is possible that the passage of an ionized particle through a high-power transistor can cause either a single-event burnout or single event gate rupture. This can overheat and even destroy the transistor in question [115].

This type of unrecoverable SEE is quite rare, and can primarily be addressed by the appropriate choice of radiation-hardened, lower-power transistors. In digital design, we are primarily concerned not with these rare effects but rather the two most common types of SEEs, [single event upsets \(SEUs\)](#) and [single event transients \(SETs\)](#). These SEEs can both cause “bitflips”, where a state toggles from 0 to 1 or from 1 to 0 in different parts of a chip. Unlike the more serious types of single event effect, SEUs and SETs are recoverable and do not permanently damage the circuits in which they occur. When designing a chip that will operate in a high radiation environment, the possibility of SEUs and SETs in the logic occurring must be taken into account.

A single event upset occurs when a [flip-flop \(FF\)](#) changes state [113] [114]. Flip-flops are a type of memory cell, constructed from logic gates, and used to store a single bit of information. In an ASIC, flip-flops can be used anywhere that data needs to be stored persistently, such as in the configuration registers or in the FIFO memory buffers across the chip⁴⁷. An SEU will cause a flip-flop to invert, so if it currently is storing a 0, it will contain a 1 after the SEU. This type of error is persistent, but correctable, meaning that the FF will stay flipped until it is written to again by the circuit.

Unlike a SEU, a single event transient is temporary, as implied by the name. A SET occurs when charge deposited by an ionizing particle causes a voltage pulse on a wire⁴⁸. The voltage pulse will propagate throughout the circuit, inverting the digital signal being transmitted along the wire. This inversion will only last for a very short amount of time (generally less than 1 ns), as once the pulse subsides the wire will return to normal [112]. Despite this, a SET can cause operational problems for a circuit if this glitch occurs at the right time. For instance, a SET could result in an incorrect value being latched into a flip-flop, effectively turning a transient error into a persistent SEU.

⁴⁷In the Verilog language, flip-flops are normally generated from the `reg` data type, and so “flip-flop” and “register” may be used partially interchangeably.

⁴⁸In the Verilog language, wires are generated from the appropriately named `wire` data type.

5.3.2 Proton Irradiation of the Prototype

Section 4.6 described the development process of an ASIC like the HCCStar. One key step is to test the physical chips produce from the fabrication process, to ensure that the ASICs actually function in the real world and not just in simulation. For an ASIC like the HCCStar, an important part of the physical testing involves testing the chip under high radiation conditions like those that will be present inside the ATLAS detector. These irradiations can be used to check the HCC’s sensitivity to single event effects.

During the physical testing of the prototype HCC, three irradiation campaigns were performed. The HCC underwent a photon/gamma ray irradiation, using a cobalt-60 source at Brookhaven National Laboratory that produced 1 MeV gamma rays. The purpose of this irradiation was not to test for single event effects but rather to measure power consumption and total ionizing dose effects over a period of several months. The other two irradiation campaigns involved bombarding the HCC with higher energy particles to try and cause logic errors. The chip was irradiated twice with 480 MeV protons at the TRIUMF Proton Irradiation Facility [116] [117], and once with a range of heavier ions at the University of Louvain’s Heavy Ion Facility [118]. The ABCStar and AMAC ASICs were irradiated as well during these proton and heavy ion campaigns.

These irradiations had two primary goals. First, we wanted to determine whether the HCC would function properly when running inside the high-energy proton beam. Second, if possible, the goal would be to try and measure the rate of single event upsets occurring in the chip. This rate could then be used to determine the expected rate of SEUs⁴⁹ per HCC when installed in ATLAS, by adjusting for the difference between the flux at TRIUMF with the expected flux during HL-LHC running. This could be used to estimate how often the chips might need to be reset, or how often data errors might occur.

The SEU error rate could be measured by continuously reading the HCC’s register block. The prototype’s configuration registers were triplicated, meaning that each bit of every configuration register is repeated three times. The true value of the register is determined through a best-out-of-three majority vote. Therefore, a single SEU that only effects one of the three copies can be detected and corrected internally, while a double SEU (that effects two of the three copies at the same time) can be detected but not corrected. When an error is detected, a status bit is set in a read-only SEU status register. Then, when the registers are read out, the SEU status register can be checked to see

⁴⁹This analysis was entirely focused on single event upsets. During the development of the prototype, we were primarily concerned with understanding and mitigating SEUs, not SETs. Only once the prototype failed to work properly did we become concerned with adding hardening against SETs as well.

how many SEUs were detected internally and the observed values of the read-write configuration registers can be compared against the programmed values to determine how many uncorrected SEUs occurred. The SEU status register is cleared after each read by issuing a fast command.

In addition, HCCStar also contains a second read-only SEU status register, used to store the result of physics packet parity metadata checking from the input channels (described in Section 4.4.2). The input channel is capable of detecting single and double SEUs in the physics packet metadata, which is Hamming encoded [104]. If either of these errors occur, status bits are set for each input channel and then latched into the other SEU status register. It was intended that this information could also be used to help compute the HCCStar’s SEU rate.

An early version of the ITk strips data acquisition system, known as ITSDAQ, was used to control the HCCStar during the TRIUMF irradiations ⁵⁰. During the irradiation, an HCC was mounted on a single-chip card in the path of the proton beam, and connected to a FPGA sitting in a shielded part of the room. That FPGA was then connected via ethernet to a laptop computer in the control room, where the ITSDAQ software was running. A set of ROOT [75] scripts were developed to control the HCCStar from this computer. These scripts continuously read the register block in order to measure the rate of SEUs. If uncorrected SEUs were detected after a read, the register block would be rewritten to leave the chip in a clean state for the next cycle.

During the first TRIUMF irradiation, functional tests were also performed after each read of the registers. Tests attempted to both send readout requests through the control path and data packets through the input channels to test physics data flow. The parity metadata status could then be polled after performing this test to assess the rate of SEUs in the input channel memories. If any operational problems were found, either in these tests or when reading registers, the HCC would be reset by the DAQ and reconfigured. Unfortunately, many operational problems were found, and these physics tests had a very high failure rate when the HCC was running in the proton beam. Physics packets would either fail to be seen by the output monitor, or arrive with corrupted metadata or missing clusters. Polling the SEU status register reported a high rate of uncorrected and corrected parity errors that was far larger than the rate of errors reported from the register block, suggesting that this parity mechanism was not working as intended.

In addition, some failure modes were encountered when attempting to read the registers themselves. Sometimes an “empty” HCC register read packet would arrive with a valid register address

⁵⁰ITSDAQ was developed internally by the ATLAS collaboration, primarily by Bruce Gallop and Peter Phillips from Rutherford Appleton Laboratory, and based on the existing DAQ system for the current ATLAS strips tracker. Support for the HCCStar was primarily contributed by Jeff Dandoy (Penn) to support other physical testing of the prototype chip; the irradiation scripts were built off of this work.

but no content. Sometimes register read packets would arrive with mismatched addresses: a packet might claim to contain the value of one register, but on manual inspection the content appeared to match the programmed value of another. Sometimes a single register read packet for one register would fail to arrive, and need to be repeated. And sometimes no register read packets would arrive at all, and the chip would cease to respond to commands altogether. A bewilderingly wide range of such issues were encountered during the first TRIUMF irradiation, requiring updates to the ITSDAQ scripts to be made live as new problems were discovered.

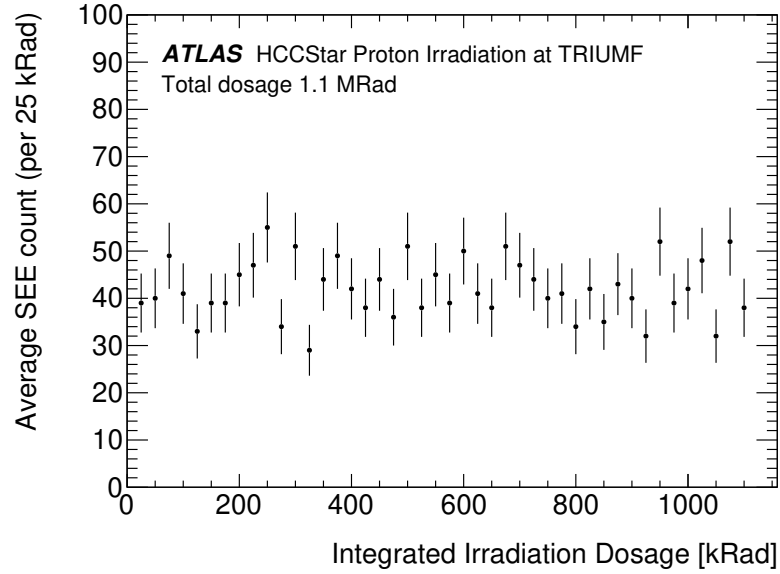


Figure 5.2: SEU rate measurement performed for the prototype HCCStar at the second TRIUMF irradiation [119]. The number of bitflips measured here is determined by continuously reading out the SEU status register, which logs whether or not any SEUs were detected in any of the configuration registers.

The problems seen in physics data flow at the first proton irradiation made it clear that the prototype would not work in a high radiation environment. It proved impossible to use this data to perform a SEU rate measurement, so the TRIUMF irradiation was repeated again two months later. In this second run, the physics mode tests were disabled and the parity metadata status bits ignored. An effort was made to harden the register read tests against the types of issues that had been encountered. This effort proved to be partially successful: while the chip still behaved badly in some instances, enough good data was collected that an SEU rate measurement could be performed. Figure 5.2 shows data from the second TRIUMF irradiation extrapolated to an expected HL-LHC rate of $1 \frac{\text{kRad}}{\text{hr}}$. With this extrapolation performed, we see that on average, the HCC's configuration

registers will see around 40 correctable SEUs each day [119].

A heavy ion irradiation was then performed at Louvain, where a SEU rate estimate was performed for varying ion sizes. At TRIUMF, all three ITk strip ASICs were irradiated simultaneously, however at Louvain only a single chip could be irradiated at a time. Additionally, due to limited data-taking time and the fact that the prototype HCC was already known not to work correctly from the TRIUMF irradiations, the heavy ion irradiation of the AMAC was prioritized instead and only limited statistics collected for the HCC. That data did appear consistent with the AMAC SEU rate, within very large uncertainties.

5.3.3 Triple Modular Redundancy

Given the problems seen when irradiating the prototype HCCStar, it has become a priority to ensure HCCStar version 1 is protected against such single event effects. The main technique that has been used to protect HCC v1 is known as [triple modular redundancy \(TMR\)](#), or triplication [120] [121]. To protect a circuit or piece of logic using TMR, three copies of the circuit are included in the design instead of just one. When laying out the circuit, the three redundant copies are placed far enough apart from each other to make it difficult for a single charged particle to cause SEEs across more than one copies. The outputs from the three circuits are combined together using best-out-of-three majority voting. An SEE that occurs on one of the three copies can be detected and corrected, assuming minimum spacing requirements are met, as the other two copies will outvote the error.

One way to apply the TMR technique would be to include three copies of every important flip-flop in the chip, so as to protect them against SEUs. As mentioned in Section 5.1.2, some key parts of the prototype HCCStar were protected in this manner, primarily the programmable configuration registers. This ensured that SEUs could not change the configuration settings of the chip. However, because not all of the flip-flops were triplicated, SEUs elsewhere could still cause operational problems, as were seen during the TRIUMF irradiations. For instance, the parity metadata problems observed at TRIUMF were determined to be a result of SEUs occurring in the control structures of the FIFO which stored some of the parity bits in HCC v0. An SEU in the read and write pointers of that FIFO would cause it to become misaligned with the packet memory. Then, when reading that FIFO, the wrong set of parity bits would be associated with a given physics packet, which would be highly likely to lead to a parity “error”. Fully triplicating all of the flip-flops would have solved this problem, as the controls of that FIFO would have been fully protected⁵¹.

⁵¹Removing the separate parity FIFO altogether— as has been done in HCC v1— also solves this problem. In HCC v1, the parity bits are stored together with the packet in the same buffer, so there is no possibility of misalignment.

However, just triplicating the flip-flops would not provide full protection against problems caused by SETs, as SETs occur in wires, not flip-flops. In fact, it may be possible for a SET to bypass the triplication entirely and affect all three copies of a flip-flop. Consider what might happen if a SET occurred on the output of the majority voter block that compares the three triplicated copies. If this occurred at the right time, it could cause the majority voter to write back an incorrect value into the triplicated copies, causing a triple SEU. There is some evidence from the TRIUMF simulations that this type of failure could occur in the prototype HCCStar, as at times a register read packet would show single bit errors in the observed value without the corresponding error detection flag having been set. An analysis of the prototype HCCStar’s regblock (done using the SEE simulations discussed below) later proved that this was possible.

To fully protect a design against both SEUs and SETs, then, it is necessary to triplicate *all* the logic, including the voters themselves. This divides the chip into three logic paths, internally referred to as “A”, “B”, and “C”. Figure 5.3 shows an illustration showing a comparison between partial TMR and full TMR. As can be seen from this drawing, in full TMR, whenever the three logic paths cross, the voting logic can be quite complex. Three voters are needed, each of which receives inputs from the A, B, and C logic paths. All three of these voters then vote, and apply the voted value to each logic path independently. If the configuration registers in the HCC were fully triplicated using this technique, a single SET in any of the voters would be unable to corrupt all three copies. However, fully triplicating the chip is quite expensive, as three copies of the logic occupy considerably more space on the ASIC, and use more power, than just one copy.

It is therefore tempting to ask if this full triplication is necessary. There are other schemes to protect against single bit errors which require much less redundancy. One group of encodings commonly used for this purpose are Hamming codes, which are used in the input channel to protect physics packet metadata [104]. Unlike TMR, which cannot distinguish between a single- and double-bit error, Hamming encoding can separately detect (but not correct) double-bit errors. In fact, in HCC v0, Hamming encoding was used in several other places, such as to protect the state machine registers in blocks like the packet builder. If a double-bit error occurred in the packet builder state machine, the packet builder would be automatically reset.

Unfortunately, the Hamming encoding of a register only provides protection against SEUs, not SETs. A SET occurring immediately before encoding or immediately after decoding the state register would still be able to cause corruption that might not be detectable. Therefore, for HCCStar version 1, the decision was made to attempt to fully triplicate as much of the logic as possible, and replace Hamming encoding with full TMR. With the exception of the input channel packet memories, the

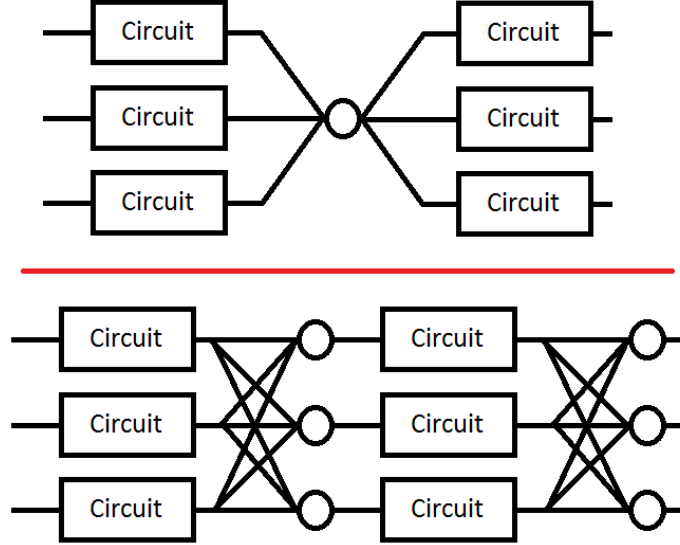


Figure 5.3: Illustration showing examples of the triple modular redundancy technique applied to digital logic, taken from Wikipedia [122]. In the above drawing, a chip has only been partially triplicated. A single majority voter, represented by the circle, connects two triplicated circuits together. Unfortunately, this means that SEEs on the majority voter could corrupt all three copies, bypassing the protection. This can be fixed by fully triplicating the voting logic as well as everything else, as is shown in the bottom diagram. In the bottom diagram, when the logic paths synchronize, each one contains its own voter. The outputs of those voters are sent back to each copy of the circuit, where a second set of voters (not drawn) is used to apply the voted value.

readout request buffer in the control path, and the HCC register read and HPR packet buffers, the rest of HCCStar version 1 has now been fully triplicated. This triplication was performed by using a tool developed by CERN’s microelectronics group called `tmrg`, which runs over the Verilog RTL source files and produces triplicated versions [121]. This triplicated RTL is then fed into the synthesis and place-and-route processes as normal, to produce a fully triplicated chip. The verification suite was extended to support running both non-TMR and TMR RTL simulations (before and after running `tmrg`) to help validate the automatic triplication⁵².

5.4 SEE Simulations

During the development of the prototype HCCStar, a limited amount of work went into understanding the potential consequences of single event effects across the chip. As noted in Section 5.1.2, some tests were developed to validate the logic that was protected. These tests simulated single

⁵²In addition, the non-TMR RTL simulations were used to produce the code coverage reports discussed in Section 5.1.3.

event upsets in the triplicated or Hamming encoded flip-flops that were present, and ensured that the SEUs were properly detected and corrected. But no effort was made to simulate the effect of SEUs in other parts of the chip, or to do any SET simulation. This was primarily due to schedule constraints when submitting the prototype HCCStar. Unfortunately, this meant that it was not possible to discover the serious SEE sensitivities that turned out to be present without fabricating the prototype and then irradiating it.

A key priority for HCCStar version 1 was to prevent this from happening again. To that end, not only has the design been mostly protected from both SEUs and SETs using triple modular redundancy, but thorough simulations of SEEs have been developed. These simulations have been implemented on top of the cocotb verification framework, and are capable of inducing SEUs or SETs in any part of the chip. The simulations have been designed for two primary purposes: first, to understand the potential effect of SEEs in untriplicated logic, and to help decide whether that untriplicated logic needs triplication. And second, to ensure that the TMR protection is working properly, and to prove that no SEE can cause problems in the triplicated logic.

To that end, two separate SEE simulation frameworks have been developed. The first, the “diagnostic” simulation framework, was used initially to understand the potential consequences of SEEs. The intention was to guide our understanding of what parts of the design were in need of triplication. However, the answer to this question turned out to be “everything”⁵³. Therefore, a second set of “rapid” SEE simulations were developed, using the same tools and infrastructure; these rapid simulations are discussed in the next section. These simulations can be run on top of the standalone simulations, as well as on top of the hybrid and module-level verification. An overview of these simulations, and a summary of results from running them over a final version of the HCC version 1 design, is presented below.

5.4.1 Diagnostic SEE Simulations

The basic idea for both the diagnostic and rapid SEE simulations is to inject both SEUs and SETs across the entire chip. SEEs are injected onto a gate-level (post-synthesis or post-PNR) HCCStar, in order to be as close as possible to the physical chip. Flip-flops are created and instantiated by the synthesis compiler, so it is only at the gate level that it is possible to fully distinguish objects sensitive to SEUs from those sensitive to SETs. Further, running SEE simulations at the gate-level

⁵³In part, this was a result from running the diagnostic simulations, but this decision was also driven by expert advice provided from the microelectronics group at CERN, who consulted on the development of HCCStar version 1. This was also made possible once decisions were taken to widen the chip and remove some of the priority/PR physics path, as described in Section 4.6.

means that we can ensure the synthesis and place and route steps did not strip out any of the triplication put in to protect against SEEs.

A full list of all flip-flops in the design, also called a “registers list”, is produced by Genus, the synthesis compiler. A list of all internal wires in the design is produced by using the Python library pyverilog to parse the post-synthesis netlist [123]. For running post-PNR simulations, the list of flip-flops can be extracted from Innovus, the software used in the place and route process. The same pyverilog parser can be applied to the post-PNR netlist to get the list of wires. Once produced, these lists are then given as inputs to the SEE simulations. If requested, these lists can be filtered so only a single submodule or block is tested at a time, but by default, SEEs are performed across the entire chip. Specific wires or registers can also be manually excluded, if they are known to be problematic. For example, SEEs on the inputs to the PLL that control the phases of the output 160 MHz clocks were found to invert those clocks. A study of the PLL determined that this was an artifact of the PLL model used in the digital simulations, and not an issue in the real PLL. Therefore, these wires were excluded.

The diagnostic simulations make use of the standalone testbench, and attempt to understand the potential impact of each possible SEU or SET. To that end, they proceed sequentially through the lists of wires and registers. For each wire, and each register, the testbench runs a set of ten unit tests in which a different part of the HCCStar’s functionality is tested. During each test, a SEU or SET on the wire or register in question is performed at a random time. SETs are injected by using a cocotb feature that can force a given wire into a specific state for a given amount of time⁵⁴. For SEUs, the Verilog library containing a representation of the Global Foundries flip-flops has been modified to support SEU injection. Additionally, SEEs can cause timing violations and lead to a flip-flop entering the metastable ‘x’ state. Because this is unphysical, and because in reality the flip-flop *would* be interpreted as either a one or a zero, the library has also been modified so that in the event of a timing violation a flip-flop will be randomly assigned a value. This “randomization-on-x” technique was first employed by the microelectronics group at CERN for other projects and adapted for the HCC as well.

The ten unit tests used by the diagnostic simulation are as follows:

- LCB communications test, which sends a single fast command and checks to see if it is received by the LCB monitor.

⁵⁴Specifically, the **Force** mechanism, added in cocotb version 1.4. For technical details, please see Appendix A.

- “Toggle” test, which waits 100 ns to see if the SEE will cause either the ABC reset or AMAC SSSH outputs to toggle.
- Clock test, which checks the stability of the output 40 MHz clock over a 250 ns period.
- LP physics test, which sends a L0 accept and L1 readout request in multi-level trigger mode with all input channels connected.
- PR physics test, identical to the LP physics test but with a R3 instead of a L1.
- ABC register read test, which sends an ABC register read command to all configured Python ABCStar models.
- ABC HPR test, which has each configured Python ABCs send an ABC HPR.
- HCC register read test, which attempts to read four HCCStar registers: the read-only register 1, the efuse register 17, and the read-write registers 32 and 42.
- HCC register write test, which attempts to write 0xffff into register 47⁵⁵, and then restore it to its default value, 0.
- HCC HPR test, which manually requests a HPR.

During each test, the testbench’s automatic checking is used to identify any errors or other unexpected behavior that might have been caused by the SEE. For instance, the LCB monitor unlocking, a register read packet containing the wrong or unexpected contents, or the HCC producing an unexpected packet are all considered SEE-induced failures if seen during the testbench. The simulations keep a log of any problems that occur during each test. If any problems were reported, the HCCStar is reset before proceeding to the next test to ensure that the next SEE occurs with the system in an expected state.

This process continues until each wire or register has been tested. Performing each SEE multiple times while the HCC may be in very different states should expose a wider class of problems than we would see if each SEE was only done once. For instance, SEEs in the HPR register or HPR state machine may have no effect if they occur when the HCC is not currently attempting to generate an HPR packet. Also, ensuring only one SEE occurs during each test means that, if problems are

⁵⁵This register controls thresholds for the LCB and R3L1 error counters; if the number of errors exceeds the threshold, a bit in the HPR status register will be set accordingly. There are no other practical consequences to changing its value, which is useful for this type of testing.

seen, it is very easy to understand what SEE cause them. This is why these simulations are called *diagnostic*.

Unfortunately, a consequence of this approach is that the simulations take a very long time to run over the entire chip. Further, some of the unit tests are more complex than others. Whether an SEE occurs randomly when the HCC is receiving an L0 accept, sending a LP to the ABCs, receiving LP physics packets in the input channel, or combining them in the packet builder could be the difference between a test succeeding or failing. Therefore, fully testing the chip with this approach would require running this slow test multiple times. Finally, because SEEs are only injected on a single wire or register with fairly long times between each SEE, there is no way to understand if multiple SEEs in different parts of the design can combine in unexpected ways. These limitations led to the creation of the second set of simulations, the rapid SEE simulation framework.

5.4.2 Rapid SEE Simulations

The rapid SEE simulation framework builds on the infrastructure and code developed for the diagnostic simulations. It uses the same wire and register files as input to decide what SEUs and SETs to perform. However, whereas the diagnostic simulations attempt to tie observed problems to specific SEEs, the rapid simulations discard this in favor of injecting SEEs as fast as possible. This means that while it may harder to debug any problems that are discovered, the chances of encountering a problem are significantly higher.

In the fully triplicated parts of the HCCStar, any SEE should be corrected by majority voting within one clock cycle. Therefore, the rapid SEE simulation attempts to insert SEEs every other clock cycle, so that any SEE performed in triplicated logic should be corrected. This means that any problems seen by the testbench as a result of an SEE must either be caused by a triplication failure, or by an SEE occurring in non-triplicated logic. Parts of the HCCStar run at 40 MHz, 160 MHz, and either 320 MHz or 640 MHz (in the output path), so the exact minimum safe time between SEEs will vary. For SEUs, this is dealt by checking which clock the flip-flop in question is gated on, and waiting for a full cycle of that clock. For wires, it is harder to automatically determine the right clock to use, so to be as safe as possible a full 40 MHz cycle must pass after each SET.

The rapid simulation is designed to run on top of a “test loop”. The test loop is a Python function which repeatedly attempts to perform some test on top of the HCCStar. What this test actually does can vary, as the simulations have been designed in a modular way so that the test loop can be swapped out. This modularity means that the rapid simulations can be ran both on the standalone

verification, with just the HCCStar, as well as with hybrid- and module- level verification. In the standalone simulation, the standard test loop enables all input channels and then repeatedly sends two readout requests, an ABC register read, and ABC HPR, and a HCC register read. The test confirms that all these output packets arrive with the right contents, and that no other unexpected packets, LCB commands, readout requests were observed by the monitors. (The automatic sending of HPRs is disabled during this test, so any unexpected packet indicates something has gone wrong). In the hybrid- and module-level simulations, the test loop tags and reads out events at realistic rates in single-level trigger mode, using the same approach that was presented in Section 5.2.

SEEs are injected on top of these test loops using the following algorithm:

1. First, the SEE injector begins by launching the test loop, which runs in parallel. SEE injection then begins.
2. We start by randomly deciding whether to perform a SEU or SET. This SEU/SET decision is generated by comparing a random number (produced uniformly, between 0 and 1) with a configurable threshold. If the random value exceeds this threshold, an SET is performed. Otherwise, an SEU is performed. Depending on the choice, a wire or register is then randomly drawn from the wires and registers lists.
3. The SEU or SET is performed. SEUs are always performed on the falling edge of whatever clock the flip-flop being flipped is gated on⁵⁶. SETs can occur at any time between two falling edges of the 160 MHz system clock. Each SET lasts for a fixed length, which by default is either 1.5 ns in the output path, or 5 ns in the rest of the chip. This is longer than most real SETs, in order to maximize the chance of causing a problem while still ensuring each SET lasts for less than a full clock cycle⁵⁷.
4. After inducing the SEU or SET, we wait for a full clock cycle as described above. On the next rising edge of the appropriate clock, the SEEs should either propagate or be corrected if they are occurring in triplicated logic.
5. The testbench checks after each SEE to see if any errors were reported, either by the test loop or by any automatic checking running in the background. If one or more errors were seen, this

⁵⁶To ensure that this is acceptable, two alternate injection schemes were also tested: performing SEUs randomly, and performing SEUs at a fixed offset of 500 ps from the *rising* edge of the clock. The concern was raised that in some cases, a SEU injected on the falling edge might not be able to fully propagate throughout the design and into other logic before being corrected on the next rising edge, hence experimenting with moving the SEUs closer to the rising edge. No significant differences were observed from making either change.

⁵⁷Alternate SET widths were tested in the hybrid- and module- level versions of the SEE simulations. See Section 5.4.3 below.

SEE injection is considered a “failure” and we proceed to the recovery and reset process, which is described below. A logfile is generated with a list of SEEs injected since the last failure (or since the simulation started), the errors that were reported by the testbench, and the result of the recovery and reset process.

6. If no errors are seen, or once the recovery and reset process has completed, we resume SEE injection. The simulation runs until one of three preprogrammed conditions have been met: a certain number of SEEs were performed, a certain number of failures have been encountered, or a certain number of cycles of the test loop have occurred.

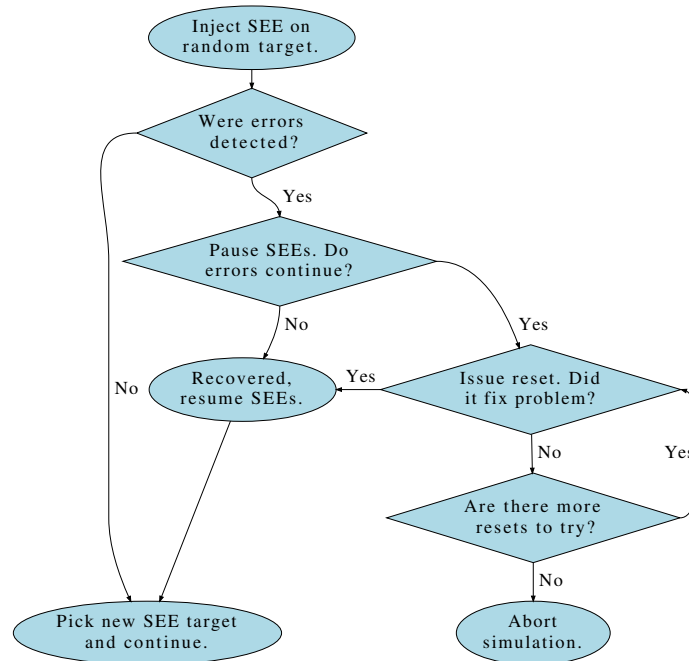


Figure 5.4: Flowchart demonstrating the “recovery and reset” process in the rapid SEE simulations, in which we try to assess the severity of any errors detected by the testbench.

The recovery and reset process tries to bring the HCCStar back into a clean state using the following algorithm. A visual representation of the recovery can also be found in Figure 5.4.

1. On the event of a failure, the simulation will halt SEE injection, but continue running the test loop in the background.
2. After each cycle of the test loop, the simulation checks to see if any additional errors have been logged by the testbench. If a cycle passes with no additional errors, then the simulation

concludes that this failure was transient and that the HCC recovered on its own. SEE injection resumes and a reset command is not issued.

3. If the HCC fails to recover on its own within a certain (configurable) number of test cycles, the simulation concludes that recovery has failed. It begins to step through a series of reset levels to attempt to fix the chip.
4. For each reset level, the reset in question is performed. The HCC then waits for a full cycle of the test loop after the reset to see if the simulation has recovered. If more errors are seen in this cycle, we conclude the reset has not been successful and proceed to the next reset level.
5. This continues until either no errors are seen after a reset, in which case a report is generated and simulation resumes, or until we come to the hard reset. If the hard reset is not successful, then the simulation is aborted, because whatever failure mode has been encountered appears to be unrecoverable.

SEE injection in the standalone simulation runs with three reset levels. First, a “soft” logic reset is attempted using the logic reset fast command. Next, if this does not work, the register reset fast command is issued, which clears all buffers and also restores all configuration registers to their default state. Performing a register reset requires reprogramming the HCCStar before running can resume. Finally, if even this does not work, a full hard reset of the HCC is performed. Performing a hard reset is even more expensive than a register reset and something that, ideally, would not be necessary very often on the real HCCStar. The hybrid- and module- level simulations perform a similar sequence, although resetting and reconfiguring a full hybrid is more expensive and more complex than just resetting the HCC.

When trying to determine if a failure has occurred, the testbench does not just look for operational issues reported by the monitors. For SEEs performed on triplicated logic, an immediate, automatic check is performed to try and assess whether the triplication is functioning properly. A SEE that occurs on triplicated logic should be corrected by a voter on the rising edge of the next clock. If this does not occur, in almost all cases, it implies that there is an error in the TMR and voting implementation in the chip. This could eventually lead to a real operational problem, if another SEE were to occur shortly afterward in one of the other two triplicated copies. Then two out of the three triplicated copies would be wrong, and the signal would be interpreted to have the wrong value.

In the rapid SEE simulations, we try to detect such issues immediately rather than wait for an operational problem to manifest. One clock cycle after performing a SEE on a triplicated object, we check to see whether that object agrees with its two triplicated copies. This can be done by string parsing: triplicated registers and most⁵⁸ triplicated wires end in “A”, “B”, or “C”. So, when performing a SEE on the “A” copy of a wire or register, we attempt to find the “B” and “C” copies and compare them. If they disagree a full clock cycle after the SEE, an error is asserted by the testbench. This bypasses the recovery-and-reset procedure and causes the HCCStar to reset immediately to clear the error. This check is primarily useful for SEUs, as SETs are not persistent by their very nature, but can be ran after both.

Running the simulations with this automatic check after SEU injection identified a number of registers with apparent TMR errors. Investigation determined that some of these appeared to be false positives. In three instances in the HCCStar (two in the input channel and one in the control path’s LP serializer), triplicated flip-flops do not get refreshed by their majority voters by design. This is because these flip-flops are only used to store data for a single clock cycle, and the rest of the time, are not active. Because they will be rewritten when they become active, a SEE that occurs at any other time does not need to be corrected⁵⁹. After carefully checking this, these false positives were manually excluded from the automatic check. However, most of the errors reported by the automatic TMR check turned out to be real issues in the RTL source code. Thanks to this check, they were fixed, removing potentially real SEU sensitivities from the final design. The rapid check continues to be run to ensure that new sensitivities are not accidentally introduced back into the source code.

Assuming there are no other bugs in the triplication itself, in principle the only failures observed in the rapid SEE simulations should be a result of injecting SEEs into the non-triplicated parts of the logic. If the simulations are ran over the entire chip, it is difficult to understand whether a problem is the result of an SEE sensitivity in the triplication or is the result of an SEE in a non-triplicated part of the logic and therefore expected. To make it possible to distinguish these two cases, we use the string parsing mentioned above to split the SEE simulations in half into TMR-only and non-TMR-only runs. In the TMR-only simulation, SEEs are only injected into the triplicated logic,

⁵⁸Wires inserted by the synthesis compiler will have autogenerated names of the form `verilog_n_1`, `verilog_n_2`, and so on, so it is not possible to use simple string parsing to see if they are triplicated. Additionally, the synthesis compiler can optimize the three logic paths independently, so the “B” instance of a wire might be missing while the “A” and “C” exist. Therefore, this string parsing is not perfectly reliable for wires.

⁵⁹At least one of these sets of flip-flops (in the input channel deserializer) can be optimized away in the place and route process, so this may only be a problem when running post-synthesis simulations.

while in the non-TMR simulations, SEEs are only injected into the non-triplicated logic⁶⁰. This way, any failures seen in the TMR-only simulation can immediately be identified as the result of a bug.

On the other hand, in the non-TMR simulations, failures that cause operational issues are expected. The main goal of these simulations is to determine whether these failures would be survivable in the real system, or whether more SEE mitigation is needed. To determine this, an observed rate of failures per SEE can be measured from the output of these simulations. By looking at the result of the recovery-and-reset process, the severity of the failure can be characterized, and rates of how often each reset will need to be invoked can be computed. Ideally, most problems will be transient and resolve themselves without the need for a reset. Then, because a full hard reset and reconfiguration is an expensive operation, it is hoped that the rate of such problems will be quite low. And finally, in principle there should be no failures which cannot be fixed by the hard reset, as these would likely require a power-cycle in reality.

5.4.3 Rapid Simulation Results

Failures due to SEEs can be quite rare. An error might only occur due to a combination of two or more SEEs within a short amount of time, or if the chip happens to be in a certain state when each SEE occurs. Therefore, it is important to run the rapid SEE simulations long enough such that each possible SEE occurs a large number of times. For the HCCStar, we agreed that we would run until the average number of SEEs per node (SEUs per flip-flop or SETs per wire) was at least 100 in both the standalone and hybrid-level simulations on the final version of the HCCStar design prior to submission. The SEE test loop for the module-level simulations was identical to that for the hybrid-level, and the module-level runs noticeably slower than the hybrid-level simulations as two instances of the hybrids are being simulated by Cadence instead of just one. Therefore, reaching this target with the module-level simulations was not seen as a priority.

Table 5.2 shows the target number of SEUs and SETs for both the triplicated and non-triplicated logic. While the relative frequency of SEUs vs SETs is a configurable parameter in the simulations, even though there are about 6 times as many wires as flip-flops it was decided to set this frequency to 50%. While this assumption is likely unphysical, we have limited intuition into the relative frequency of SEUs vs SETs in a physical chip. Performing an equal number of SEUs and SETs also allows

⁶⁰Wires generated by the synthesis compiler are considered “non-triplicated” for these purposes, even though some of them may in fact be triplicated. Since we want to guarantee that all logic in the TMR-only simulation is, in fact, triplicated, it’s safer to sort anything we are unsure about into the non-TMR simulation.

Type	Registers	SEU Goal	Wires	SET Goal	SEE Goal
Triplicated	18342	1.8M	122771	12.3M	24.6M
Non-Triplicated	23326	2.3M	126220	12.6M	25.2M

Table 5.2: Table summarizing the numbers of registers (flip-flops) and wires targeted by SEUs and SETs, respectively, for the triplicated and non-triplicated parts of the HCCStar’s digital logic. The goal for the rapid SEE simulations is to perform each SEE at least 100 times while performing an equal number of SEUs and SETs. Because there are more wires than flip-flops, the total number of SEEs needed is twice the target number of SETs.

Configuration	SEEs / Day	Instances	Total SEEs / Day
Triplicated HCC-Only	6.0M	1	6.0M
Non-Triplicated HCC-Only	2.5M	2	5.0M
Triplicated Hybrid-Level	4.0M	1	4.0M
Non-Triplicated Hybrid-Level	1.0M	6	6.0M

Table 5.3: Summary of the SEE injection performance for the standalone (HCC-only) and hybrid-level setups, showing the approximate number of SEEs that can be performed in a 24 hour run of the rapid simulations. Because the non-triplicated configurations regularly generate error conditions that require the HCCStar to be reset, they run slower than the standalone simulations and were parallelized.

us to get a better sense of the effects of the combination of the two phenomena by maximizing the chances of a SEU and SET occurring back-to-back. The cost is that, because there are significantly more flip-flops than wires, we will need to perform an average of around 600 SEUs per flip-flop in order to reach the target of 100 SETs per wire.

The nightly continuous integration system described in Section 5.1.4 was used to run these jobs. Because the non-TMR simulations regularly need to pause SEE injection to reset the HCCStar, these jobs run slower than the TMR-only simulations. Therefore, the non-TMR simulations were parallelized in order to reach equivalent performance with the TMR-only setup. Table 5.3 summarizes the performance and parallelization for both the standalone and hybrid-level configurations. These jobs were ran nightly for several months while the HCCStar design was finalized, and then for a few weeks after a final post-PNR netlist in order to reach the statistical target.

During this time, a number of issues were identified and fixed, including both real issues with the HCCStar and technical issues with the simulations. While not exhaustive, some key findings from this work are summarized below:

- As mentioned in Section 5.4.2, issues were found with the Verilog model of the PLL. Glitches on the PLL’s configuration settings generated by SETs led to odd behavior, including the permanent inversion of one or more of the internal clocks generated by the model. Additionally, issues were found with the PLL reset mechanism when running a gate-level simulation of the

HCC: SETs were capable of causing a PLL reset, and if the PLL reset was performed without a subsequent full chip reset, the chip would be left in a broken state. Analog simulations of the PLL determined these problems to be simulation artifacts.

- Serious SEE sensitivities were found in the output path serializer, which serializes the 8b10b words (or 8-bit words, in encoded mode) that make up HCCStar output packets. Any SEE on parts of the serializer would cause the output path to enter an unrecoverable state requiring a reset to fix. As there is no mechanism to just reset the serializer, this would mean issuing a register reset and reconfiguring the HCCStar. The design was fixed to remove these sensitivities.
- It was found that SETs occurring at exactly the right time could cause a hard reset of the HCCStar. Unfortunately, because a hard reset of the HCCStar also resets the ABCStar, this would mean that an entire hybrid would need to be reconfigured. The chip was updated to add deglitchers to the reset circuitry, in order to prevent the chip from being reset by short “glitch” pulses caused by a SET.
- The hybrid-level simulations run with the RTL version of the ABCStar, rather than a gate-level simulation. Under some circumstances, it was found that the ABCStar’s serial output would output a non-binary, unphysical ‘x’ state, which would then propagate through the HCCStar via the input channel and lead to major problems. To deal with this, the hybrid-level simulations immediately issue a hard reset if they detect an ‘x’ on the output of any ABCStar and reconfigure the entire hybrid. Since the ‘x’ states are unphysical, these hard resets are unlikely to be necessary in normal operations.
- Both of the above issues were ultimately determined to be exacerbated by the use of long, 5 ns SETs. Because most real SETs will last for less than 1 ns, the deglitcher functionality in the HCCStar was only capable of handling SET widths up to about 2 ns. The long SETs, therefore, lead to issues in the hybrid-level simulations that would not occur in the real system. As these issues primarily involve communication with a real ABCStar, it was agreed to switch to 1 ns SETs for the hybrid- and module-level simulations in order to get a more realistic error rate estimate.

The rapid SEE simulations were then ran over what was believed to be the final post-PNR netlist in order to achieve the required 100 SEE statistical goal. As shown in Tables 5.2 and 5.3, this required

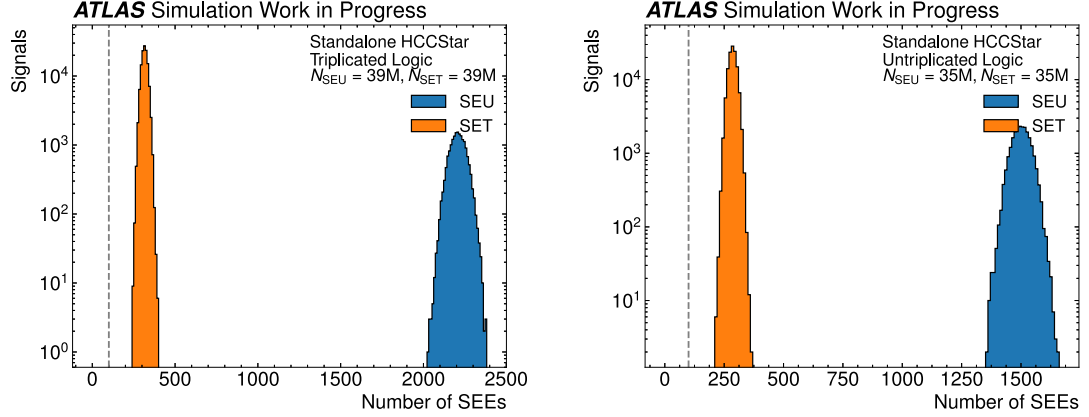


Figure 5.5: Plots showing the average numbers of SEUs per flip-flop and SETs per wire in the triplicated (left) and non-triplicated (right) standalone SEE simulations over a two-week period. The dashed grey line represents the target of 100 SEEs per node; as can be seen in the plots, this goal was exceeded for both SEUs and SETs.

performing at least 25 million SEEs total, which took just under a week in all four configurations. The simulations were allowed to run for longer while other final checks were performed on the HCCStar prior to submission in order to gather as much data as possible. Figure 5.5 shows the total statistics gathered in two weeks of running the standalone simulations. In this time, an average of approximately 300 SETs per wire and 1800 SEUs per flip-flop performed, significantly exceeding the target goal. In the triplicated simulation, two errors were observed in this two week period: in one instance, a spurious HPR packet was emitted by the HCCStar, while in the other case, physics data was missing from one input channel for one event. A slightly higher rate of errors was seen in the hybrid-level simulations during the same period of time, with 13 instances of temporary data corruption in an output physics packet and two instances in which the automatic triplication check reported an error in a flip-flop.

In principle, performing SEEs in the triplicated logic should not be able to cause any errors at all, if every SEE is immediately corrected before the next one is injected. In practice, this cannot be guaranteed for all logic, especially SETs, which might propagate through multiple blocks before being corrected. Further, the rate of these errors is extremely low, and the errors themselves not that severe. In all cases, the HCCStar was able to recover on its own without any need for a reset. The rate of these problems is significantly lower than the rate of data errors seen in the non-triplicated simulations, described below. In the standalone simulation, with around 300 SETs per wire (and 1800 SEUs per flip-flop) performed and only 2 errors, the error rate is below the statistical target of 100 SEEs per node set at the run. While this is not true in the hybrid-level setup, these simulations

Recovery Mode	Errors	Rate (%)	Days to Error	Total Errors/Day	Events to Error
Clean Recovery	148846	0.21	0.47	53k	1.63M
Logic Reset	3173	0.0045	22	1.1k	78.5M
Register Reset	7638	0.011	9	2.7k	32.0M
Hard Reset	6	8.6×10^{-6}	11667	2.1	41.1B

Table 5.4: Summary of the error rates seen during the non-triplicated standalone (HCC-only) simulations. The first column shows the total number of errors detected during a two-week period, while the second column shows this as a percentage relative to the total number of SEEs injected (70 million). The remaining columns attempt to project how often errors of each type would occur in the real ATLAS detector: first for a single HCCStar, and then for the approximately 25000 HCCStars across all of ITk. Finally, the last column estimates, assuming a 1 MHz readout rate, the number of events it would take for a given error to be encountered.

Recovery Mode	Errors	Rate (%)	Days to Error	Total Errors/Day	Events to Error
Clean Recovery	77291	0.31	0.22	115k	0.75M
BCR Recovery	3013	0.012	8	3k	28.8M
Logic Reset	2340	0.0094	11	2.3k	37.6M
Register Reset	1514	0.0061	16	1.5k	57.6M
Hard Reset	3	1.2×10^{-5}	8000	3	28.8B
Non-Binary Pin	2	8.0×10^{-6}	12500	2	43.2B

Table 5.5: Summary of the error rates seen during the non-triplicated hybrid-level simulations. See Table 5.4 for explanations of the columns. Note that here, the error rates were computed from 25 million SEEs (exactly enough to achieve the target statistical goal). In the hybrid-level simulations, the frequency of clean recoveries was discovered to be underestimated due to the very high SEE injection rate, and so a corrective factor of 1.5 was applied to that rate calculation. The “BCR” recovery row summarizes the number of errors which were fixed by a bunch counter reset; the standalone simulation does not perform these as a separate step. The last row lists the number of “non-binary” (‘x’) states on the ABCStar output observed during the run, in which a hard reset was automatically performed. These are not expected to occur in reality but are listed for completeness.

attempt to model realistic worst-case HL-LHC operating conditions, and in these conditions we do expect a very small rate of data errors even without SEE injection. Additionally, the low rates of errors in both setups and the high rate of SEEs suggests that it may require a combination of specific multiple SEEs within a very short period of time to cause a problem: something that cannot be protected against⁶¹. This was explicitly proven to be the case for at least one of the two errors in the standalone setup: two SEEs occurring within 100 ns in one input channel while in the process of reading out a packet to the packet builder were capable of causing data loss, despite the presence of triplication. Given these facts, these error rates were determined to be acceptable for HCC version 1.

⁶¹At least, not without quintuplicating all the logic instead of triplicating it. Fivefold redundancy was used for a few key registers in the prototype HCC, but— as explained in Section 5.3.3— quintuplication of all the logic would be necessary to provide full protection, which would of course be impossible given the space constraints.

Recovery Mode	Errors	Rate (%)	Days to Error	Total Errors/Day	Events to Error
Clean Recovery	105064	0.31	0.22	115k	0.75M
BCR Recovery	2163	0.006	16	1.5k	57.6M
Logic Reset	5286	0.015	7	3.7k	23.4M
Register Reset	2086	0.0061	16	1.5k	57.6M
Hard Reset	3	8.8×10^{-6}	11000	2	28.8B
Non-Binary Pin	1	2.9×10^{-6}	34000	1	86.4B

Table 5.6: Summary of the error rates seen during the non-triplicated hybrid-level simulations. See Table 5.4 and 5.5 for explanations of the columns; note that here, the error rates were computed from 34 million SEEs. Good agreement is seen between the module- and hybrid-level error rates, except for BCIDs and logic resets. Technical complexities in simulating two hybrids at once were found to explain this difference.

For the non-triplicated simulations, error rates were computed for both the standalone and hybrid-level configurations and are summarized in Tables 5.4 and Table 5.5. Once sufficient statistics were acquired from the hybrid-level simulation, the module-level simulation was then run; its results are summarized in Table 5.6. As seen in these tables, while transient issues and data errors are quite common, it is relatively rare to encounter an issue that needs a reset in both the standalone and hybrid-level setups. An attempt was then made to extrapolate these error rates to HL-LHC running conditions using data taken from the proton irradiation of the prototypes. As described in Section 5.3.2, we expect that the configuration registers of each HCC will see around 40 SEUs per day of data-taking. By comparing the size of the register block to the total number of untriplicated flip-flops, we find that we expect around 500 SEUs in the untriplicated parts of the logic in each HCC every day. While we do not have good measurements of how often SETs will occur in the logic, since we perform an equal number of SEUs and SETs in the simulations, we assume we will see approximately the same number of SETs as SEUs, for a total of 1000 SEEs per day per chip in the non-triplicated parts of the detector.

With this assumption made, the number of errors per day are computed in Tables 5.4, 5.5, and 5.6. This calculation is done both for an individual HCCStar and for the ITk strip detector as a whole, which contains approximately 25,000 HCCs total. The hybrid-level and module-level simulations are much more realistic than the standalone, but numbers from the standalone are presented as well for comparison purposes. In all three setups, we see that it might take on the order of 10,000 days of running— or somewhere between 20 and 30 years— for a single HCCStar to see a hard reset. While this means two or three hybrids might need a hard reset every day, it also means that there will be some modules which may never need a reset during the entire lifetime of

the HL-LHC⁶². An analysis was performed to understand the source of this handful of hard resets, and they were found to be caused by SEEs placed on the input 160 MHz clock. It appears that the hard resets are a result of instability in the PLL model, which as described above, is known to be a simulation artifact. That suggests that in reality, we may never see hard resets at all during the HL-LHC in any module.

The rates of softer resets and data errors are larger, but still quite survivable during data-taking. Because these rates were generally found to be acceptable, and due to limited time and resources, detailed studies to understand the sources of all observed errors were not performed. Fixing or studying issues requiring a hard reset was prioritized instead; note that issues with the PLL model that cause hard resets may also lead to logic or register resets in some cases, but this was not conclusively proven. That said, some sources of register resets (such as the output path serializer, mentioned above) were identified and fixed prior to submission. And it is known that, because the input and output pads of the HCCStar are not fully triplicated—something that is impossible given the space constraints—it is not possible to fully protect against all logic errors despite triplication elsewhere. For instance, SEEs on the serial line connecting an ABCStar to an HCC input channel can cause multiple physics packets to be lost, which will eventually trigger the HCC’s flow control mechanism, as it thinks there are outstanding readout requests. This will cause the HCC to halt transmission of readout requests until a logic reset can be issued. Therefore, some rate of resets is to be expected.

Of course, no matter how thorough these simulations may be, it is impossible to know whether or not version 1 of the HCCStar will work without fabricating and irradiating it. Even if we had the space to fully triplicate all the logic, and even if we had managed to fix every problem found by the SEE simulations, it is possible that proton irradiations would discover more unexpected features. Still, we feel that this work has helped significantly reduce the risk of that happening, and helped build our confidence that HCCStar V1 will work as intended.

⁶²This is especially true given that lower occupancy modules, that are further from the interaction point, will see less activity and fewer SEEs than modules closest to the interaction point.

CHAPTER 6

VBF+MET: Higgs to Invisible Overview

The last two chapters focused on upgrades to be installed in the ATLAS detector for high-luminosity operations in the coming years. The rest of this thesis will present an analysis that was performed using 139 fb^{-1} of proton-proton collision data recorded at the current ATLAS detector (described back in Chapter 3) with a center-of-mass energy of $\sqrt{s} = 13\text{ TeV}$. This analysis is a search for new physics beyond the Standard Model: specifically, a search for invisible decays of a Higgs boson produced via vector boson fusion. As explained in Chapter 2, “vector boson” refers to a W or Z weak boson. In vector boson fusion, a pair of W or Z bosons produced in a proton-proton collision interact with each other and “fuse” into a single particle, such as another vector boson or— in this case— the scalar Higgs boson⁶³.

Once the Higgs is produced via the VBF process, it then decays invisibly. In this context (as explained in Chapter 3), “invisible” means that the Higgs decays into a pair of electrically neutral particles that do not decay in the ATLAS detector. Since they are neutral, they will not leave tracks in the tracker, and since they do not decay, they will not interact and deposit energy into the calorimeters. Therefore, their presence can only be inferred by a momentum imbalance in the transverse plane, known as the missing transverse momentum or missing transverse energy (MET) and labelled $E_{\text{T}}^{\text{miss}}$. Since the analysis involves looking for events in the detector with missing transverse momentum plus evidence of vector boson fusion of the Higgs, it is often referred to internally by ATLAS as “VBF+MET” or “VBF Higgs to Invisible”.

This chapter gives a quick introduction and overview of the VBF+MET analysis performed with the full run 2 dataset (an integrated luminosity of 139 fb^{-1}). Section 6.1 describes the motivation

⁶³More details about the Higgs and electroweak physics relevant to this analysis can be found back in Chapter 2, although some information is repeated again here as convenient when explaining the motivation for this analysis.

for performing this search in the VBF channel, as well as the history of previous recent results from both the ATLAS and CMS collaborations. A summary of the current analysis strategy and the results presented in this thesis is then given in Section 6.2. Section 6.3 then describes how we attempt to identify Higgs to invisible events in the data recorded by the ATLAS detector, and what variables and quantities are used to select these events. Section 6.4 then presents the experimental uncertainties on this selection that must be considered when interpreting the results. And finally, Section 6.5 presents a simulation of the invisible Higgs process that was used to develop this selection, and shows the predicted efficiency of applying it to simulated data.

6.1 Introduction to VBF+MET

In the Standard Model, the only “invisible” particle is the neutrino, so for an invisible Higgs decay to occur it must involve an all-neutrino final state. The only way for this to occur in the Standard Model is for the Higgs to first decay to a pair of Z bosons, which then each decay to two neutrinos. Because the Higgs (at 125.1 GeV [124]) is less than twice the mass of the Z (at 91.2 GeV [12]), the probability of this occurring is extremely low. One of the Z bosons must be off-shell, and so this $H \rightarrow ZZ^* \rightarrow \nu\bar{\nu}\nu\bar{\nu}$ decay has a branching ratio of approximately $\mathcal{B}_{H \rightarrow \text{inv.}} = 1.05 \times 10^{-3}$, meaning that this will occur less than 1% of the time for a given Higgs boson [124]. The LHC is not sensitive enough to detect evidence such a process, if this is really the rate at which it occurs.

However, the chances of this decay occurring can be significantly increased in various Beyond the Standard Model scenarios. One potentially interesting case, as discussed in Chapter 2, is so-called “Higgs portal” dark matter [8]. The idea runs as follows: the Higgs, as a scalar boson, can couple to almost every massive particle in the Standard Model, with the apparent exception of the neutrino, and the strength of that coupling is responsible for giving those particles their mass. It is possible that if dark matter is comprised of one or more new, currently undiscovered massive particles that these dark matter particles would *also* couple to the Higgs, and perhaps derive their mass through this coupling as well. If this is the case, and if the mass of the dark matter particle χ is less than $\frac{1}{2}m_h = 62.5$ GeV, then this decay might occur frequently enough to be observed at the LHC. A dark matter particle would by definition be “invisible”, since dark matter (like the neutrino) is electrically neutral and long-lived (or entirely stable). Therefore, searching for Higgs to invisible decays is a way to probe this idea at a collider.

The Higgs portal does not place many requirements on the nature of the dark matter particle(s) other than requiring that they couple to the Higgs. In this way, it is more generic than a targeted

search for a [WIMP](#) [38] or another specific dark matter candidate. Results from the Higgs to invisible analysis can however be interpreted as a limit on a particular dark matter model. Given a specific Lagrangian describing how a given dark matter candidate interacts with the Higgs, a limit on the branching ratio $\mathcal{B}_{H \rightarrow \text{inv.}}$ can be used to set a limit on the existence of that candidate over a varying mass range. This makes this analysis complementary with results from targeted direct and indirect detection dark matter experiments. A direct detection experiment looks at evidence of dark matter interacting directly with the Standard Model, while indirect detection tends to look for evidence of dark matter self-interaction that might produce Standard Model particles. Higgs to Invisible is instead a reverse of this: a search for a way in which Standard Model particles, in this case the Higgs, could produce dark matter. The combination of analyses and experiments using these three different perspectives on this interaction is a very powerful strategy to help understand the true nature of dark matter.

6.1.1 Vector Boson versus Gluon-Gluon Fusion

Other particles in the Standard Model can also decay to an all-invisible final state: most notably the Z boson, as mentioned above. Distinguishing the $Z \rightarrow \nu\nu$ decay from the $H \rightarrow \text{inv.}$ decay is the main challenge of any invisible Higgs search, and requires finding a way to distinguish production of the Higgs from production of the Z . Higgs production can occur through several different mechanisms, and Figure 6.1 shows Feynman diagrams of four of the most frequent: [Gluon-Gluon Fusion \(ggF\)](#), [Vector Boson Fusion \(VBF\)](#), [Top Fusion \(ttH\)](#), and [Higgs-strahlung \(VH\)](#) [31] [125]. Of the four, ggF occurs the most frequently, followed by VBF: with a center-of-mass energy of $\sqrt{s} = 13 \text{ TeV}$, ggF has a production cross section of 48.6 pb, while VBF is an order of magnitude lower at 3.78 pb [124].

It is reasonable to ask why then the analysis presented in this thesis is “VBF+MET” and not “ggF+MET”. Unfortunately, even though ggF production is much more frequent, experimental problems mean that it is much harder to identify ggF $H \rightarrow \text{inv.}$ events in the detector. For one thing, as shown in Figure 6.1c, the tree-level gluon-gluon fusion process produces an all invisible final state, which in the detector would just show up as a missing transverse momentum excess. Since it isn’t possible to distinguish E_T^{miss} from a Higgs decay from E_T^{miss} from a Z decay, this presents an immediate problem. Searches for ggF Higgs to invisible tend instead to look for the case where an extra quark is radiated out as initial state radiation, leading to a single extra jet in the final state in addition to the E_T^{miss} excess. Unfortunately, the inclusive cross section of Z production at

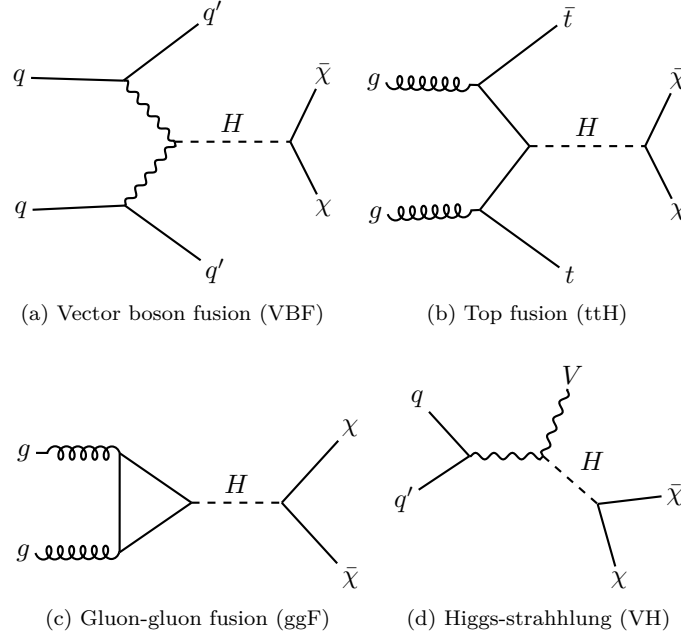


Figure 6.1: Feynman diagrams showing potential invisible Higgs decay in the four main Higgs production channels: VBF (a), ttH (b), ggF (c), and VH (d). This thesis presents a search for invisible Higgs decay in the VBF channel.

$\sqrt{s} = 13 \text{ TeV}$ is 58.43 nb [126] [127], and as the $Z \rightarrow \nu\nu$ branching ratio is 20% [12], its total cross section is 11.69 nb. This is orders of magnitude larger than the corresponding gluon-gluon fusion Higgs cross section, and requiring a single additional jet will not do much to change this. That means that the sensitivity and discovery potential of a ggF $H \rightarrow \text{inv.}$ analysis is quite low⁶⁴.

Fortunately, the VBF channel has several advantages over ggF in this regard. First, as shown in Figure 6.1a, the VBF final state is *not* all-invisible but also contains the two quarks which produced the vector bosons that fused into the Higgs. These VBF jets have a relatively distinct topology, which as described below can be used to help distinguish VBF Higgs to invisible events from Standard Model processes like $Z \rightarrow \nu\nu$. While the Z can also be produced via vector boson fusion, like the Higgs, the production cross section for a Z produced in association with two jets [129] [130] is much lower than the inclusive Z cross section and closer to that of the VBF Higgs [126]. This should mean that if the Higgs is produced via VBF and does decay invisibly, it will be easier to see in the data. Far more information about how the invisible Higgs “signal” is distinguished from the $Z \rightarrow \nu\nu$

⁶⁴That said, ATLAS and CMS do perform “jet+MET” or “mono-jet” searches that look for exactly this signature, as they have many other interpretations. In preliminary results with full run 2 data (139 fb^{-1}) from ATLAS, an observed (expected) limit was set as $\mathcal{B}_{H \rightarrow \text{inv.}} = 0.63(0.57)$ [128]. This result is already worse than the previous partial run 2 (36.1 fb^{-1}) results shown below in Section 6.1.2, illustrating the lack of sensitivity in this channel.

“background” can be found in Section 6.2 as well as in Chapter 8, but suffice it to say that despite the lower cross section, these features mean the VBF channel is the most sensitive for performing a Higgs to invisible search.

6.1.2 Previous Results

Invisible Higgs analyses have been performed at the LHC before, both in the VBF channel and in other channels. Figures 6.2 and 6.3) show a summary of results from the ATLAS [131] and CMS collaborations [132], respectively, taken with run 1 and partial run 2 data. As of yet, none of the prior analyses have seen any significant evidence of invisible Higgs decays, and therefore the plots show limits on the branching ratio $\mathcal{B}_{H \rightarrow \text{inv.}}$. The limits are presented as both an “expected” and “observed” limit; the “expected” limit is the lowest limit that could be set if the data agrees perfectly with the Standard Model, while the “observed” limit is what was actually seen. If the observed limits are within the uncertainties on the expected limit, this indicates good agreement with the null hypothesis of the Standard Model. A large difference between the two, on the other hand, could be evidence of new physics.

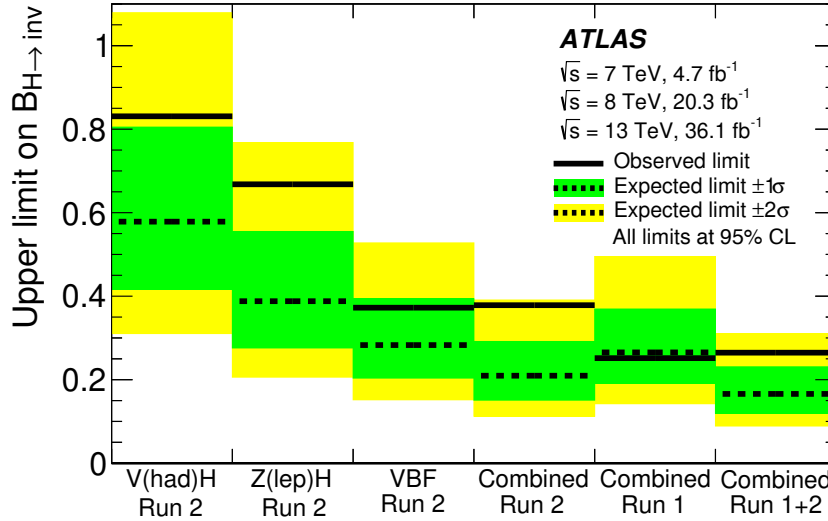


Figure 6.2: Limits on the branching ratio $\mathcal{B}_{H \rightarrow \text{inv.}}$ set by the ATLAS collaboration with run 1 ($\sqrt{s} = 8 \text{ TeV}$) and partial run 2 ($\sqrt{s} = 13 \text{ TeV}$) data [131]. The three entries on the left show limits from run 2 analyses performed with 36 fb^{-1} of data recorded in 2015 and 2016; the channels are then combined together, and then combined again with data from run 1. As expected, the VBF channel is more sensitive than the other two Higgs-strahlung channels, and is the main contributor to the combined $\mathcal{B}_{H \rightarrow \text{inv.}}$ limit.

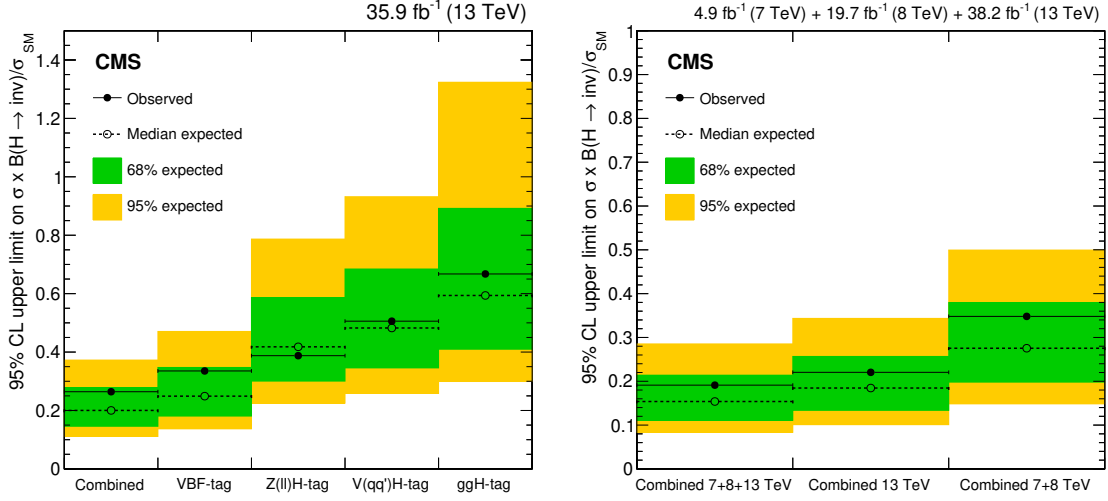


Figure 6.3: Limits on the branching ratio $\mathcal{B}_{H \rightarrow \text{inv.}}$ set by the CMS collaboration with run 1 ($\sqrt{s} = 8$ TeV) and partial run 2 ($\sqrt{s} = 13$ TeV) data [132]. The plot on the left shows analyses performed in different Higgs production channels; as with the ATLAS results shown in Figure 6.2, the VBF channel is the most sensitive. The plot on the right then shows combinations with run 1 data.

In the ATLAS collaboration, a version of the VBF+MET analysis was performed both in run 1, with $\sqrt{s} = 7$ TeV and 8 TeV data [133], and more recently in run 2, with 36.1 fb^{-1} of $\sqrt{s} = 13$ TeV data taken in 2015 and 2016⁶⁵ [135]. Figure 6.2 shows this partial run 2 result alongside two other partial run 2 results in Higgs-strahlung channels [136] [137]; as argued above, the VBF channel sets the lowest limit on $\mathcal{B}_{H \rightarrow \text{inv.}}$ out of the three. The channels can be combined together to improve the limit even further, however, and also combined with data from run 1 to set the lowest possible limit. The observed (expected) limits from the three partial run 2 results and their combination were as follows:

- V(had)H: $0.83 \text{ } (0.58^{+0.23}_{-0.16})$ [137]
- Z(lep)H: $0.67 \text{ } (0.39^{+0.17}_{-0.11})$ [136]
- VBF: $0.37 \text{ } (0.28^{+0.11}_{-0.08})$ [135]
- Combination: $0.26 \text{ } (0.17^{+0.07}_{-0.05})$ [131]

These results can be compared to similar partial run 2 results from the CMS collaboration, as shown in Figure 6.3. Unlike ATLAS, a gluon-gluon fusion limit is included in these CMS results,

⁶⁵The ATLAS group at the University of Pennsylvania has also been involved in both these prior results. Most recently, Elliot Lipeles also worked on 36.1 fb^{-1} analysis with a previous graduate student, Bill Balunas [134].

but as argued above, it is actually the least sensitive channel. Here, an observed (expected) limit of 0.33 (0.25) was seen in the VBF channel, and an observed (expected) limit of 0.19 (0.15) was seen from the full combination of both run 1 and (partial) run 2 data [132]. These limits are comparable but slightly lower than the equivalents from the ATLAS collaboration, as shown above. It's worth noting that both the ATLAS and CMS results up until now have seen slight excesses, where the observed limit is a bit larger than the expected. It is not at all uncommon to observe statistical fluctuations leading to one or two σ differences between data and prediction, so these excesses are not that interesting. Still, repeating the searches with more data will reduce the expected limits further and help to see whether these excesses— which, it must be emphasized again, are not currently significant— are early signs of new physics or merely statistical fluctuations.

The full run 2 VBF+MET analysis presented in this thesis builds on the prior 36.1 fb^{-1} version, and follows a similar analysis strategy⁶⁶. In 2017, an additional 44 fb^{-1} was collected, followed by 59 fb^{-1} in 2018 [56], increasing the total size of the dataset by a factor of 4. Since the sensitivity of the limit depends on the statistical uncertainty on the dataset, naively we might expect an improvement by a factor of $\sqrt{4} = 2$, bringing the expected limit down to 0.14. There are several challenges that must be overcome in order to realize this improvement, however, as statistical uncertainties on the dataset are only one of the effects that contribute to the expected sensitivity of the limit. One major problem is our ability to understand the $Z \rightarrow \nu\nu$ background process, which is partially reliant on Monte Carlo simulation in addition to real data. Generating $Z \rightarrow \nu\nu$ events is computationally expensive, and so Monte Carlo simulation statistics were a major limiting factor on the previous 36.1 fb^{-1} analysis, to the point that they comprised one of the dominant uncertainties on the result. Improving the limit therefore also requires figuring out a computationally efficient way to improve these Monte Carlo statistics; a detailed discussion of this problem and its solution can be found in Chapter 7.

6.2 Analysis Overview and Strategy

Searches for new physics generally involve a few common steps, although the details can vary considerably. This section presents a high-level overview of the components and steps involved in the 139 fb^{-1} , full run 2 VBF+MET analysis. Speaking very broadly, there are three things that need to be done: a signal region needs to be defined, a background estimate performed in that signal

⁶⁶Some comparisons to this previous result are made throughout the thesis, but readers interested in more detail on it could consider consulting Bill Balunas's thesis, which outlines the entirety of that version of the analysis in detail [134].

region, and a statistical framework developed to understand the results. In addition, simulations of real proton-proton collision data that model the relevant processes are needed for all three steps. And various sources of uncertainties need to be considered throughout the process and applied when using the statistical framework to interpret the results.

6.2.1 Signal Region Definition

First, many different interactions and processes can occur in a proton-proton collision. A procedure is needed to filter the ATLAS dataset in order to select only events that could conceivably have been caused by the process being searched for, which is known as the “signal”, and separate them out from the “background” noise generated by other physics processes. This is generally done by defining a selection, often known as a series of “cuts” on the data, that creates a so-called [signal region \(SR\)](#)⁶⁷. For events to enter the VBF+MET signal region, they must have a large E_T^{miss} and two jets that match the vector boson fusion topology. A visual rendering of one possible such event is seen in Figure 6.4, and a full description of the signal region definition in this analysis can be found below in Section 6.3.

6.2.2 Use of Monte Carlo Simulation

When developing the event selection, it is important to not look at real data, as this could easily lead to bias in the signal region definition. Avoiding looking at real data is known as “blinding” the signal region. While the analysis is blinded, simulations of the signal and background processes are used to develop and optimize region definitions instead of real data. With simulated datasets, analyzers are free to try out different cuts in order to maximize the number of signal events, S , accepted and minimizing the number of background events, B . The signal region is kept blind while optimizing S/B until the region definitions are frozen and all other inputs prepared, until approval to “unblind” is granted during an internal ATLAS review process.

The simulated datasets used in this process are commonly referred to as *Monte Carlo*, as they are produced by repeated random sampling of a probability distribution using Monte Carlo methods⁶⁸.

⁶⁷Another approach, that has become more common in recent years, is to use machine learning to try and define the signal region instead, by training a neural network to distinguish signal from background. This approach was explored for the full run 2 analysis, but ultimately rejected, as it produced relatively minor improvements in the expected limit (on the order of 10%) for a relatively large gain in complexity. Interested readers with access to internal ATLAS documents can find the details in the internal note for this analysis, which can be found here: <https://cds.cern.ch/record/2717301>

⁶⁸Developed by John von Neumann, Stanislaw Ulam, Nicholas Metropolis, and others at Los Alamos National Laboratory [138], and named, according to Metropolis, after the Monaco casino, as evidently Ulam had an uncle who liked to gamble there [139]. Though in fact, the method had been independently developed by Enrico Fermi several

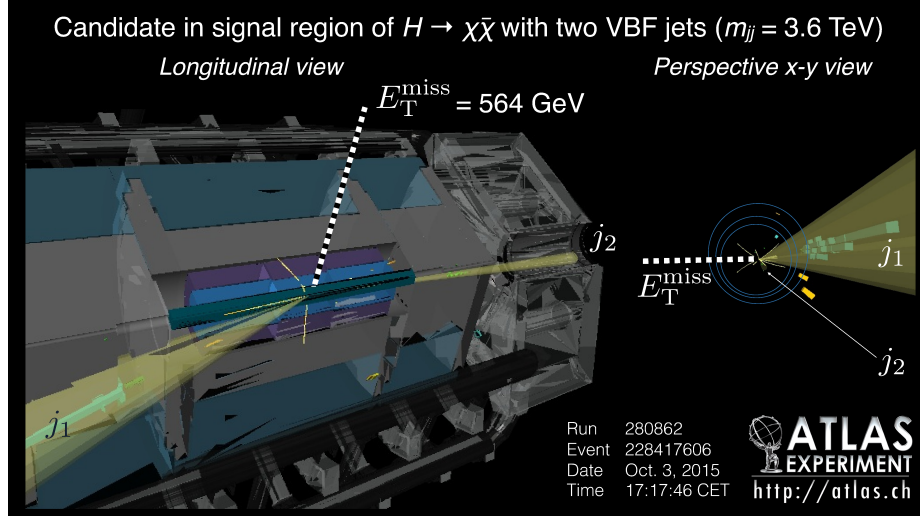


Figure 6.4: Event display showing a 3D rendering of a candidate VBF Higgs to invisible event in the ATLAS detector [135]. The event, which was recorded in 2015, consists of a dijet system with two forward jets in opposite hemispheres of the detector with a large dijet invariant mass, as well as a large missing transverse momentum that could arise from the decay of a Higgs boson. See Section 6.3 for more details on the requirements imposed to select candidate events like this one.

Dedicated software packages developed by particle theorists are employed to produce Monte Carlo datasets for high energy physics, which are then fed through a simulation of the ATLAS detector to emulate the effects of particle identification and reconstruction algorithms. These methods are quite complex, and much more information on how MC samples are generated can be found in Chapter 7. Note that Monte Carlo events are not just used to help define the signal region, but to also predict the number of events (both signal and background) expected there, in a process known as background estimation, the second of the three main steps.

6.2.3 Background Estimation

It is effectively impossible to define a signal region that will *only* select signal events, as some amount of contamination from the Standard Model background processes is more or less inevitable. This can be because a Standard Model process is indistinguishable from the signal process being searched for, as is the case with $Z \rightarrow \nu\nu$ and Higgs to invisible decays, or it can be because of measurement errors and inefficiencies when recording data in the detector. For instance, while the W boson decays into a visible charged lepton as well as a neutrino, if the lepton is not identified or “lost” in the detector, a $W \rightarrow l\nu$ event might also appear to have an all invisible final state and also be confused for Higgs

years prior [139].

to invisible. The size of the background contamination in the signal region therefore needs to be estimated, so that the analyzer can know whether or not the observed number of data events in the signal region is consistent with the Standard Model– or whether or not it indicates the discovery of new physics.

In the VBF+MET analysis, background estimation is performed using Monte Carlo simulations of the backgrounds, plus real data in so-called orthogonal [control region \(CR\)](#). A control region is generally used to “control” a particular background: they are defined to be as similar as possible to the signal region, except no signal is present (so it is safe to look at data there before unblinding the signal region). Since these control regions should contain no signal, any difference between MC prediction and data should only arise from modelling problems in the Monte Carlo. Therefore, the difference between data and simulation in these regions can be used to correct or rescale the MC prediction in the signal region, in order to make it more accurate. The background estimation techniques and control regions used are described in much more detail in [Chapter 8](#), but the two main CRs are one-lepton and two-lepton versions of the signal region. The one-lepton control region primarily selects $W \rightarrow l\nu$ events and is used to measure this “lost lepton” background, while the two-lepton control region primarily selects $Z \rightarrow ll$ events. Because the $Z \rightarrow ll$ and $Z \rightarrow \nu\nu$ processes are very similar, this two-lepton region can be used to estimate the $Z \rightarrow \nu\nu$ background. These control regions only select events with electrons or muons, as unfortunately tau leptons are much more difficult to reconstruct.

6.2.4 Fitting and Uncertainties

Once the signal region is unblinded, it is necessary to have some way of interpreting the result. A simultaneous likelihood fit is performed in the signal region and various control regions that adjusts the background prediction in order to better explain the data. The fit also takes into account the simulated model of the invisible Higgs signal process (generated using Monte Carlo as explained above), and tries to see if the signal can be used to explain any discrepancies between data and background, or if the observed data is more consistent with a background-only hypothesis. The results of the fit can then be used to set an upper limit on the invisible Higgs branching ratio at some confidence level. All values of the branching ratio that are greater than this limit are excluded to within the confidence level, which is conventionally 95%. [Chapter 9](#) explains the statistical methods used to fit the data and calculate these limits, and also presents the results.

Various sources of uncertainty need to be considered as part of this process. There are *statistical*

uncertainties, which arise due to random fluctuations in the sampled data, and *systematic* uncertainties on the methods and procedures used to collect and interpret the data. Statistical uncertainties will decrease if more data is collected, but systematic uncertainties will not. There are a variety of both theoretical and experimental systematic uncertainties on the algorithms and definitions used to identify and select events, on the simulated Monte Carlo datasets, and on the various background estimation procedures employed in this analysis. All these uncertainties are treated as “nuisance parameters” on the background prediction in the likelihood fit, and can vary simultaneously during the fit. The overall systematic uncertainty on the limit is then determined by the nuisance parameters that had the largest impact during the fit process. Discussion of various sources of uncertainty on different aspects of the analysis can be found throughout the next several chapters.

6.2.5 Analysis Timeline

Work on the above steps for the 139 fb^{-1} analysis began in 2018, immediately after the 36.1 fb^{-1} result was published. While the overall analysis strategy has not changed, the signal region definition was completely reoptimized as part of this work, which improved the expected limit considerably even before the addition of more data. Additionally, work was done to improve the background estimation, with a focus on fixing the Monte Carlo simulation statistics problem mentioned above. There have since been two versions of the full 139 fb^{-1} analysis, both of which are documented in this thesis. An initial, unpublished “preliminary” result was released in April 2020 as a physics briefing by the ATLAS collaboration⁶⁹ [140]. Following the preliminary result, several improvements to the background estimation process were made over the course of the next year, and a paper with a second “final” full run 2 result is now forthcoming.

6.3 Event Selection

The VBF+MET analysis is performed over the full run 2 dataset, with an integrated luminosity of 139 fb^{-1} [56]. Data is initially selected using the ATLAS trigger system, after which reconstruction algorithms are run in order to identify particles and objects like jets. Requirements are then imposed on the various identified objects in order to define a signal region and select VBF-like candidate events. The signal region itself is then broken down into a number of sub-regions known as bins, in order to allow for the background estimate to vary independently as a function of the binned

⁶⁹This was intended to be presented at the Rencontres de Moriond conference, which was scheduled for late March, 2020, but was cancelled due to the COVID-19 pandemic.

variable. These steps are all explained in more detail below: Section 6.3.1 summarizes the triggers used, Section 6.3.2 the object definitions, Section 6.3.3 the signal region definition, and Section 6.3.4 the binning strategy.

6.3.1 Triggers

The trigger system, initially introduced in Section 3.3, is used to select candidate events which appear to have a large amount of missing transverse momentum that could have come from the invisible decay of a Higgs boson. Both L1 and HLT E_T^{miss} triggers are used to select data [141]. The L1 trigger uses coarse-grained information from the calorimeters to approximate the energy of the event, while the HLT trigger uses calibrated, finer-grained calorimeter cell information to first find jets before computing E_T^{miss} . Only jets that appear to have $p_T > 7 \text{ GeV}$ are considered when applying the HLT E_T^{miss} trigger. The thresholds of the triggers were increased over the run 2 data-taking period as the pile-up, μ , increased, and the exact implementation also varied. In 2015, the HLT threshold was first set to 70 GeV, before increasing to 90 GeV in 2016 and then 110 GeV in 2017 and 2018. The L1 threshold was also briefly increased from 50 GeV to 55 GeV in 2017 when running with very high pileup conditions, as shown in Figure 3.3.

For the one- and two-lepton W and Z control regions, data is selected using single-lepton and di-lepton triggers instead that require an event to have one or two electrons or muons. The logical “OR” of both sets of triggers is applied in both regions in order to increase acceptance, since (for instance) an event in the Z control region might appear to only have one lepton at trigger-level but actually have two after reconstruction, and vice versa for the W . Note that because the E_T^{miss} only uses calorimeter level data, and because muons, like neutrinos and other invisible particles, escape the calorimeter, a muon is treated as invisible from the point of view of the E_T^{miss} trigger. Because of this effect, the lepton triggers are also ORed together with the E_T^{miss} trigger when selecting muon events.

6.3.2 Object and Variable Definitions

Before we can define the signal region itself, it is first necessary to define the objects that are part of the signal region definition. This section briefly summarizes the definitions used in this analysis for jets, for the missing transverse momentum, and for leptons and photons that are then used in the cuts that make up the signal region. While these definitions are based off of the standard techniques defined in Section 3.4, the exact algorithms and parameters involved can vary. For

instance, anti- k_t [87] may be the standard jet-finding algorithm, but there are alternative choices, and the size parameter R is also variable. So while the VBF signal has two jets, the exact definition of “jet” may vary⁷⁰. In addition, this section also defines some relatively non-standard variables that are used as part of the signal region definition. It can be useful to derive quantities beyond the four-momenta of objects in (p_T, η, ϕ, E) -space when trying to describe events, and some more complex variables used in the analysis are defined below.

6.3.2.1 Jets

Hadronic jets are identified and constructed using the particle flow [86] and anti- k_t [87] algorithms. The implementation of anti- k_t in the FastJet software package [142] is used, with a radius of $R = 0.4$. To be considered in this analysis, a candidate jet must have transverse momentum $p_T > 25$ GeV and pseudorapidity $|\eta| < 4.5$. Additionally, to remove contamination from pile-up, jets with lower momentum in the central region must pass the jet vertex tagging algorithm introduced in Section 3.4. Any jet with $p_T < 60$ GeV and $|\eta| < 2.5$ must also have a JVT score [90] greater than 0.20, otherwise it is rejected as pile-up. This JVT requirement gives an efficiency of around 98% for identifying hard-scatter jets. Standard calibration and cleaning algorithms are also applied in order ensure that the jets are really produced from proton-proton collisions, and not from non-collision backgrounds [143].

In vector boson fusion, two incoming quarks radiate weak bosons and then continue outward, producing (at least) two jets in the final event. Therefore, we are often concerned with the “dijet system”: the vector sum of these leading two jets, which should in general have the most transverse momentum of any jets in an event. One variable that is useful to characterize this is the dijet invariant mass, m_{jj} , which is defined in Equation 6.1 below. A high m_{jj} means that the jets are well separated in η in addition to having large transverse momenta.

$$m_{jj} = \sqrt{2p_T(j_1)p_T(j_2)(\cosh(\Delta\eta_{jj}) - \cos(\Delta\phi_{jj}))} \quad (6.1)$$

A VBF event might have more than two jets due to final state radiation or fragmentation. If additional quarks or gluons are radiated from the outgoing jets, they should be in the forward regions of the detector— not the central region. Additional jets in the central region would suggest that this is more likely to be a background event. Equation 6.2 defines a measure of an additional i th jet’s

⁷⁰For the most part, the definitions presented here do follow the standard recommendations from the ATLAS “combined performance” (CP) groups, which study and produce algorithms and tools to identify particles and objects in reconstructed data. Additional detail, beyond the overview presented in Section 3.4, can be found in the linked references.

centrality relative to the dijet system, for $i > 2$ [144]. If the i th jet is located midway in η between the first two jets, it will have a centrality of 1, but this will approach zero as the jet becomes further and further forward.

$$C_i = \exp \left\{ - \left(\frac{2\eta_i - (\eta_1 + \eta_2)}{\Delta\eta_{jj}} \right)^2 \right\} \quad (6.2)$$

To ensure that an additional jet is likely to be final state radiation, we can compare its mass to the mass of the dijet system using the “relative mass” m_{rel}^i defined below in Equation 6.3. If the invariant mass of the vector sum of the additional jet with either of the two leading jets is large relative to m_{jj} , this implies that the event is unlikely to be consistent with vector boson fusion.

$$m_{\text{rel}}^i = \frac{\min(m_{j_1 j_3}, m_{j_2 j_3})}{m_{jj}} \quad (6.3)$$

6.3.2.2 Leptons and Photons

As shown in Figure 6.1a, a VBF+MET event does not contain any “visible” charged leptons in the final state. Therefore, a veto is applied: any event containing electrons or muons will be rejected from the signal region. For the purposes of this veto, the lepton definitions are quite loose, and only minimal quality requirements applied. Any electron [83] or muon [85] with $p_T > 4 \text{ GeV}$ and $|\eta| < 2.7$ in an event will cause that event to be rejected provided they pass the “loose” and “very loose” working points of the electron and muon identification algorithms. These working points correspond to different thresholds on the discriminants used in these algorithms and are set by the ATLAS Combined Performance groups. Here, even very weak evidence that an event might have a lepton will cause it to be vetoed from the signal region. On the other hand, more stringent definitions are used in the W and Z control regions. In the W CR, electrons must pass the “tight” identification and muons the “medium” identification working points for an event to be accepted, while in the Z CR, the “loose” working points are used for both.

Additionally, isolation and track association requirements are imposed in the control regions. Isolation is defined by drawing a cone of some radius around either the associated track or calorimeter deposits of the object, and measuring the momentum or energy of any tracks or topological clusters respectively within that radius [83] [85]. Electrons must have $p_T^{\text{varcone20}}/p_T < 0.15$ and $E_T^{\text{topocone20}}/p_T < 0.2$ to be accepted into the control region, where these variables are defined with a radius of $R = 0.2$. The “varcone” for track isolation here means that the cone radius actually varies with the p_T of the lepton, such that $R^{\text{varcone}} = \min(\frac{10}{p_T}, R)$. Muons must have $p_T^{\text{varcone30}}/p_T < 0.15$

and $E_T^{\text{topocone20}}/p_T < 0.03$ to be accepted, within a radius of $R = 0.3$. The electron or muon must also have an associated track consistent with having originated at the event’s primary vertex, as determined with a cut on the longitudinal and transverse impact parameters z_0 and d_0 . Electrons must have $|z_0 \sin(\theta)| < 0.5$ mm and $|d_0|/\sigma(d_0) < 5$, while muons have the same longitudinal requirement but require $|d_0|/\sigma(d_0) < 3$ instead.

While there are also no photons in the final state, in principle a photon can always be easily emitted as initial or final state radiation by one of the quarks. However, rather than look for invisible decay in association with a photon as part of this analysis, a separate targeted search (known as “VBF+MET+Gamma”) was set up to probe this signature [145]. In order to remove overlap between the two analyses, and ensure they are orthogonal, a photon veto was applied. The photon veto is defined using exactly the same photon definition as used in the VBF+MET+Gamma analysis. Events with photons that have $p_T > 15$ GeV and $|\eta| < 2.7$ (provided they are not within the transition region between the barrel and end-cap calorimeters ($1.37 < |\eta| < 1.52$)) will also be removed, provided the photon passes the “tight” identification working point and a tight isolation requirement. For photons, the tight isolation is defined as $E_T^{\text{topocone40}} < 0.40p_T + 2.45$ GeV and $p_T^{\text{cone20}}/p_T < 0.05$ [146].

6.3.2.3 Overlap Removal

Each type of object described above is identified using a separate set of tools, and so it is possible for jets, leptons, and photons to be identified which physically overlap in the detector. This can occur if, for instance, the same track and calorimeter deposit(s) are identified as both a muon and an electron by the electron and muon identification tools. A special overlap removal procedure is applied to remove overlapping objects and decide which one to keep. This is generally done by calculating the separation ΔR between the two objects, along with other considerations such as whether there are matching tracks. Standard ATLAS recommendations are used to perform overlap removal in this analysis; they are summarized below:

- If photons are within $\Delta R < 0.4$ of a lepton, they are rejected in favor of the lepton. On the other hand, if a photon is within $\Delta R < 0.4$ of a jet, the jet is rejected and the photon kept.
- If two electrons share a track, the one with the highest p_T is accepted.
- If a muon and an electron share a track, and the muon has calorimeter deposits, then the muon is rejected, as electrons are more likely to interact in the calorimeter. On the other hand, if

there are no calorimeter deposits, the muon is accepted and the electron rejected.

- If a jet and lepton overlap within $\Delta R < \min(0.4, 0.04 + 10/p_T(l))$, then the lepton is rejected and the jet accepted.
- On the other hand, if a jet is within $\Delta R < 0.2$ of a lepton, then the jet is rejected and the lepton accepted. For muons, the jet also must have less than three associated tracks.

6.3.2.4 Missing Transverse Momentum

The missing transverse momentum E_T^{miss} in an event is then defined as the magnitude of the negative vector sum of the transverse momenta of all the visible objects listed above [89]. Electrons, muons, photons, jets, are all included, along with any tracks associated with the primary vertex that were not matched to an object. When performing this sum, jets with momenta as low as $p_T > 20 \text{ GeV}$ are included, even though this is below the jet p_T requirement of 25 GeV . Jets with $p_T < 60 \text{ GeV}$ and $|\eta| < 2.4$ must also have a JVT score less than 0.50 to be included, which is again looser than the pileup rejection used for visible jets. Note that the vector sum of tracks not associated with the primary vertex, which is included as part of the E_T^{miss} , is known as the “soft term”, and labelled E_T^{soft} to distinguish this component from the sum of visible “hard” objects [89].

A related variable introduced for this analysis is $E_T^{\text{jet,no-jvt}}$: the magnitude of the vector sum of all jets with $p_T > 20 \text{ GeV}$ in the event. Unlike the jet component of the missing transverse momentum, $E_T^{\text{jet,no-jvt}}$ contains all jets without a JVT requirement applied. Cutting on this variable helps to remove events that only appear to have missing transverse momentum because they have real, non-pileup jets that fail the JVT requirement.

Because the E_T^{miss} is simply defined as the negative vector sum of all visible objects, an observed object can be marked as “invisible” by removing it from that vector sum. This is done in the one- and two- lepton control regions, where a corrected or “no lepton” E_T^{miss} is defined by marking the charged leptons as invisible. The main reason to introduce this corrected variable is to avoid changing the missing transverse momentum requirement (listed below) between the signal region and control regions. In the signal region, the missing transverse momentum approximates the transverse momentum of a boson decaying to an all-invisible final state (either the Higgs, in the case of the signal, or a $Z \rightarrow \nu\nu$ event, in the case of the background). By marking the leptons as invisible, the corrected missing transverse momentum will continue to correspond to the transverse momentum of the weak boson p_T^V in the control regions, where the weak boson instead decays to one or more visible leptons. More on the use of this variable can be found in Section 8.1.

6.3.3 Signal Region Definition

Using these definitions, the requirements listed below are imposed on events in order to enter the signal region. They can roughly be divided into two categories: requirements on the visible jets to try and select events where vector boson fusion may have occurred, and requirements on the missing transverse momentum E_T^{miss} to try and select events where a Higgs may have decayed invisibly:

- Events must contain no electrons, muons, or photons, as defined above.
- Events must contain exactly two, three, or four jets: $2 < N_{\text{jets}} < 4$.
- Events must contain less than two b -tagged jets, as determined using the multivariate MV2c10 b -tagging algorithm [88]. The discriminant from this algorithm must be less than 77% for a jet to *not* be considered b -tagged⁷¹.
- The leading jet must have $p_T > 80 \text{ GeV}$, and the subleading jet must have $p_T > 50 \text{ GeV}$.
- The leading and subleading jets must pass the forward jet vertex tagger algorithm introduced in Section 3.4 [91]: if the event has $E_T^{\text{miss}} < 200 \text{ GeV}$, the FJVT score for each jet must be less than 0.2, otherwise it must be less than 0.5.
- If the event has more than two jets, the third and fourth jets must not be central: the maximum centrality, as defined above, must be $\max(C_3, C_4) < 0.6$.
- Additionally, the maximum relative mass of the third and fourth jets, as defined above, must be $\max(m_{\text{rel}}^3, m_{\text{rel}}^4) < 0.05$.
- The invariant mass of the vector sum of the two leading jets (known as the “dijet system”), must be at least $m_{\text{jj}} > 0.8 \text{ TeV}$.
- The difference in pseudorapidity between the two leading jets must be at least $|\Delta\eta_{\text{jj}}| > 3.8$, and the jets must be in opposite hemispheres with $\eta(j_1)\eta(j_2) < 0$.
- The leading two jets must not be back-to-back, with an azimuthal separation $|\Delta\phi_{\text{jj}}| < 2.0$.
- The missing transverse momentum must be at least $E_T^{\text{miss}} > 160 \text{ GeV}$, with a soft track term of only $E_T^{\text{soft}} < 20 \text{ GeV}$.

⁷¹As with the photon veto, this is applied to ensure this analysis remains orthogonal with a $t\bar{t}H$ search, where the top fusion process shown in Figure 6.1b will lead to b -jets in the final state.

- The magnitude of the vector sum of all jets with $p_T > 20 \text{ GeV}$ in the event, with no JVT requirement applied, must be at least $E_T^{\text{jet,no-jvt}} > 140 \text{ GeV}$.

The above selection was reoptimized for the full run 2 analysis; both the preliminary and final 139 fb^{-1} results use the same set of cuts, with one key exception. In the preliminary 139 fb^{-1} analysis, the missing transverse momentum was required to be at least $E_T^{\text{miss}} > 200 \text{ GeV}$, while the corresponding $E_T^{\text{jet,no-jvt}}$ requirement was increased to 180 GeV . These higher E_T^{miss} cuts were needed in order to reduce the number of QCD multijet events with fake missing transverse momentum that were contaminating the signal region; the probability of an event having fake E_T^{miss} becomes lower as the E_T^{miss} threshold increases. For the final 139 fb^{-1} result, work done was to improve the modelling of this multijet background, enabling the E_T^{miss} cut(s) to be lowered. This issue is described in more detail in Section 8.4.

6.3.4 Binning

The signal region is further divided into a number of “bins”, into which events are sorted. Both the strength of the invisible Higgs signal, and the amount of background contamination, can vary across a number of different variables. Some areas of the signal region might be relatively pure in signal, with a high S/B , but have very low statistics. Others might have relatively high statistics, but have a very large amount of one or more background processes. In some areas the background estimate itself might be statistically limited, or the Monte Carlo used to model some of the backgrounds might have modelling problems as a function of one or more variables. Dividing the signal region into multiple bins allows the background prediction in each bin to vary independently, and allows the contribution of each bin to the predicted limit to vary as well.

The 36 fb^{-1} analysis was binned in one variable: the invariant mass of the dijet system, m_{jj} [135]. When this analysis was reoptimized, we moved to a multidimensional binning that used *three* variables: m_{jj} , the azimuthal separation $\Delta\phi_{\text{jj}}$, and the number of jets N_{jets} . The binning strategy used in the preliminary 139 fb^{-1} analysis is shown below in Figure 6.5: events with exactly two jets are divided into two regions with low (0-1) and high (1-2) $\Delta\phi_{\text{jj}}$. These regions are then each split into five m_{jj} bins: $800 < m_{\text{jj}} < 1000 \text{ GeV}$, $1000 < m_{\text{jj}} < 1500 \text{ GeV}$, $1500 < m_{\text{jj}} < 2000 \text{ GeV}$, $2000 < m_{\text{jj}} < 3500 \text{ GeV}$, and $m_{\text{jj}} > 3500 \text{ GeV}$, giving a total of ten bins. Finally, events with $N_{\text{jets}} > 2$ are grouped into an eleventh bin. As shown in the plot, the highest m_{jj} bins tend to be the most sensitive to the invisible Higgs signal, and have the lowest amount of background contamination.

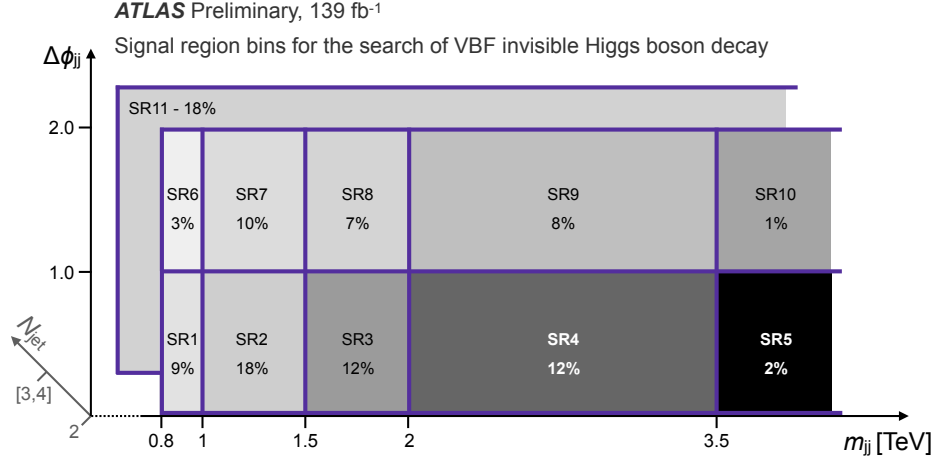


Figure 6.5: Plot illustrating the binning scheme used to separate the signal region into “bins”, as used in the preliminary 139 fb⁻¹ analysis. The preliminary result was binned in three variables: m_{jj} , $\Delta\phi_{jj}$, and N_{jets} as shown in the plot for a total of 11 bins. The percentages in each bin illustrate the fraction of expected $H \rightarrow \text{inv.}$ signal, while the shading illustrates the relative size of the ratio of signal to background, S/B . As can be seen in the plot, the smaller m_{jj} bins do not have much sensitivity, while the largest m_{jj} bins have very little background and a large amount of signal.

For the final 139 fb⁻¹ analysis, this strategy was slightly modified. The bins used in the preliminary analysis described above all required $E_T^{\text{miss}} > 200$ GeV. When this was lowered to 160 GeV, the missing transverse momentum cut in the existing eleven bins was left unchanged. Instead, a new $160 < E_T^{\text{miss}} < 200$ GeV region was added, split into three bins: $1500 < m_{jj} < 1000$ GeV, $2000 < m_{jj} < 3500$ GeV, and $m_{jj} > 3500$ GeV. The region with $m_{jj} < 1500$ GeV and $E_T^{\text{miss}} < 200$ GeV remained excluded from the analysis, as it is not that sensitive to the signal and the multijet contamination remained relatively large. In addition to adding these three bins, the $N_{\text{jets}} > 2$ bin was also split into three $1500 < m_{jj} < 1000$ GeV, $2000 < m_{jj} < 3500$ GeV, and $m_{jj} > 3500$ GeV bins as well, following a suggestion made during the internal review process, as it appeared to improve the overall sensitivity of the limit. That gives a total of 16 bins, as shown in Figure 6.6.

A numbering scheme is commonly used to quickly refer to the different analysis bins in shorthand form, in which bins are ordered first by $\Delta\phi_{jj}$ and then m_{jj} . So, bins 1-5 are the low $\Delta\phi_{jj}$ bins, with bin 1 the $800 < m_{jj} < 1000$ GeV bin, and so on. In the final analysis, bins 11-13 are the $N_{\text{jets}} > 2$ bins, again sorted by m_{jj} , and bins 14-16 the lower E_T^{miss} bins. This numbering scheme is used in a number of the results plots shown in Chapter 9.

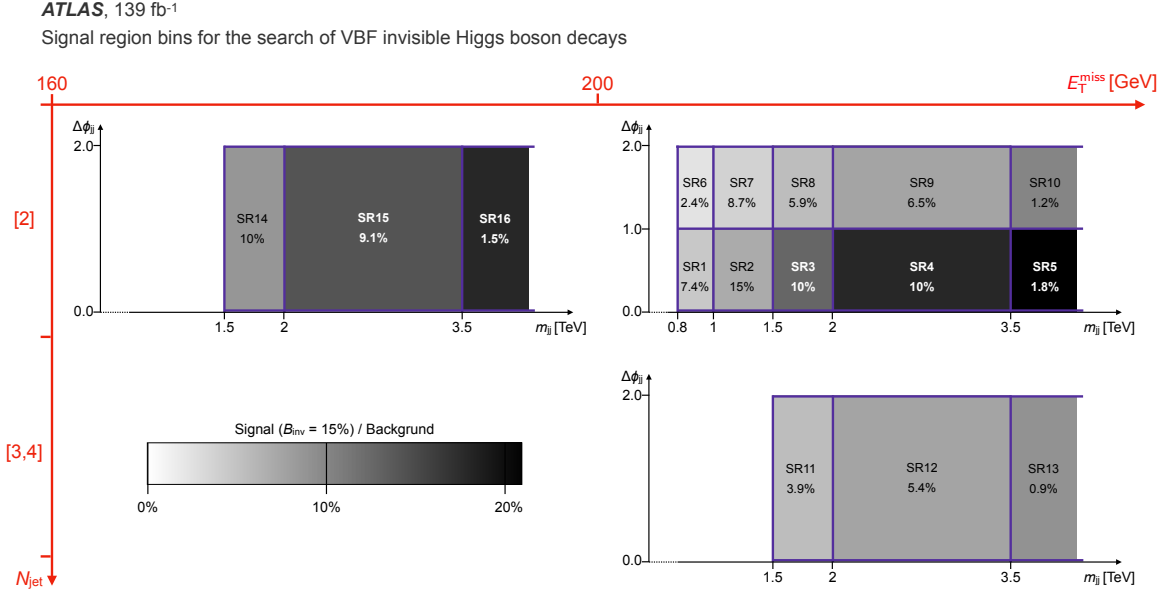


Figure 6.6: Plot illustrating the binning scheme used to separate the signal region into “bins”, as used in the final 139 fb⁻¹ analysis. The scheme shown in Figure 6.5 was modified through the addition of a low $160 < E_T^{\text{miss}} < 200$ GeV region, split into three m_{jj} bins, as well as by the division of the inclusive $N_{jets} > 2$ bin into three separate m_{jj} bins, for a total of 16.

6.4 Experimental Uncertainties

The definitions and selections described above are not perfect. It is not possible to identify an electron without any probability of error, or to measure the energy of a jet with perfect accuracy. The efficiencies of the reconstruction and identification algorithms are known within some uncertainty, and these uncertainties must be applied as a systematic effect when trying to predict the number of events in the signal region. Since these systematic uncertainties involve effects related to detector effects, they are known as *experimental* systematic uncertainties, to distinguish them from other *theoretical* systematic uncertainties on the signal and background modelling that are not detector dependent. This section presents a brief overview of the various sources of experimental error⁷², while the theory systematics are described below in Section 6.5 and throughout Chapters 7 and 8.

One way in which experimental effects can be applied to the signal region prediction is by means of scale factors. Despite our best efforts, algorithms used to reconstruct events and identify objects may behave differently when run over simulated Monte Carlo datasets instead of real data from the ATLAS detector. These differences can be studied and quantified, sometimes as a function of one

⁷²As with the object definitions, many of these systematics are computed using the latest recommendations and tools developed by the ATLAS Combined Performance groups, and were not independently rederived for the VBF+MET analysis. Readers interested in more detail should consult the linked references.

or more variables. A scale factor can then be used to correct the Monte Carlo prediction in order to make it better able to explain real data. The scale factors will have a corresponding uncertainty, which is taken into account as an experimental systematic. Several of the effects mentioned below involve the use of one or more scale factors.

6.4.1 Triggers

The E_T^{miss} triggers [141] used to record data in the signal region have what is known as a “turn-on curve”. That is, the trigger level E_T^{miss} only takes into account some calorimeter information, and so it cannot fully agree with the E_T^{miss} calculated offline after fully reconstructing the event. Because the turn-on curve can in principle be different between data and Monte Carlo, and because the trigger is not fully efficient when requiring $E_T^{\text{miss}} > 160 \text{ GeV}$, a scale factor and corresponding systematic uncertainty is needed to correct for any potential mismodelling. This scale factor is measured as the ratio of the efficiency in data ϵ_{data} to the efficiency in MC, ϵ_{MC} , so determining this requires a way to calculate ϵ_{data} . Because the ATLAS trigger system does not just consist of a single trigger but a large menu of many different triggers, it is possible to use a second, independent trigger in order to measure the ϵ_{data} . By counting the number of events which pass just the second trigger, and then comparing to the number of events which pass the combination of the second trigger with the E_T^{miss} trigger, the efficiency can be calculated.

For this analysis, the single-lepton (single- μ and single- e triggers) are used to calculate ϵ_{data} as a function of the missing transverse momentum after reconstruction. ϵ_{MC} is then much easier to compute, since the number of events before and after emulating the trigger are both known quantities. Since ϵ_{data} was determined using single-lepton triggered data, ϵ_{MC} should also be computed using Monte Carlo samples that have both a visible lepton and E_T^{miss} , so $W \rightarrow l\nu$ MC is used. This assumes that $W \rightarrow l\nu$ MC can be used to extrapolate to the zero-lepton signal region, and so the efficiency for $Z \rightarrow \nu\nu$ is also computed. When studying these effects, the efficiency from the muon samples was found to agree better with the $Z \rightarrow \nu\nu$ than the electron samples, and so the decision was made to only use $W \rightarrow \mu\nu$ MC and single muon triggered events to calculate the transfer factors.

$$\text{SF}(E_T^{\text{miss}}) = \frac{\epsilon_{\text{data}}}{\epsilon_{\text{MC}}} \frac{1}{2} \left(1 + \text{erf} \left(\frac{E_T^{\text{miss}} - p_0}{p_1 \sqrt{2}} \right) \right) \quad (6.4)$$

The function shown above in Equation 6.4 was fit to the scale factors, and the uncertainties on the fit parameters p_0 and p_1 was taken as a systematic uncertainty. Additionally, the difference between the efficiency curve for $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ events was assigned as another systematic

uncertainty. Both of these effects were found to be relatively small, on the order of less than 1%. Note that because the E_T^{miss} trigger thresholds varied over the course of run 2 as explained above, this process was performed separately for the events recorded with each trigger threshold. Similar methods were used to apply scale factors for the single-lepton and di-lepton triggers used in the control regions [147] [148].

In addition to the trigger-level effects, the amount of data recorded by ATLAS is not known with perfect accuracy. Across the entirety of run 2, ATLAS recorded an integrated luminosity of 139 fb^{-1} with a combined uncertainty of 1.7% [56]. Since the shape of the background estimate needs to be scaled by the integrated luminosity, this uncertainty on the luminosity also needs to be applied as a systematic uncertainty.

6.4.2 Leptons

Like the trigger case explained above, the identification and reconstruction algorithms for electrons and muons have an efficiency which may differ between data and Monte Carlo. Scale factors to cover this difference are calculated and applied for both electrons and muons, and so systematic uncertainties on those scale factors are assigned [82] [85]. In addition, the electron and muon reconstruction algorithms depend on the energy scale and resolution chosen [83] [149]. The algorithms are ran with variations of these parameters in order to ensure that their choice does not have a major impact on the result, and these variations are used to calculate systematic uncertainties.

One complication for the VBF+MET analysis is that, as described above, a lepton veto is applied in the signal region. But if the efficiency of lepton identification can vary between data and MC, it follows that the *inefficiency* will also vary, which means that the lepton veto could behave differently. An inefficiency scale factor, and corresponding systematic uncertainty, is therefore needed to correct the prediction in the signal region when vetoing events with leptons. These factors are known as “anti-ID scale factors”, and computed separately for electron and muon events. The anti-ID scale factor is defined as $(1 - \epsilon_{\text{data}})/(1 - \epsilon_{\text{MC}})$, where ϵ is the efficiency of the lepton identification algorithm. These scale factors were determined for electrons and muons as follows:

- For electrons, anti-ID scale factors were computed as functions of p_T and η using preliminary ATLAS recommendations⁷³, and applied to all Monte Carlo events with electrons with $p_T > 4 \text{ GeV}$ and $|\eta| < 2.47$ before simulating the effects of the identification and reconstruction

⁷³For readers with access to internal ATLAS documents and pages, see this wiki page for the technical details: <https://twiki.cern.ch/twiki/bin/view/AtlasProtected/ElectronEfficiencyAntiID>

algorithms. The uncertainty on this scale factor is taken as a systematic, and the average correction factor is 1.06 ± 0.20 . The total impact of this systematic across the entire signal region is approximately 0.8%.

- For muons, recommendations from ATLAS on how to compute anti-ID scale factors were not available at the time. Therefore, studies were done using Monte Carlo samples, in which the average scale factor was found to be very close to 1. The decision was taken to apply a scale factor of 1 ± 0.20 to events with at least one muon with $p_T > 4 \text{ GeV}$ and $|\eta| < 2.47$ prior to reconstruction, meaning that the prediction is not actually rescaled but a systematic uncertainty is still applied. This 20% uncertainty was chosen to match what was seen above for electrons, and has a total impact of approximately 0.4%.

6.4.3 Jets and E_T^{miss}

Experimental systematics are also applied due to jet reconstruction algorithms. The two main systematics involving jets are the choices of the jet energy scale and jet energy resolution [150] [151]. Much like the corresponding parameters used in lepton identification, the jet energy scale and jet energy resolution are varied, and a systematic uncertainty assigned by comparing the default parameter choice to the result of the variation. In addition to these variations, uncertainties on the jet vertex tagging [90] and forward jet vertex tagging [91] scores used in the signal region definition also need to be taken into account.

The missing transverse momentum is calculated from the jets and leptons, and so when considering the uncertainties on the E_T^{miss} , the jet and lepton systematics mentioned above have already been taken into account. One effect that has not already been considered, however, is the “soft term”, E_T^{soft} . Track-finding resolution and scale uncertainties lead to uncertainties when calculating E_T^{soft} , and so they are included as separate systematics in the analysis [152].

6.5 Invisible Higgs Signal Modelling

As mentioned above, the event selection described in Section 6.3 was reoptimized from the 36.1 fb^{-1} analysis. This reoptimization was done while the signal region was blinded, without looking at real data. Instead, a simulated Monte Carlo dataset of the Higgs to Invisible signal process, along with Monte Carlo simulations of the background processes, were used in order to iterate and refine the signal region selection. This section presents a summary of how the signal modelling is done, some

of the uncertainties involved, and the simulated efficiency of the reoptimized signal region. More technical details on Monte Carlo generation, especially for the background processes, can be found in the next chapter.

6.5.1 Signal Monte Carlo

Higgs to Invisible signal processes are simulated by forcing a Higgs boson to decay to a ZZ^* pair, which are then forced to decay to four neutrinos– the only all invisible final state the Higgs can produce. While this is not the exact signal process being searched for (we are interested in the Higgs decaying to some new invisible particle, not a neutrino), it is a reasonable enough approximation. In these signal samples, the Higgs will always decay invisibly: in other words, the invisible branching ratio is set to $\mathcal{B}_{H \rightarrow \text{inv.}} = 1$. During the fit, the branching ratio will be varied in order to find the value which best explains the observed data.

The main target for the VBF+MET analysis is, as the name implies, a Higgs that was produced via vector boson fusion. However, in principle, a Higgs can also be produced in association with two jets that satisfy the VBF-like signal region jet criteria via one of the other production modes shown in Figure 6.1. Therefore, it makes sense to produce Higgs to invisible samples for more than just the VBF production process, and use them as part of the signal as well. ATLAS searches for invisible Higgs decays in other channels like Higgs-strahlung as well, and so samples for each production mode are produced centrally for use by all of the different analyses. In this analysis, ggF and VH samples are included as part of the signal in addition to VBF events.

When generating Monte Carlo for a process, a critical question is which diagrams should be included. The tree-level, [Leading Order \(LO\)](#) VBF Higgs to Invisible process shown in Figure 6.1a can of course be perturbatively modified by the emission of extra particles, creating [Next to Leading Order \(NLO\)](#) diagrams. And those diagrams can then be further modified to create [Next to Next to Leading Order \(NNLO\)](#) diagrams, and so on. The VBF signal MC used here is generated at NLO in the strong force, α_s , meaning that diagrams containing up to one additional quark or gluon (and one additional QCD vertex) are considered during event generation. The VH signal was also produced at NLO, while the ggF signal was produced at NNLO, an even higher level of accuracy. The VBF and ggF cross sections were reweighted upwards to NNLO or N3LO⁷⁴ in the strong force, respectively, using calculations from the LHC Higgs working group [124]. Uncertainties on these cross sections are applied as a theoretical systematic uncertainty. Additionally, while the

⁷⁴That is, “next to NNLO” or “next to next to next to leading order”, considering one level of corrections beyond NNLO.

MC samples do not include NLO electroweak diagrams (at NLO in α , e.g. via the emission of an additional photon), dedicated NLO electroweak corrections are also included. The program HAWK was used to calculate these for the VBF sample [153], and the correction as a function of Higgs p_T was found to be $\alpha_{\text{NLO EWK}}^{\text{HAWK}} = -0.000350 \text{ GeV}^{-1} p_T - 0.0430$. Uncertainties on these electroweak corrections are also computed as about 2% of the total signal, and applied as another theory systematic.

There are additional theory systematics on the signal modelling associated with choices made during Monte Carlo generation. Algorithms and some scales and parameters need to be chosen when producing a simulated dataset, and ideally the generated events should be independent of these choices. Variations of these choices are performed in order to verify this, and theory systematic uncertainties assigned by comparing the variation to the default value. A detailed explanation of these choices and these systematics in the signal samples requires a much more thorough explanation of how MC generation actually works. See Sections 7.1 and 7.2 for these details in the next chapter.

6.5.2 Blinded Signal Region

Plots of these invisible Higgs signal samples, along with background Monte Carlo, are shown above in Figure 6.7 with the reoptimized signal region definition from above applied. The ratio S/B is shown in the bottom panel, along with a very rough approximation of the potential sensitivity of a 95 confidence level limit on the invisible Higgs branching ratio, which is defined below in Equation 6.5. Since the invisible Higgs branching ratio will depend on the statistics of the W and Z control regions as well as the signal region, this approximation tries to take into account the numbers of simulated events in these regions as well, assuming a 2% contribution from the W control region. The analysis was reoptimized in order to maximize S/B and also minimize σ_μ^{95} , in order to try and set as low as possible a limit on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$ as possible⁷⁵. As can be seen in the plots, as the dijet invariant mass m_{jj} increases, the amount of background decreases and the analysis becomes sensitive to the signal.

$$\sigma_\mu^{95} = \frac{2}{N_{\text{signal}}} \sqrt{N_{\text{background}} + (0.02N_W)^2 + (\sigma_z N_Z)^2} \quad (6.5)$$

⁷⁵Note that this prediction of the limit assumes the Z control region is more important than the W control region, because it was previously used to model the $Z \rightarrow \nu\nu$ background (during the preliminary 139 fb^{-1} analysis). As of the final version of the 139 fb^{-1} analysis, this is no longer true: the W CR is the main region used to model both the $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ backgrounds (see Section 8.2). This might suggest the next round of the analysis should again be reoptimized with this in mind.

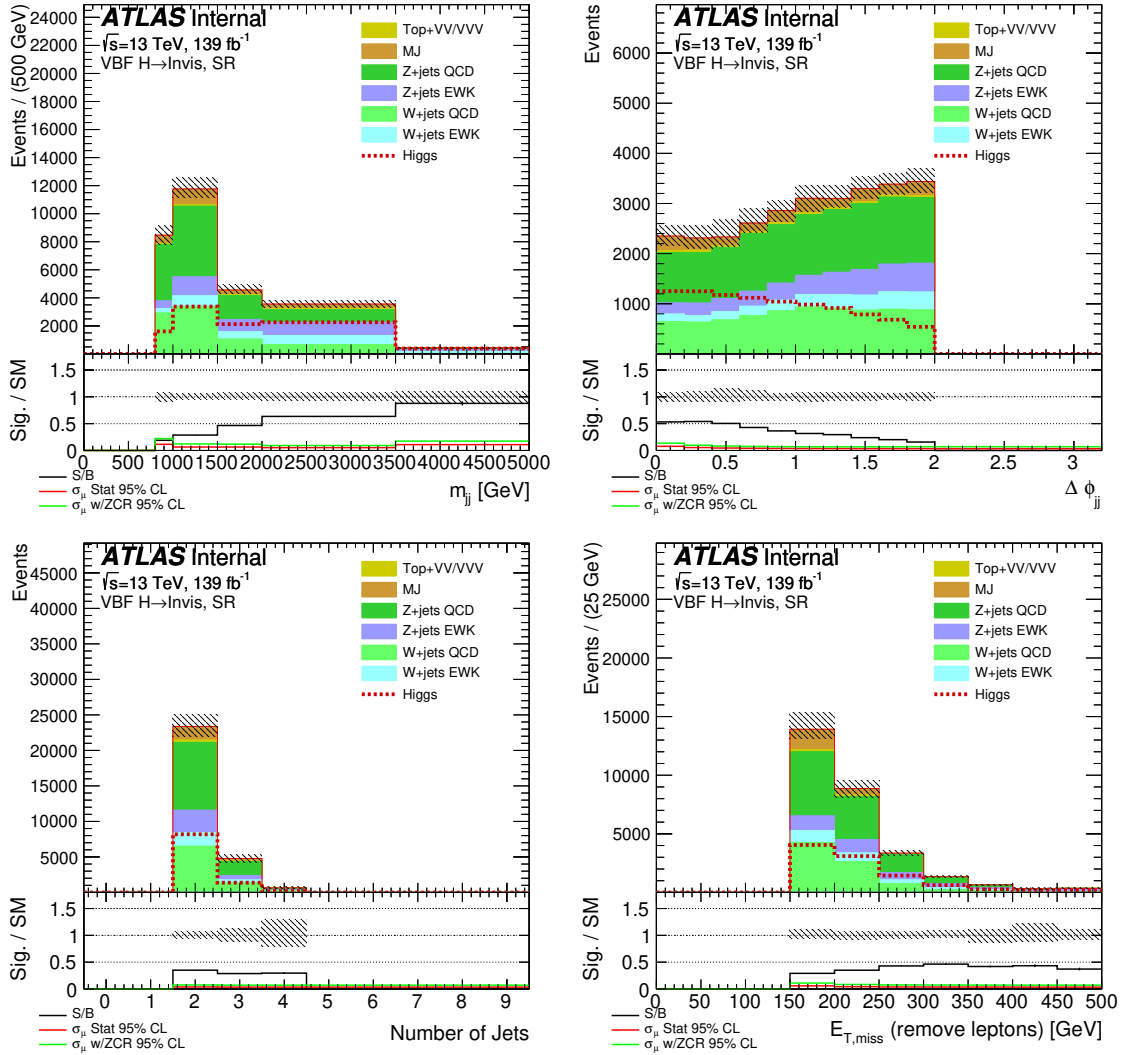


Figure 6.7: Pre-fit plots of m_{ii} , $\Delta\phi_{ii}$, N_{jets} , and $E_{\text{T}}^{\text{miss}}$ in the VBF+MET signal region. No data is shown, as the signal region is blinded: the red dashed line shows a Monte Carlo simulation of the Higgs to invisible signal compared against Monte Carlo simulations of the various background processes. The error band includes both statistical and systematic uncertainties.

Table 6.1 then shows the efficiency of each cut in the signal region definition on the VBF Higgs to Invisible signal. Note that this includes a few redundant cuts for technical reasons; an initial lepton veto is applied twice before a stricter one is applied again later in the flow. Also, there are two “jet timing” cuts listed here that were ultimately not included in the optimized signal region definition. These cuts would have required the two leading jets to have been recorded within 11 ns of a bunch crossing to reduce the possibility of accidentally selecting “out-of-time” pile-up jets from

a previous event. This cut had a negligible impact on pile-up mitigation and so was removed in favor of the JVT and FJVT requirements.

Cut	VBF			ggF		
	Events	Efficiency	Stepwise Eff.	Events	Efficiency	Stepwise Eff.
Initial	523500	1	1	6752600	1	1
Higgs $p_T > 75$ GeV	268000	0.51200	0.51196	268000	0.03970	0.03970
Initial Skim	47600	0.09100	0.17780	25900	0.00380	0.09660
MET Trigger	34100	0.06520	0.71620	17280	0.00260	0.6670
Jet Cleaning	33200	0.06350	0.97440	17020	0.00250	0.98470
Initial Lepton Veto	33100	0.06330	0.99640	16850	0.0030	0.99020
$N_{\text{photon}} = 0$	33000	0.06300	0.99670	16730	0.00250	0.99300
$2 < N_{\text{jets}} < 4$	31700	0.06050	0.95940	13660	0.00200	0.8160
$\max(C_3, C_4) < 0.6$	28700	0.05480	0.90590	9300	0.00140	0.68050
$\max(m_{\text{rel}}^3, m_{\text{rel}}^4) < 0.05$	22800	0.04350	0.79400	5430	0.0010	0.58370
$p_T^{j_1} > 80$ GeV	21100	0.04030	0.92580	4800	0.00070	0.88410
$p_T^{j_2} > 50$ GeV	21100	0.04020	0.99800	4750	0.0010	0.98960
b -jet Veto ($N_b < 2$)	21000	0.04020	0.99980	4740	0.0010	0.99940
Lepton Veto	20500	0.03920	0.97430	4520	0.00070	0.95190
$E_T^{\text{soft}} < 20$ GeV	20000	0.03820	0.97570	4130	0.00060	0.91360
Timing of j_1	20000	0.03820	0.99980	4120	0.00060	0.99950
Timing of j_2	20000	0.03810	0.99760	4100	0.00060	0.99490
FJVT	19800	0.03790	0.99318	4020	0.00059	0.97900
$E_T^{\text{miss}} > 160$ GeV	11800	0.02250	0.59400	2380	0.00040	0.59200
$E_T^{\text{jet, no-jvt}} > 140$ GeV	11700	0.02240	0.99730	2370	0.00040	0.99710
$\Delta\phi_{jj} < 2$	10200	0.01950	0.86990	1920	0.00030	0.811500
$\eta_{j_1}\eta_{j_2} < 0$	10100	0.01930	0.98870	1870	0.00030	0.97090
$\Delta\eta_{jj} > 3.8$	9200	0.01770	0.91520	1480	0.00020	0.79070
$m_{jj} > 800$ GeV	8400	0.01600	0.90360	1080	0.00020	0.73390
m_{jj} - N_{jets} Binning	7600	0.01460	0.91440	960	0.00010	0.88560
m_{jj} - N_{jets} Binning	5900	0.01130	0.77370	710	0.0001	0.73850

Table 6.1: Cutflow table showing the global and individual (stepwise) efficiency of each cut on the VBF and ggF Higgs to Invisible signal process, as calculated on the MC samples. The event yields are normalized to a luminosity of 139fb^{-1} and a VBF Higgs production cross section of 3.782pb . The second and third entries are both “initial skimming” cuts to reduce the size of the dataset before proceeding; in the third, events are required to have two jets with $\Delta\eta_{jj} > 2.0$ and $p_T > 50, 40$ GeV. In the two final entries, events are required to have $m_{jj} > 1500\text{GeV}$ if they have $N_{\text{jets}} > 2$ or $160 < E_T^{\text{miss}} < 200\text{GeV}$, in order to match the analysis binning shown in Figure 6.6.

6.5.3 Background Processes

There are four main sources of background that are considered in this analysis, as shown in the plots in Figure 6.7. Some have already been introduced above, but to summarize, these backgrounds are:

- Z+jets events, where the Z boson decays to a pair of neutrinos. This is the main background of the analysis. This $Z \rightarrow \nu\nu$ background is modelled using Monte Carlo, and corrected with

data from a dedicated control region.

- W+jets events, where the W boson decays to a visible charged lepton and a neutrino, but we subsequently fail to reconstruct the charged lepton in the detector. This can occur due to various reasons: hadronic τ decays, leptons that are too far forward, or due to inefficiencies in the lepton reconstruction algorithms. Along with $Z \rightarrow \nu\nu$, this is the other primary background of the analysis. This $W \rightarrow l\nu$ background is also modelled using Monte Carlo, and corrected with data from a dedicated control region.
- QCD “multi-jet” events, where the event appears to have a transverse momentum imbalance due to jet measurement errors. These multijet events with fake E_T^{miss} are not modelled directly using Monte Carlo; rather, a complex data-driven estimate is performed, as described in Sections 8.4 and 8.5. This is a relatively small background compared to the two V+Jets processes.
- Other backgrounds include top processes (single- t or $t\bar{t}$) or multi-boson events (VV , VVV , etc.). These processes can also in principle decay to an all-hadronic final state with missing transverse momentum that appears to satisfy the VBF jet requirements, though— as shown in Figure 6.7— this is relatively rare. Due to their small size, these processes are just modelled using Monte Carlo without a data-driven correction.

The Monte Carlo methods used to generate all these processes are described in detail in the next chapter.

CHAPTER 7

Efficient Monte Carlo Generation

As the last chapter discussed, there are many uses for simulated collision data when performing a physics analysis. Generating this simulated data, known as Monte Carlo, can be quite complex, as the particle physics and field theory being simulated is quite complex. Calculations are perturbative, and diagrams for a given process are only included up to a certain order and a certain number of loops and extra interactions. This means that a sophisticated understanding of QFT is required to generate MC for a given process: in fact, particle theorists have collaborated to develop various Monte Carlo generators, which are used by the experimental community. These generators are software packages which have been programmed to simulate strong and electroweak physics processes up to a certain level of accuracy, frequently only **LO** or **NLO**. Multiple generators might be chained together to produce a single dataset: often, one program will be used to handle the “matrix element” calculation, where a Feynman diagram is randomly generated; and another might be used to handle the “parton shower” phase, where quarks and gluons shower and hadronize, producing more quarks, gluons, and eventually composite hadrons which would show up in the detector as jets. These two stages are explained in more detail below.

The events produced by a Monte Carlo generator are known as “truth-level” events. At truth-level, all information about the event should be available: for instance, the fact that a given particle is a muon, or an up quark, or a photon is saved as part of the event record, so particles can be perfectly identified. At truth-level, neutrinos are also directly available in the event record, and so can be inspected directly without having to calculate the missing transverse momentum. In principle, the truth record can also contain information about the initial-state particles that interacted and the intermediate particles produced from that interaction, allowing final-state particles to be associated

with specific vertices⁷⁶. Having access to the truth record can be useful when performing studies, but to use these MC datasets in an analysis, it is necessary to first simulate the effect of seeing the event in the ATLAS detector. This “simulation” step uses a sophisticated model of ATLAS to simulate the reconstruction algorithms described in Chapter 3 over the truth event, producing “reco”-level MC [154]. This chapter focuses on truth-level studies, while the next chapter explores how the reconstructed MC will be used to estimate backgrounds in the analysis.

More detail on these Monte Carlo generators is described in Section 7.1; since the focus of this chapter is on truth-level work, this section only briefly covers the simulation/reconstruction step. Next, an overview of all the signal and background samples used in the VBF+MET analysis is presented in Section 7.2; this contains an overview of the various theoretical uncertainties for each sample as well. The remainder of the chapter focuses on the particular issues involved in efficiently generating high-statistics V+Jets background samples, something I was directly involved in attempting to solve. Section 7.3 explores the use of matrix element level m_{jj} filtering to produce V+Jets QCD Monte Carlo. And Section 7.4 then describes a study attempting to understand differences that appear in these V+Jets samples between W and Z processes. Finally, details about the electroweak V+Jets samples can be found in Section 7.5.

7.1 Monte Carlo Methods

The process of generating Monte Carlo datasets for particle physics is quite complex, and involves many steps and highly specialized software. Many different algorithms for computing matrix elements and parton showers are implemented in different generators. This section tries to give a general overview of the main steps involved and the principles behind them involved in producing MC for use in an ATLAS analysis, as well as an overview of the theoretical uncertainties involved. Because the studies in this chapter were done using the Sherpa generator, some detailed information about how Sherpa implements some specific parts of event generation (such as the merging of matrix element and parton shower calculations) are included [155] [156]. A full review of all generators and event generation algorithms is beyond the scope of this thesis, but more information can be found in the linked references [154].

⁷⁶In practice, this is not always the case. The generators will save most of the information they have available... but what ‘most’ means, and how easy that information is to interpret, may vary from generator to generator, and it may not always be possible to untangle it and reconstruct intermediate particles.

7.1.1 Matrix Element Calculations

When generating a Monte Carlo dataset, the core process or processes being generated first needs to be specified. A single process may include many different Feynman diagrams up to varying orders in the QCD and electroweak coupling constants, α_s and α . For example, if simulating the production of Z bosons from a proton-proton collision that then decay to $Z \rightarrow \nu\nu$ plus at least two jets, the tree-level process would be $pp \rightarrow \nu\bar{\nu}jj$ ⁷⁷. But diagrams at NLO in either α_s or α can of course be added with the emission of extra hadrons or photons or loops, and then NNLO corrections could be added, and so on and so forth. Additionally, “at least two jets” means that the processes $pp \rightarrow \nu\bar{\nu}jjj$, $pp \rightarrow \nu\bar{\nu}jjjj$, and so on would theoretically be included. It is computationally not possible to produce events with all possible matrix elements for this process, so a more restricted phase space needs to be defined. Higher order corrections and additional QCD effects not included in the matrix element phase space will be considered during the parton shower process, as explained below. Note that the phase space might also contain one or more cuts on variables like the momenta of the initial or final state particles, in order to further restrict the phase space being considered. More discussion on the use of cuts to constrain the phase space can be found below in Section 7.3.

After specifying the phase space, events are generated using random sampling to select a “point” in that phase space. A point corresponds to a specific Feynman diagram, with the particles in that diagram having specific four-momenta. In order to do this, the probability of selecting that point—that is, the probability the particles in a given process will have certain momentum values—needs to be known. This is difficult to determine analytically, and so a numerical integration method is used instead [157]. The matrix element function is integrated by evaluating it for a large number of phase space points, labelled $\vec{x}$. This determines inclusive cross section of the processes in the phase space and to calculate a probability distribution $g(\vec{x})$, which is normalized to unity. Then, when events are generated, a random value of $\vec{x}$ is selected, the matrix element $f(\vec{x})$ evaluated, and the probability of selecting that point $g(\vec{x})$ determined [158]. A critical concept is that, because some included diagrams and some momenta values will be more likely than others, the generated events need to be assigned an *event weight*, $w(\vec{x}) = f(\vec{x})/g(\vec{x})$. While $g(\vec{x})$ is normalized to unity, individual events can in principle be generated with weights that are significantly below or above one. The expectation value of the (normalized) integral is then the average event weight, $\langle w(\vec{x}) \rangle$, and its variance the average event weight squared, $\langle w^2(\vec{x}) \rangle$. If N events are then generated in this phase space, the true number of events generated is not N but the sum of event weights $N' = \sum_i^N w(i)$.

⁷⁷Here “j” indicates a final-state quark or gluon which will evolve into a jet during the parton shower and hadronization step. It is not yet a jet during the matrix element calculation.

And likewise, the uncertainty from N events is the sum of the event weight squared [158].

A number of tools to perform phase space integration and then generate matrix elements have been developed that encode knowledge about many different processes. In many cases these involve the use of “multi-channel” methods, in which the matrix element phase space is split and each diagram or type of diagram comprising it is integrated independently, and then combined. Sherpa, for example, contains two built-in packages called AMEGIC++ [157] and Comix [159] which implement different algorithms that make use of multi-channel methods for performing these calculations [155]. Several other generators, such as MadGraph [160], are capable of saving the output of the matrix element calculation as an intermediate format, known as a Les Houches or LHE file [161]. These LHE files are then passed as an input to another generator to perform the parton shower step, described below.

7.1.2 Parton Showers

The matrix element will generally contain a small number of final-state quarks or gluons that are referred to as partons. In a real event, these partons will undergo a hadronic shower, creating many more quarks and gluons, which then hadronize and produce jets. This effect needs to be simulated as part of the Monte Carlo generation step. It is computationally impossible to include these QCD effects as part of the matrix element phase space, and so parton shower models have been developed instead. There are several different approaches, but generally speaking a parton shower is modelled by continually evaluating the Sudakov form factors that govern the probability of a parton emitting another particle [162]. This process is continuously performed, creating emissions with lower and lower momenta until some cut-off value is reached, often on the order of 1 GeV. After this point, a hadronization procedure will be used to form composite particles like mesons and baryons, which will form jets.

A number of different algorithms have been developed for simulating parton showers. Sherpa implements two [155], one based on Catani-Seymour dipole factorisation [163] [162] and another involving dipole resummation known as DIRE [164]. Other implementations can be found in other generators, such as Pythia [165]. In many cases Pythia will be used to shower the LHE files containing matrix elements from another generator. A source of uncertainty comes from which parton shower implementation was chosen during event generation. When the matrix element has been saved as an LHE file, it can be possible to simulate the same event with different generators, for instance comparing the results of the Pythia shower with the one implemented in the generator package

Herwig. Any difference between the showers can be taken as a theory systematic. In other cases, this is not possible, such as when using a generator like Sherpa with tight integration between the matrix element and parton shower. In this case, the generator may be able to calculate an internal parton shower uncertainty by varying parameters and storing this variation as another event weight.

7.1.3 Parton Distribution Functions

When simulating LHC collisions, the matrix element will also contain two initial state partons, which will have each been part of the two protons that are actually colliding with each other. Which partons interact with each other is determined by the structure or [parton distribution function \(PDF\)](#) that define the composition of the proton [12]. A PDF needs to be passed as an input to the Monte Carlo generator and used in both the matrix element and parton shower. The PDF is both used to determine the probability of seeing a particular quark or gluon in the initial state of the matrix element and the probability of initial state radiation in the parton shower. In addition, other partons in the matrix element may also interact beyond the core process, showering and forming jets of their own. Taking all these effects into account requires these functions as an input.

Parton distribution functions are calculated by fits to scattering data taken from both fixed-target and collider experiments, including the LHC. Because these structure functions are defined perturbatively in the strong force, these calculations are performed to varying orders in α_s . Several different collaborations, including NNPDF [166] and MMHT [167], perform these calculations using different methods at LO, NLO, and even NNLO precision. Combinations of these PDF sets and recommendations for which to use are also prepared by the PDF4LHC working group [168]. The choice of PDF set is therefore another theoretical systematic uncertainty that needs to be taken into account. Generators can perform PDF variations in which the event is evaluated using multiple functions. These variations are also saved as event weights, and the default choice can be compared to the impact of the variations to get an uncertainty.

7.1.4 Scale Variations

The event generation process depends on several scales, which can also be varied during event generation to calculate uncertainties on these scale choices. The renormalisation scale, μ_s , is the scale at which the running QCD coupling constant α_s is evaluated. Similarly, the parton distribution functions are also scale dependent, and so μ_F is the scale at which they are evaluated. A generated sample should ideally not depend on these scale choices, and so generators will perform scale

variations, similar to the PDF and parton shower variations, in which the scales μ_S and μ_F are automatically shifted up and down during event generation. This will usually be done with the scales each varied independently, as well as with both shifted together. The results of these variations are saved as event weights to see if they had any impact. The difference between the default choices and the variations can then be assigned as another theory systematic [169].

7.1.5 MC@NLO Matching and Multijet Merging

Naively, matrix elements and parton showers can be combined by taking the partons in a matrix element and running them through the parton shower. This might work for simple setups, where only a single tree-level diagram was considered as part of the matrix element phase space. However, things immediately become more complex when considering the effects of NLO QCD corrections or matrix elements with different numbers of final-state partons. For example, during the shower process, a leading order diagram with two final state partons might end up with three final state jets due to a high-energy emission. But NLO corrections to that diagram might add additional partons, and if they are included in the matrix element, this will lead to double counting. Similarly, if $pp \rightarrow jjj$ diagrams with three final state partons are included, there will also be problems, so a careful treatment of these effects is needed. “Matching” refers to combining NLO corrections in the matrix element with the parton shower, while “merging” or “multijet merging” refers to combining matrix elements with different numbers of partons together. A brief overview of some of the methods used for both matching and merging during event generation in Sherpa is given in this section [155].

MC@NLO [170] is a common prescription for matching NLO QCD calculations, and a variation on this method is implemented in Sherpa. The basic concept is that the parton shower is only performed up to some cut-off resummation scale, μ_Q [171]. Emissions above the resummation scale will be included as part of the matrix element calculation instead; note that this scale is another potential source of systematic uncertainty, and so samples can be generated with μ_Q variations as well⁷⁸. In order to make this possible, the matrix elements are divided according to their initial conditions into “H” (“hard”) and “S” (“standard” or “soft”) events, which are then treated differently and separately. In H events, the first, highest energy parton emission has already occurred due to NLO effects prior to selecting the phase space point, while this is not true in S events [170]. Another major consequence of the MC@NLO technique is that, because of the way event weights are calculated, events can be produced with *negative* weights [170]. Due to negative weight events, the

⁷⁸Unlike the μ_S and μ_F variations, which are done as event weights, in Sherpa 2.2 the μ_Q variations are produced as separate samples, which are then compared to the baseline sample.

sum of event weights can be much smaller than the total number of unweighted events generated for a given sample. Depending on the exact fraction of negative weight events in a given configuration, this can increase the total number of events that need to be generated to reduce uncertainties due to Monte Carlo statistics.

Several methods have also been implemented for multijet merging, in which matrix elements with different numbers of partons can be combined. One method implemented in SHERPA is MEPS⁷⁹ [172]; this method was first developed for LO matrix elements but has since been combined with MC@NLO to create a “MEPS@NLO” technique [173]. The merging techniques involve determining whether or not a given parton emission should be part of the matrix element—therefore increasing the matrix element multiplicity from n partons to $n + 1$ —or part of the parton shower. In the MEPS@NLO implementation in Sherpa, this is done by “clustering” the partons, using a modified variant of the k_t jet clustering algorithm described in Section 3.4 [87]. For a pair of partons i and j , a measure Q_{ij} known as the jet criterion is defined below in Equation 7.1 [173]:

$$Q_{ij}^2 = 2p_i p_j \min_{k \neq i,j} \frac{2}{C_{i,j}^k + C_{j,i}^k} \quad (7.1)$$

The minimum here is evaluated over all possible colored particles in the event, which can serve as a spectator parton k . The definition of $C_{i,j}^k$ depends on whether or not the parton i is a final-state parton or an initial-state parton. In the final state case, the assumption is that the two partons i and j were produced from some single parton ij , and $C_{i,j}^k$ is defined in Equation 7.2. Whereas if i is an initial state parton, then the assumption is that a process $i \rightarrow (ij)j$ is being modelled instead, and $C_{i,j}^k = C_{(ij),j}^k$ [173].

$$C_{i,j}^k = \frac{p_i p_k}{(p_i + p_k) p_j} \quad (7.2)$$

The measure Q_{ij}^2 is evaluated for all pairs of partons in the event. The minimum value of Q_{ij} is then found, and the pair of partons i, j that gives rise to it clustered together. If this minimum value is below a threshold Q_{cut} , this particular emission is considered low enough energy that it should be modelled via the parton shower. In this case, the process continues until the minimum is above Q_{cut} , and the remaining partons will be part of the matrix element [173]. This jet criterion can then be used to separate the matrix element and parton shower phase space, allowing for higher multiplicity matrix elements to be merged together. Note that the choice of Q_{cut} is another scale that may have

⁷⁹Short for “Matrix Element + Parton Shower”.

a systematic effect⁸⁰. If the matrix element and parton shower calculations are mostly compatible with each other, changing the threshold should not have a major impact on the generated dataset. To assess this, variation samples are produced with the merging scale shifted up or down in order to compare with the nominal value and compute an additional theory systematic.

7.1.6 Simulation and Pileup Reweighting

The rest of this section has focused on event generation, which produces truth-level events. Monte Carlo datasets also need to undergo a simulation of the ATLAS reconstruction procedure in order to be compared with real data. A simulation of the ATLAS detector has been implemented in the Geant 4 framework, which models the interactions of particles with matter [175]. The truth level particles from each event are passed through simulations of the tracker, calorimeter, and muon spectrometer, at which point the output data can be passed directly through the standard ATLAS reconstruction framework. As part of this process, a pileup reweighting procedure needs to be performed. Events are produced with a fixed number of proton-proton interactions, μ , but the pile-up varies over the course of the data-taking period. Pile-up events are simulated using the generator Pythia [165] and added to the samples during simulation and reconstruction, and a reweighting function calculated from both data and MC is used to reweight the pile-up as a function of μ .

7.2 Overview of Samples

Using the methods outlined above, signal and background Monte Carlo samples were produced for use in the VBF+MET analysis. This section steps through each of the samples and summarizes the details of how they were generated, listing the generator software used and parameters that were set. In addition, a brief overview of some of the theoretical systematic uncertainties associated with some of these samples is given. As noted above, MC generation is a complex topic, and so more detail about how the components of each generator work can be found in the referenced papers.

7.2.1 Signal Monte Carlo

The basics of the signal Monte Carlo was already introduced in Section 6.5: we generate samples with VBF, ggF, and VH Higgs bosons that decay to invisible final states, which is modelled as the $ZZ^* \rightarrow \nu\bar{\nu}\nu\bar{\nu}$ process. Having explained in more detail how MC production works above, we can

⁸⁰This scale is sometimes known as the “CKKW scale”, as one of the first merging algorithms implemented using these methods was called CKKW [174].

now give more technical details about how this signal MC is generated. The matrix elements for the invisible Higgs signal samples are produced using the tools and input parameters listed below, all of which employ variations on the Powheg method [176]. For all three samples, Pythia 8 is [165] is used to perform the parton shower.

- VBF samples are generated using Powheg 19.2.5.5 [177] at NLO in the strong force, α_s . NLO parton distribution functions from the PDF4LHC15 set [168] are used. As noted in Section 6.5, the program HAWK is used to provide NLO electroweak corrections to this sample [153].
- ggF samples are generated using Powheg NNLOPS at NNLO [178], and using a corresponding NNLO PDF set from PDF4LHC15.
- VH samples are generated using PowhegBox v2 [179] at NLO, and the PDF4LHC15 PDF set is used.

Theoretical systematic uncertainties on these samples include parton shower, PDF, and scale variations, as explained above in Section 7.1. For the VBF signal samples, the following uncertainties are considered:

- Scale variations on the choices of the factorization (μ_F) and renormalisation (μ_R) scales. While these variations can be performed automatically by Pythia, we use a more thorough calculation developed by the LHC Higgs working group, who provide scale uncertainties calculated in several m_{jj} and Higgs p_T bins. These uncertainties are approximately 1% to 3%.
- PDF uncertainties were calculated by considering 31 different choices of the PDF set, and saving each choice as an event weight during generation. The m_{jj} of the event is then calculated for each variation, and this distribution compared with the default value in order to give an uncertainty. These uncertainties are approximately 1% to 2%.
- Parton shower uncertainty: calculated by showering additional VBF invisible Higgs events with Herwig 7 and comparing the Herwig and Powheg/Pythia samples in a minimal VBF-like truth selection⁸¹. The uncertainty is calculated for each m_{jj} and $\Delta\phi_{jj}$ bin from the ratio of the two generators, and are approximately 2% to 4%.

⁸¹Specifically, requiring the leading truth jets– determined by running anti- k_t with $R = 0.4$ – to have $p_T(j_1) > 80 \text{ GeV}$ and $p_T(j_2) > 50 \text{ GeV}$. Additionally, the event is required to have $m_{jj} > 800 \text{ GeV}$, $\Delta\phi_{jj} < 2.5$, $\Delta\eta_{jj} > 3.0$, and $E_T^{\text{miss}} > 160 \text{ GeV}$.

- An additional source of uncertainty comes from the jet veto applied in the signal region. The scale variation uncertainties are determined inclusively, without the jet veto applied, and then used in $N_{\text{jets}} = 2$ or $3 \leq N_{\text{jets}} \leq 4$ bins. The effect of the jet veto has been found to potentially change the scale variation uncertainties, and so another systematic uncertainty is applied using a procedure known as the Stewart-Tackmann method to cover the difference [180]. In the Stewart-Tackmann method, the uncertainty is determined separately with and without the veto applied in order to quantify the veto’s impact. For the VBF signal, this uncertainty is roughly 3%.

The same set of uncertainties is calculated for the ggF Higgs to Invisible sample. Note that for the ggF samples, another NNLO generator was not available to vary the parton shower, and so an internal parton shower uncertainty weight calculated by Pythia is used instead. The dominant uncertainty on the gluon-gluon fusion samples is the Stewart-Tackmann jet veto uncertainty, on the order of 45%. The ggF signal is only about 10% of the total signal, so this should not have a major impact on the result. Since the VH signal is even less of the total signal, at approximately 1%, theory systematics on the VH will have even less impact and are not calculated.

7.2.2 V+Jets Monte Carlo

A “V+Jets” event contains a single weak boson, either a W or a Z , produced in association with one or more jets, and are so named because both weak bosons are vector-like particles. The weak boson can then decay either hadronically (to quarks) or leptonically (to charged leptons and/or neutrinos). As noted in Section 6.5.3, V+Jets events where the weak boson decays leptonically ($Z \rightarrow \nu\nu$, $W \rightarrow l\nu$) are the main background of this analysis. If the weak boson is produced in association with jets that appear VBF-like, and if the boson decays invisibly (either because it is a Z that produces two neutrinos, or because it is a W and the visible lepton is not reconstructed) it can be very difficult to distinguish these processes from the VBF invisible Higgs signal. Therefore, careful attention is required when modelling these backgrounds, and it is important that Monte Carlo simulations of these processes be produced with as high an accuracy as possible. As a result, most of the rest of this chapter is focused on the generation of these V+Jets samples.

There are a number of Feynman diagrams in which the interaction of two quarks can produce a weak boson. These diagrams can be classified by counting the number of QCD/strong vertices or electroweak vertices. At leading order, those that are proportional to $\alpha_s^2\alpha^2$ are classified as “QCD” V+jets diagrams, meaning that two quarks fuse together to create a weak boson, which

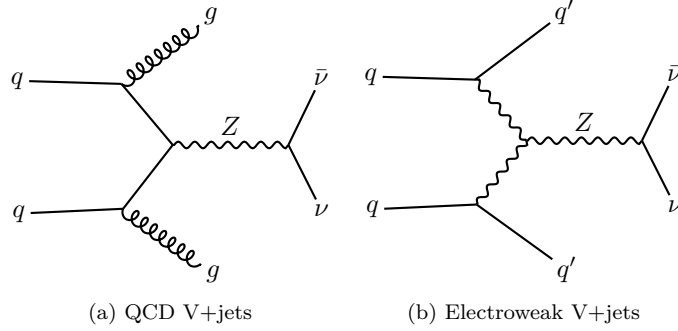


Figure 7.1: Example QCD (a) and electroweak (b) Feynman diagrams for the $Z \rightarrow \nu\nu$ plus two jet processes at leading order; the QCD diagram has two QCD and two electroweak vertices and is of order $\alpha_s^2\alpha^2$, while the electroweak diagram has four electroweak vertices, and is of order α^4 . Diagrams like these, and similar versions with W bosons instead of Z bosons, represent the main background to invisible Higgs searches in the VBF channel and are collectively referred to as “V+Jets”.

then decays leptonically. On the other hand, the leading order “electroweak” V+Jets diagrams contain four electroweak vertices: this can include the vector boson fusion process, except that instead of producing a scalar Higgs boson, the two vector bosons fuse together to create another vector boson. Examples of both diagrams are shown above in Figure 7.1. While the two processes are difficult to distinguish experimentally, the QCD and electroweak V+Jets Monte Carlo samples are produced separately using different generators. In addition, there is a third “interference” term from the mixing of the QCD and electroweak diagrams, of order $\alpha_s\alpha^3$, that also needs to be considered, along with an “interference” diagram that is proportional to $\alpha^s\alpha^3$.

The Feynman diagrams in Figure 7.1 only include leading order terms, where the final state contains exactly two jets and one weak boson. However, we can of course modify these diagrams by adding additional vertices and decays; a quark might emit a gluon or a photon as final state radiation. Figure 7.2 visualizes the various next to leading order corrections to the LO terms shown above as powers of α and α_s . Given the importance of the V+Jets backgrounds, and the fact that these NLO corrections are known to not be small, the V+Jets MC is produced with NLO corrections applied to the extent it is technically feasible.

The QCD V+Jets samples were produced using version 2.2 of the Sherpa generator [155]. As noted above, unlike other generators Sherpa performs both the matrix element calculation (using the Comix [159] and OpenLoops [181] libraries) and parton shower (using an internal implementation based on the ME+PS@NLO prescription) together. These V+Jets samples are produced using the NNPDF3.0 NNLO PDF set [166], and contain matrix elements with up to two partons at

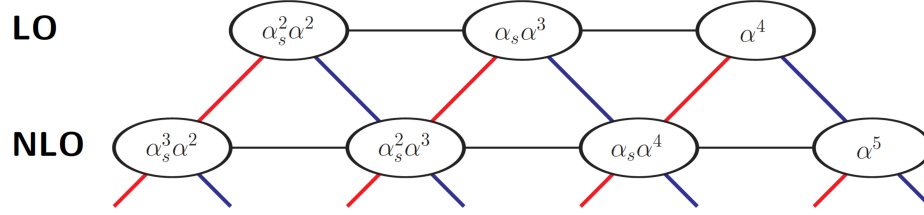


Figure 7.2: Illustration that shows the different **LO** and **NLO** matrix elements which comprise the V+Jets background process (specifically, a Z or W plus at least two jets). At leading order, the $\alpha_s^2 \alpha^2$ and α^4 are the two main classes of diagrams— examples of these can be seen in Figure 7.1— and there is also an interference term, $\alpha_s^3 \alpha$. NLO matrix elements can be written for each of these terms by adding either one additional QCD vertex or one additional electroweak vertex, as shown in the figure.

NLO ($\alpha_s^3 \alpha^2$), plus 3- and 4- parton matrix elements at LO ($\alpha_s^2 \alpha^2$). As shown in Figure 6.7, these samples are the dominant component of the V+Jets background at low- m_{jj} , and the cross section falls off considerably at higher values of m_{jj} , where the electroweak V+Jets becomes dominant. Unfortunately, because the VBF+MET analysis is most sensitive to the invisible Higgs signal at high m_{jj} , the small QCD V+Jets contribution nevertheless needs to be generated with high statistics, as otherwise the background estimation will be statistically limited in the most sensitive bins of the signal region. This was a major limiting factor in the previous 36.1 fb^{-1} version of the analysis.

Unfortunately, the QCD V+Jets samples are expensive to produce at NLO, with Sherpa 2.2 taking approximately two minutes to generate a single event. The cross section for $Z(\nu\nu)$ +jets at $m_{jj} > 1000 \text{ GeV}$ is roughly a factor of 50 smaller than the inclusive cross section; this means that generating a single extra high- m_{jj} event would require generating around 50 unnecessary low- m_{jj} events, taking nearly 2 hours total. As this would be a rather excessive waste of computational resources, an alternate procedure was developed involving matrix element filtering. By calculating a “parton-level” m_{jj} from the pre-shower matrix element partons, the QCD samples can be split into low- and high- m_{jj} slices. The statistics of the high- m_{jj} slice can then be artificially increased without also having to increase the low- m_{jj} slice. This procedure was ultimately used for the 139 fb^{-1} analysis; more information about the development and validation of the filter can be found below in Section 7.3.

Two different sets of electroweak V+Jets samples were generated. In the preliminary 139 fb^{-1} analysis, a Sherpa 2.2 sample was also used, however unlike the QCD V+Jets, it only contained leading order (α^4) matrix elements with up to four partons. Technical issues with Sherpa prevented the generation of a NLO electroweak sample, so this sample was reweighted to NLO using a sam-

ple produced using the Herwig generator [182]. For the final 139fb^{-1} analysis, this reweighting procedure was abandoned, and a NLO sample produced using Herwig 7.2.1 was used directly instead [183]. This sample contained matrix elements that were NLO in up to one final state parton ($\alpha_s\alpha^4$), and calculated using Matchbox and the VBFNLO library [184]. The parton shower was performed using the Herwig angular-order parton shower model, and the MMHT2014 NLO PDF [167] was used. More details about the generation of the electroweak V+Jets samples and some of the technical issues encountered can be found in Section 7.5.

Theory systematics from scale, PDF, and parton shower variations were determined for both the QCD and electroweak V+Jets samples. Some detailed results can be found in Section 8.1.3. Note that the Herwig electroweak V+Jets samples do not have PDF variations, and so these variations were taken from the LO Sherpa 2.2 samples and reweighted (using the same procedure as done in the preliminary analysis).

A dedicated sample for the $\alpha_s\alpha^3$ interference term was also produced. The generator package MadGraph 5 [160] was used to calculate leading order matrix elements, and Pythia 8 [165] was used to run the parton shower and hadronization. The PDF4LHC15 PDF set was used for this sample. Studies found that the overall contribution from the interference term was negligible in the analysis signal region, and therefore it was not used further as part of the analysis background estimation.

7.2.3 Other Backgrounds

The remaining backgrounds for which Monte Carlo samples are generated include multi-boson (both QCD and electroweak “VV+jets”) samples, top processes (single- t , $t\bar{t}$, etc.), and QCD multijet events. These backgrounds are much less significant than the single V+jets processes described above, and with the exception of the QCD multijet events, less sophisticated techniques are required to model them⁸². Since these backgrounds are less important, only a brief summary of the technical details involved in the generation of these samples is presented below:

- QCD multi-boson samples are produced using Sherpa 2.2 [155] at NLO in up to one parton, and LO in up to three partons. Like the QCD V+Jets samples described above, the NNPDF3.0 NNLO PDF set [166] and the Sherpa parton shower are used to simulate these. As this background is small relative to the V+Jets samples, m_{jj} slicing is not needed to increase the statistics.

⁸²The multijet background involves events with fake E_T^{miss} due to detector effects; while the background is small, a sophisticated procedure is required to estimate it. Multijet MC samples are used as an input to this procedure, as explained in Section 8.4.

- Electroweak multi-boson samples are also produced using Sherpa 2.2, but at LO instead of NLO, using the ME+PS@LO parton shower prescription and NNPDF3.0 NNLO PDF set [166].
- Top processes are generated using Powheg Box v2 at NLO, with the NNPDF3.0 NLO PDF set. Pythia 8.230 [165] is used for the parton shower. The EvtGen package is used to handle the decays of b and c quarks.
- QCD multijet events are generated and showered using Pythia 8.230 at LO, with the NNPDF2.3 LO PDF set.

7.3 Generating QCD V+Jets at High M_{jj}

As explained above in Section 7.2.2, the most sensitive region of the VBF+MET signal region is at high dijet invariant mass, m_{jj} . This means that Monte Carlo samples for background processes in this region need to have sufficient statistics, or otherwise the analysis's sensitivity will be limited. This is a particular problem for the QCD V+Jets backgrounds, as these events tend to have relatively low dijet masses. In one sense, this is a good thing, as the reason the analysis is more sensitive to the invisible Higgs signal at high m_{jj} is *because* the QCD V+Jets backgrounds are suppressed there. Unfortunately, these backgrounds still need to be modelled with low uncertainties in order to realize the sensitivity. The uncertainty on a cross section prediction from Monte Carlo scales as $1/\sqrt{N}$, where N is the total number of events generated⁸³. This follows from Poisson statistics, as the uncertainty on the number of counted events is simply $\sqrt{N}$. The number of counted events is normalized to σ/N to get the cross section, which leads to an uncertainty of $\sigma/\sqrt{N}$. This means that reducing the uncertainty on the MC prediction requires generating more events, which in this case means more QCD V+Jets events at high m_{jj} .

Table 7.1 illustrates the difference in cross section between low- and high- m_{jj} for several V+Jets processes. As noted previously, a factor of approximately 50 difference in cross section means that it is not efficient to simply generate more events inclusively, as for every extra high- m_{jj} event, roughly 50 unnecessary extra low- m_{jj} events will also be produced. Historically, the full reconstruction simulation of the ATLAS detector has been the most expensive part of producing Monte Carlo, and so one possible solution would be to generate the events inclusively in Sherpa, but then apply a truth-level m_{jj} filter before proceeding to simulation. This might be possible if the V+Jets events were

⁸³Actually, the sum of event weights— as explained above in Section 7.1.1, this does not need to be equal to the raw number of events generated.

m_{jj} Slice (GeV)	σ (pb)		
	$Z \rightarrow \nu\nu$	$Z \rightarrow \mu\mu$	$W \rightarrow \mu\nu$
0-500	46.097	8.09	61.079
500-1000	4.373	0.79	6.01
1000+	1.385	0.25	1.89

Table 7.1: Example cross sections for three V+Jets processes, $Z \rightarrow \nu\nu$, $Z \rightarrow \mu\mu$, and $W \rightarrow \mu\nu$, produced using Sherpa 2.2 with up to four matrix element partons, two of which can be NLO. A matrix-element filter was applied, requiring $140 < p_T^V < 220$ GeV for all samples; additionally, matrix-element m_{jj} filtering was used to split each process into three slices. As can be seen from these numbers, there is a massive difference between the cross section of the lowest and highest m_{jj} slices; similar differences are seen in other p_T^V slices as well.

only produced at LO, as leading order event generation in Sherpa is reasonably fast. Unfortunately, V+Jets NLO event generation in Sherpa is extremely slow, with a single event taking an average of nearly two minutes to generate on the LHC computing grid used to produce these samples, and with the bulk of that time spent on the NLO matrix element calculation. That means that truth-level filtering will not be sufficient to efficiently generate a high-stats high- m_{jj} sample.

Since the matrix element calculation is the slow part of event generation, an alternative approach would be to implement a filter at the matrix element level instead. If the phase space of matrix elements could be reduced to only select events with high m_{jj} , then a significant amount of time could be saved. This is the approach that was developed for the QCD V+Jets samples.

7.3.1 Matrix Element Filtering

Sherpa supports matrix element filtering [155], and ATLAS already uses it to slice V+Jets samples by the transverse momentum of the vector boson, p_T^V . By taking the vector sum of the two leptons in the core, matrix element process, a “parton-level” or “matrix element” p_T^V can be calculated. A cut can then be imposed on this variable when integrating and randomly sampling the phase space to select events, and so samples can be generated which only contain events with a matrix element p_T^V within some range, such as $140 < p_T^V < 220$ GeV. Then, in principle, the statistics of the $140 < p_T^V < 220$ GeV slice can be increased without needing to also generate matrix elements at other values of p_T^V and use a truth-level filter. Note that the QCD V+Jets samples used in this analysis are also sliced in p_T^V as well as m_{jj} , since for $Z \rightarrow \nu\nu$ events, the transverse momentum of the weak boson will, after reconstruction, mostly correspond to the missing transverse momentum. Since we require all events to have $E_T^{\text{miss}} > 160$ GeV, we do not need to generate events with low values of p_T^V .

There is one major caveat, however. The above statement is only true if the matrix element filter is perfectly efficient. The matrix element variables are not necessarily the same as the final truth-level variables. Depending on how the generator works, the kinematics at the matrix element level can be shifted considerably after the merging with the parton shower calculation. While the leptons in the matrix element will not actually be ran through the parton shower, the overall energy of the event can still be adjusted up or down, and the leptons *can* radiate final state radiation in the form of photons. This means that the matrix element p_T^V for a $Z \rightarrow \nu\nu$ event will be close to, but not always identical to, the “final-state” p_T^V constructed from the vector sum of the two final-state neutrinos. Even for p_T^V , then, the matrix element filter will not be perfectly efficient: there will always be some “leakage” of events out of the slice boundaries. In the best case scenario, this leakage will be relatively small. In the worst case scenario, this leakage could be quite large, and potentially even lead to discontinuities in the truth-level p_T^V distribution at the boundaries between slices.

The problems described above are significantly compounded when trying to select or filter on a variable involving the hadronic parts of a matrix element, like m_{jj} . Jets are produced during the parton shower process, and the matrix element only contains pre-parton-shower hadronic partons, which represent quarks or gluons in the core process. These partons are not physical, and are not necessarily well modelled before the shower. In principle, the highest energy truth jets in an event after showering should correspond to the highest energy partons at the matrix element level. In practice, this may not be true, depending on the details of how Sherpa implements these steps. The energy of a parton in the matrix element might change considerably before and after the shower, or a single parton might split into multiple jets. Additional quark or gluon emissions can also be added during the parton shower that can create new jets that do not correspond to a matrix element parton at all. Even defining m_{jj} at the parton level is therefore somewhat difficult.

These problems can be illustrated clearly in Figure 7.3, where an attempt to implement a matrix-element m_{jj} filter in Sherpa was tested. The parton-level m_{jj} shown here was defined as the invariant mass of the vector sum of the two highest p_T partons in the matrix element. The partons are first “clustered” using anti- k_t with $R = 0.4$ in order to combine any overlapping particles before calculating the mass⁸⁴. Test samples were produced using Sherpa 2.2 with this filter implemented in three m_{jj} slices: $0 < m_{jj} < 500$ GeV, $500 < m_{jj} < 1000$ GeV, and $m_{jj} > 1000$ GeV. The truth-level m_{jj} was then calculated by running the anti- k_t algorithm with a $R = 0.4$ over the final-state hadrons, and plotted with a minimal VBF-like selection applied: events must have at least two truth jets,

⁸⁴Note that in the vast majority of cases, this clustering has no impact, as partons rarely overlap.

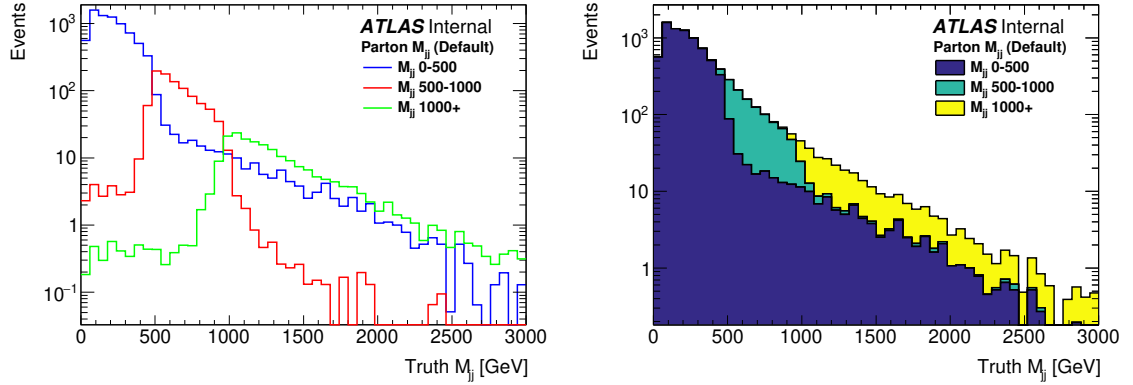


Figure 7.3: Plots showing the truth-level m_{jj} calculated from $Z \rightarrow \nu\nu$ MC generated with $140 < p_T^V < 220$ GeV and m_{jj} slicing at the matrix element level, using Sherpa 2.2. A minimal VBF-like jet selection has been applied, requiring leading and subleading jets to have $p_T > 80, 50$ GeV, respectively, and a third jet veto at 30 GeV has been imposed. The left plot shows each slice plotted separately, while the right shows them stacked together. The filter inefficiency, causing many events to be generated with truth $m_{jj} > 500$ GeV in the $0 < m_{jj} < 500$ GeV sliced sample, is clearly visible.

with the leading $p_T(j_1) > 80$ GeV and the subleading $p_T(j_2) > 50$ GeV. No additional jets with $p_T > 30$ GeV were allowed⁸⁵. As seen in the plots, while the slicing does somewhat appear to work, there is significant contamination from the lowest $0 < m_{jj} < 500$ GeV slice into the two higher slices.

This is a significant problem, because luminosity is defined as $N\sigma$, and so the samples need to be scaled by their cross sections when they are combined. This means that the uncertainties coming from the low m_{jj} sample will also be scaled by the low- m_{jj} cross section, which as shown in Table 7.1 is a factor of 50 larger than the cross section from the high- m_{jj} slice. So the uncertainty from the leakage will dominate the uncertainty from the $m_{jj} > 1500$ GeV slice, which means that the analysis sensitivity is still constrained by the statistics from the low- m_{jj} slice. In other words, this matrix element filtering has not been successful in solving the problem.

7.3.2 Emulating the Filter

Given these findings, a significant amount of work was then performed to understand the origin of the leakage, and, ideally, how to reduce it. These studies were undertaken with guidance and help from Sherpa developers and experts in and outside the ATLAS collaboration, most notably Frank Siegert and Marek Schöenherr, who provided invaluable, detailed information about how the matrix

⁸⁵The re-optimized VBF+MET analysis moved away from the strict third jet veto used in the 36.1 fb^{-1} analysis and towards a centrality veto, as described in Section 6.3; these studies, however, were launched during the reoptimization process before this was finalized.

element generation process actually works.

The initial goal of these studies was to try and emulate the matrix element filter using the generator truth record. If the matrix element partons used in the filter to calculate the parton-level m_{jj} could be identified, then it would be possible to try and understand what exactly changed between the partons and truth-level jets that leads to “leaking” events. Sherpa 2.2 does save this matrix element information in the truth record, which is stored using the generator-independent HepMC format [185]. In HepMC, particles produced via the generator are assigned a numerical “status code”, which indicates the type of particle. Unfortunately, while the HepMC format is generator independent, the meaning of these status codes are often generator dependent. In Sherpa 2.2.2 and higher, truth particles are assigned status codes according to the following convention⁸⁶, the first four of which are roughly standardized [186]:

- Status 1: stable final state particles, such as leptons and photons.
- Status 2: unstable final state particles, such as some composite hadrons (mesons or baryons).
- Status 3: matrix element particles that represent the core process.
- Status 4: incoming particles, which in this context means the two protons that interact and create the event in question.
- Status 11: intermediate particles from various stages of the event generation. Among other things, this includes the quarks and gluons produced from the parton shower before the hadronization step, where baryons and mesons are created.
- Status 20: matrix element particles, but only for MC@NLO S events.

The NLO Sherpa samples are generated using a version of the MC@NLO procedure explained above in Section 7.1, and so there are two types of events: S and H events [170]. Note that for MC@NLO S events, there are two sets of matrix element partons saved in Sherpa: those with status code 3, and those with status code 20. This is because, for a S event, the first emission of an additional parton beyond the core process is handled specially, and so gets saved as part of the status 3 events. However, this parton is not present when any matrix element filtering is applied, and so for these events, it would not be possible to emulate the filter after the fact using the status 3 partons. Instead, the partons that were actually present at the filtering stage are stored as status

⁸⁶Sherpa may sometimes use additional status codes as well; this is not meant to be an exhaustive list.

20 partons⁸⁷. For MC@NLO H events, this first emission is not treated differently, and so status 20 partons are not saved. Therefore, to emulate a parton-level filter after event generation for a NLO Sherpa sample, one should select the status 20 partons if they are present (indicating a S event), or use the status 3 if they are not (indicating a H event).

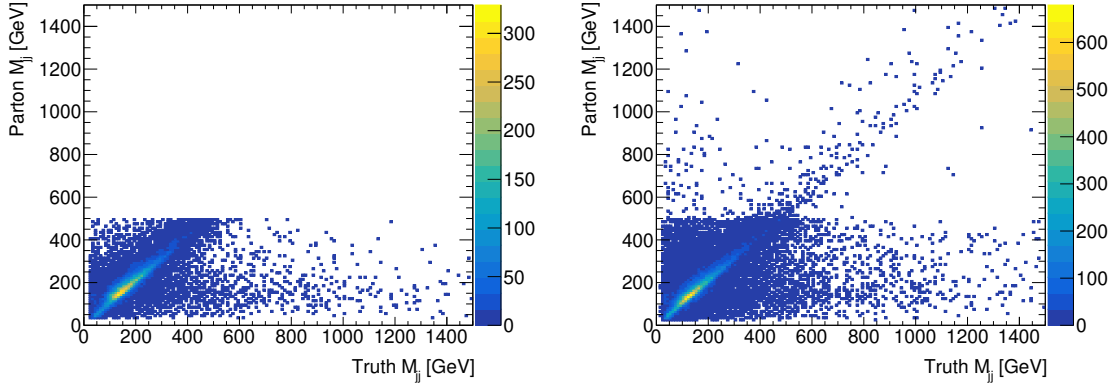


Figure 7.4: Plots showing the truth-level and matrix-element level m_{jj} plotted against each other from one of the $Z \rightarrow \nu\nu$ test samples generated with a $0 < m_{jj} < 500$ GeV filter. The left plot shows only MC@NLO [170] “S” events, i.e. those with status 20 partons in the truth record, and the right plot shows only “H” events, i.e. those that do not have status 20 partons. An attempt was made to emulate the filter and calculate a matrix element m_{jj} from the truth record saved for each event, but as shown here, this is not entirely successful for the “H” events.

This emulation procedure was then done in the $0 < m_{jj} < 500$ GeV sample, and the results are shown in Figure 7.4. Here, the “parton m_{jj} ” was calculated by selecting the right particle collection as described above, and then taking the invariant mass of the vector sum of the two outgoing partons with the highest p_T . However, as can be seen in the plots, this approach only seems to work for the S events and not the H events, as there is a small population of H events for which the parton-level m_{jj} is apparently above the 500 GeV threshold. This seems to indicate that it is not completely possible to emulate a filter for at least a small fraction of the H events. Further checks showed that these un-emulatable events are primarily, but not exclusively, negative weight events. This problem was reported to and acknowledged by the Sherpa developers, but was ignored in the following analysis since it appears to only represents a small fraction of the overall sample.

Having figured out how to (mostly) emulate the filter, several studies were then conducted. One possible explanation for the filter inefficiency might be that, in events with more than two partons,

⁸⁷This feature was only added in Sherpa 2.2.2; unfortunately, that means in older releases of Sherpa, status 20 partons are not saved. That means that it is not possible to emulate a parton-level filter in previous versions of Sherpa, which is unfortunate because version 2.2.1 was used by ATLAS for the centrally produced, unfiltered V+Jets samples. This is why studies undertaken in this section used version 2.2.4 instead.

the “wrong” partons are being used to calculate the matrix element m_{jj} . This could be because the parton shower shifts the momentum of the subleading parton down and the momentum of a third parton up, or it could be because the leading or subleading parton fragments and produces multiple jets in the parton shower. If this is part of the problem, then alternate methods of calculating the matrix element “ m_{jj} ” could potentially reduce the leakage. For instance, one alternative might be to select the combination of two or more partons that give the highest possible m_{jj} , and hopefully cause events to be more likely to appear in the higher m_{jj} slices. Another option, which is only possible when emulating the filter, is to select the pair of partons that have an invariant mass closest to the truth m_{jj} . Calculating this “best m_{jj} ” and then re-dividing the samples using it would at least reveal whether or not there was any possible combination of partons which would lead to a higher filter efficiency.

Unfortunately, neither the “maximum” or “best” parton-level m_{jj} filters were found to have a major impact on the leakage from the low m_{jj} slice. Another set of studies then looked at the truth-level jets in the leaking events to see if they match to any partons at all. For every event, ΔR matching was performed, in which the separation in $\eta - \phi$ space between each truth jet and each parton was calculated. A jet was considered to be matched if its closest parton was within a radius of $\Delta R < 0.4$ around its center. In many cases, it was found that only one of the two leading jets was matched to a parton in events with matrix element $m_{jj} < 500 \text{ GeV}$ but truth $m_{jj} > 500 \text{ GeV}$. An additional test of this for MC@NLO S events found that the first NLO emission, which is stored in the HepMC event record as a status 3 particle, frequently did match with the subleading truth jet when the other matrix element partons did not. Combined, this clearly indicates that the filtering inefficiency is due to jets created by extra emissions during the parton shower process that are not part of the matrix element at all. And since they are not part of the matrix element, they cannot be filtered over.

7.3.3 Alternate Merging Criteria

Given the above results, one potential strategy for fixing the m_{jj} filter would be to find a way to include the extra parton emissions in the matrix element calculation. In Sherpa, the boundaries of the matrix element’s phase space are set by the merging algorithm, which was explained above in Section 7.1.5. A metric is defined and used to evaluate the energy of each potential parton emission, and if that energy is below a certain threshold, the emission is removed from the matrix element. That metric, known as the “jet criterion”, is a variant of the k_t jet-finding algorithm outlined in

Section 6.3 [87] which takes into account the flavour of the partons; its functional form was given above in Section 7.1.5 [173]. After discussion with the Sherpa developers, it was suggested that the default Sherpa algorithm outlined above emissions differently than the k_t or anti- k_t jet-finding algorithms would [87]. We then decided to test the impact of generating new m_{jj} -sliced samples with these alternate merging criteria, to see if this lead to an improvement in efficiency.

An initial test was performed at leading order. Two sets of $Z \rightarrow \nu\nu$ samples with up to four leading order partons were produced using Sherpa 2.2.4, one with the k_t criterion, and one with the default criterion. These samples made use of the same m_{jj} slicing, and were split into three $0 < m_{jj} < 500$ GeV, $500 < m_{jj} < 1000$ GeV, and $m_{jj} > 1000$ GeV slices. The truth m_{jj} was then calculated after event generation and compared between the two criteria. A significant reduction in leakage from the low- m_{jj} slice was observed in the k_t sample relative to the default sample, and so another set of NLO test samples was then produced. Three sets of samples were produced up to NLO in two partons with the default, k_t , and anti- k_t merging criteria⁸⁸. The same m_{jj} slicing was used, and initially the samples were limited to $140 < p_T^V < 220$ GeV. The results for the k_t samples are shown in Figure 7.5, and compared to Figure 7.3 above we see roughly an order of magnitude reduction in the leakage from the low- m_{jj} slice. Comparable results were seen from the anti- k_t test.

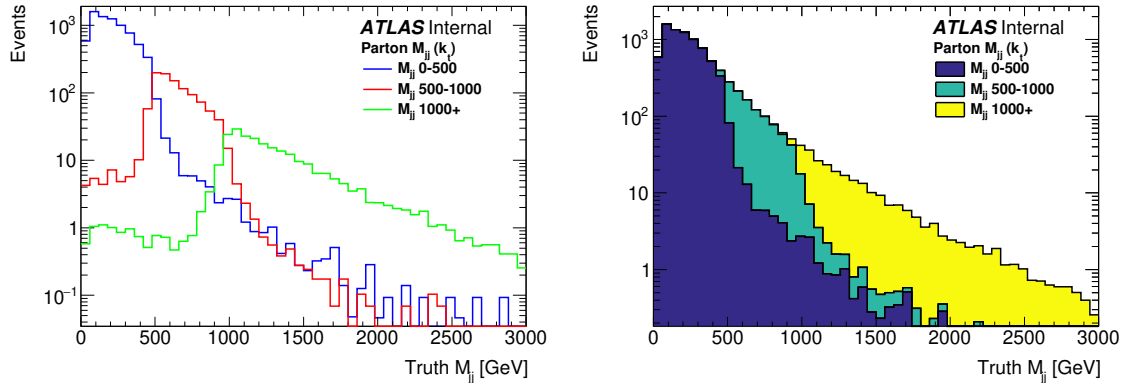


Figure 7.5: Plots showing the truth m_{jj} distributions from $Z \rightarrow \nu\nu$ samples generated using Sherpa 2.2.4 with $140 < p_T^V < 220$ GeV and a k_t merging criterion. A major improvement in matrix element m_{jj} filter efficiency can be seen compared to the samples produced with the same configuration and the default merging criterion shown above in Figure 7.3.

Table 7.2 shows the number of normalized events from each slice that have truth $m_{jj} > 1000$ GeV, for all three criteria. The exact impact of this change can be seen by comparing the parton multi-

⁸⁸For various technical reasons, the k_t and anti- k_t merging criteria behave identically at LO: the merging is much less involved when only leading order diagrams are considered. At NLO, however, the two may behave differently.

Parton m_{jj} Slice (GeV)	Truth $m_{jj} > 1000 \text{ GeV (fb}^{-1}\text{)}$		
	Default	k_t	Anti- k_t
0-500	83.62	11.66	12.64
500-1000	11.18	14.72	14.44
1000+	176.10	213.94	207.27

Table 7.2: Table comparing the number of events seen with truth $m_{jj} > 1000 \text{ GeV}$, leading, sub-leading jet $p_T > 80, 50 \text{ GeV}$ and with a third jet veto at 30 GeV across the three parton m_{jj} slices for three different choices of the jet merging criterion: default, k_t , and anti- k_t . Event yields are normalized to 1 fb^{-1} . Note that the number of events from the lowest parton m_{jj} slice is much lower in the k_t and anti- k_t merged samples compared to the default; the “leakage” into the $m_{jj} > 100 \text{ GeV}$ slice is significantly reduced.

plicity of the three samples, as shown in Figure 7.6. The default sample has noticeably more events with a single parton in the matrix element (and therefore, a parton-level $m_{jj} = 0$, which will always be in the low- m_{jj} slice), and significantly fewer with four partons, indicating the alternate merging criteria are correctly moving more emissions into the matrix element. Since the results from the k_t and anti- k_t samples were comparable, and since the k_t algorithm was considered a more “natural” merging criteria by the Sherpa developers due to the differences in how it and anti- k_t treat soft partons, the decision was made to proceed with k_t .

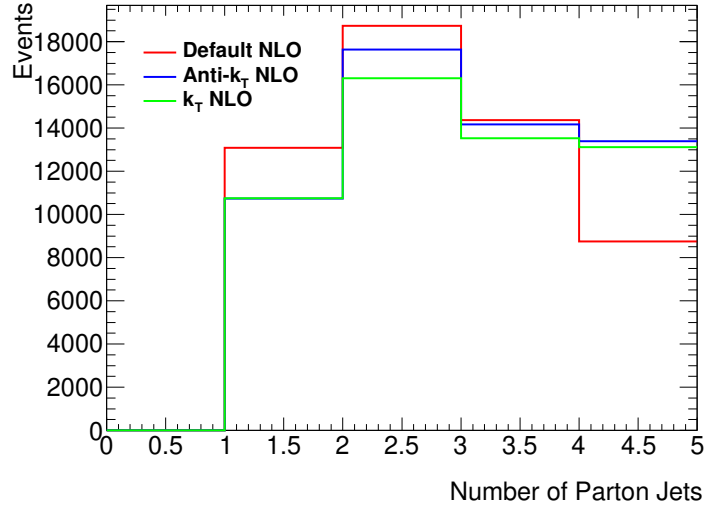


Figure 7.6: Comparison of the matrix element parton multiplicity between the default, k_t , and anti- k_t samples with $140 < p_T^V < 220 \text{ GeV}$. The change in merging criterion from default to a k_t -based algorithm causes an increase in the parton multiplicity, making it less likely the parton emission corresponding to a subleading jet won’t be in the matrix element.

Studies were then performed to validate the k_t -merged samples to ensure that there were no

unintended consequences from switching the merging criterion away from the Sherpa default. Figure 7.7 shows a comparison of the combined m_{jj} shape between default and k_t -merged samples with all three slices added together. While there appears to be a slope in the m_{jj} in the default sample relative to the k_t -merged one, this is not necessarily a problem. The Sherpa V+Jets samples are known to overpredict real collision data at high m_{jj} , and so having this distribution shifted down slightly at high m_{jj} might indicate that the k_t merged samples are actually better modelled. Other tests looked at other variables, such as $\Delta\eta_{jj}$ and $\Delta\phi_{jj}$, and no major differences were observed.

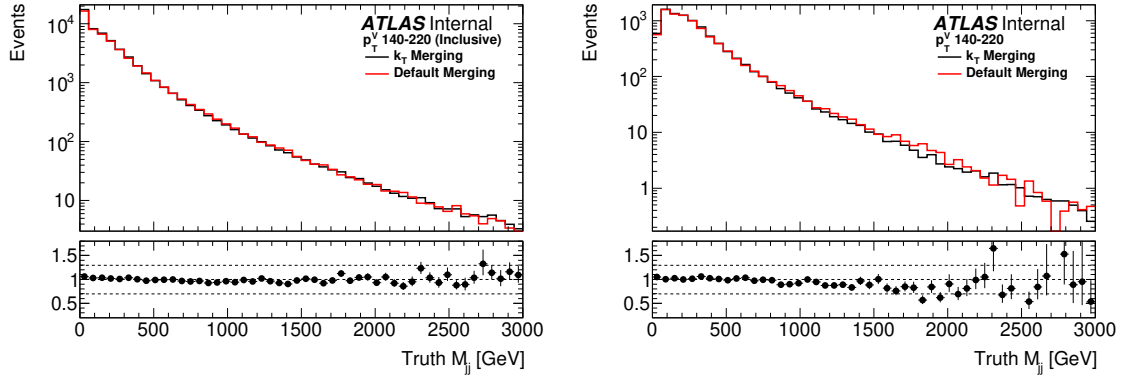


Figure 7.7: Plots comparing the truth m_{jj} between the default and k_t -merged $Z \rightarrow \nu\nu$ samples generated with Sherpa 2.2, $140 < p_T^V < 220$ GeV, and matrix element m_{jj} slicing. The left plot shows the distributions without any cuts applied, while the right shows the comparison with a VBF jet cut selection.

These initial tests were only performed in a single p_T^V slice. Once the k_t merging was shown to work in the $140 < p_T^V < 220$ GeV range, additional NLO $Z \rightarrow \nu\nu$ samples were generated in nearby slices, such as $220 < p_T^V < 280$ GeV and $280 < p_T^V < 350$ GeV. Comparing the effects of m_{jj} slicing for these higher p_T^V samples showed a similar difference between the default and k_t merging criteria, although the leakage using the default criterion was not as bad at increased values of p_T^V to begin with. A test was then done to make sure that the different slices would combine smoothly, with no discontinuities or boundary effects. Figure 7.8 shows combined p_T^V distributions from both sets of samples. As shown here, not only do the two distributions seem to agree, but no significant discontinuities are observed at the slicing boundary of $p_T^V = 220$ GeV⁸⁹.

Finally, a test was done in which the m_{jj} sliced, k_t -merged samples were combined with $Z \rightarrow \nu\nu$ MC generated without m_{jj} slicing in the region $p_T^V > 500$ GeV. These events have a minimal impact

⁸⁹These plots also illustrate the point made above: that even the truth-level p_T^V does not fully agree with the matrix element value. If it did, there would be no events outside the slicing boundaries $140 < p_T^V < 500$ GeV, but the presence of these events shows that even this comparatively well-behaved filter is not perfectly efficient.

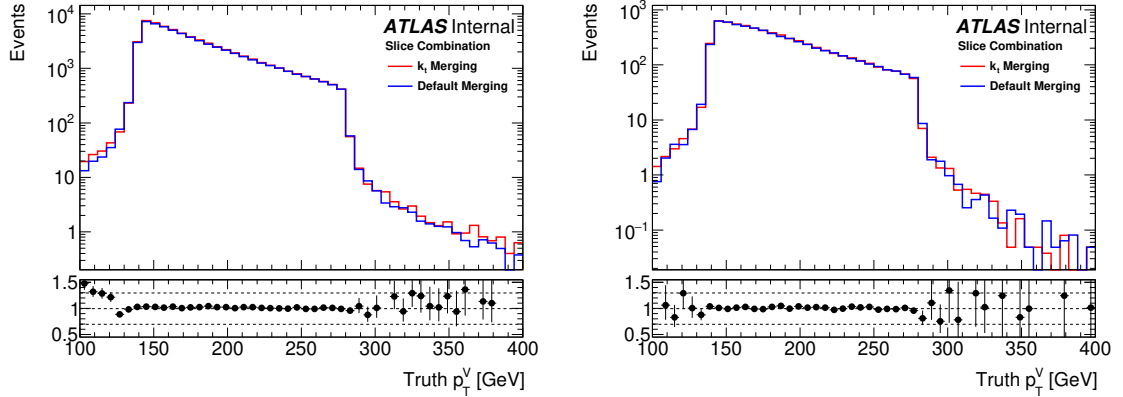


Figure 7.8: Plots comparing the truth p_T^V between the default and k_t -merged $Z \rightarrow \nu\nu$ samples generated with Sherpa 2.2, $140 < p_T^V < 280$ GeV, and matrix element m_{jj} slicing. The left plot shows the distributions without any cuts applied, while the right shows the comparison with a VBF jet cut selection. Good agreement is seen across the slice boundary of $p_T^V = 220$ GeV.

on the analysis's sensitivity, and the cross section of the higher p_T^V slices is quite low, so m_{jj} slicing is not needed to increase statistics here. The combined p_T^V distribution also appeared smooth across this slicing boundary.

7.3.4 Optimizing M_{jj} Slicing

It appears that k_t merging significantly improves the matrix-element m_{jj} filter for these V+Jets samples. The next question is whether this improvement is good enough, and if so, how much Monte Carlo needs to be generated in order to ensure the statistical uncertainties are small? Some optimization calculations were performed to try and answer this question.

As explained above, the uncertainty on the predicted cross section from a given Monte Carlo slice is $\sigma/\sqrt{N}$, where N is the number of events generated and σ the cross section. It therefore follows that the MC uncertainty in a given event selection should be $\sigma\sqrt{\frac{f}{N}}$, where f is the efficiency of the cuts that define the selection. If the MC sample is split into several slices, each with N_i events generated for a cross section σ_i , then the total uncertainty $\delta\sigma$ is the square root of the sum of $\sigma_i^2 \frac{f_i}{N_i}$ for each slice. This can also be written as a fractional uncertainty by taking the ratio of the total uncertainty to the total cross section prediction, $\sum_i \sigma_i f_i$. For this sample, there are three m_{jj} slices, and so the uncertainty can be written as a two-dimensional function if we assume that the total number of events that can be generated is held fixed. This gives the equation below for the fractional uncertainty, written in terms of N_1 and N_2 as the sizes of the two low- m_{jj} slices, and the parameter $Q = N_1 + N_2 + N_3$, the total number of events to generate.

$$\frac{\delta N}{N} = \frac{1}{\sum_j \sigma_j f_j} \sqrt{\frac{\sigma_1^2 f_1}{N_1} + \frac{\sigma_2^2 f_2}{N_2} + \frac{\sigma_3^2 f_3}{Q - N_1 - N_2}} \quad (7.3)$$

$$N_i^{\text{optimal}} = \frac{Q \sigma_i \sqrt{f_i}}{\sum_j \sigma_j \sqrt{f_j}} \quad (7.4)$$

Equation 7.3 can be minimized to find the optimal number of events N_i^{optimal} to generate for any of the three slices, as shown below in Equation 7.4. The efficiency f_i can be calculated for each slice from the studies shown above, and the parameter Q is assumed to be a constant. This calculation was done for both the k_t and default m_{jj} sliced test samples with $140 < p_T^V < 220$ GeV that were generated above. The fractional uncertainty was plotted as a function of N_1 (the $0 < m_{jj} < 500$ GeV slice) and N_2 (the $500 < m_{jj} < 1000$ GeV) assuming $Q = 3$ million events in Figure 7.9. As these plots show, switching from the default samples to the k_t samples does shift the optimal fraction, with less events required in the low slice, but only somewhat. Even with the minimal amount of leakage in the k_t -merged samples, N_1 and N_3 evidently need to be roughly equal to minimize the uncertainty. That said, the overall uncertainty does appear to be lower when using k_t merging.

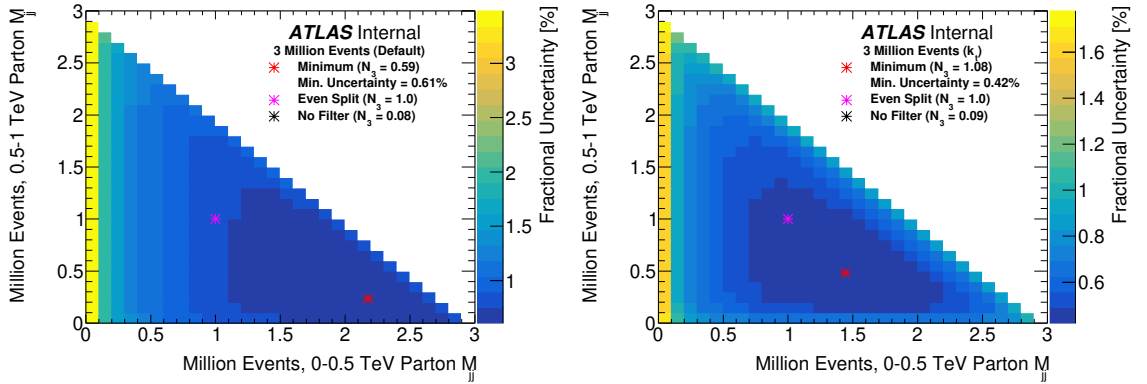


Figure 7.9: Plots of Equation 7.3 for both the default (left) and k_t -merged (right) $Z \rightarrow \nu\nu$ samples generated with m_{jj} slicing and $140 < p_T^V < 220$ GeV. The filter efficiency f_i was determined by applying a minimal VBF truth jet selection. The color indicates the relative size of the fractional uncertainty from varying the size of the low- m_{jj} sample (on the x -axis) and the medium- m_{jj} sample (on the y -axis). The plots were made assuming 3 million events could be generated total for each sample, so the size of the $m_{jj} > 1$ TeV sample is fixed by $3M - x - y$. The optimal size of this sample is shown by the location of the red dot.

Additional tests were then done to see if an alternate slicing configuration might lead to better results. Three ideas were tested:

- First, slicing in alternate variables in addition to m_{jj} , such as $\Delta\phi_{jj}$. The full run 2 VBF+MET analysis was reoptimized at this point to include $\Delta\phi_{jj}$ binning, as described in Section 6.3.4. This seemed to have little impact on the overall efficiency or on the uncertainty.
- Second, changing the m_{jj} slice boundaries, either by shifting the thresholds to something like $0 < m_{jj} < 400 \text{ GeV}$ and $400 < m_{jj} < 900 \text{ GeV}$ or by dividing the sample into more than three slices. A five-slice configuration with additional high- m_{jj} slices was tested, for instance. This also appeared to have little impact.
- Third, by moving towards the use of the “maximum m_{jj} ” parton-level filter suggested above in Section 7.3.2, now that more partons are available in the matrix element to choose from. This did have an impact, shifting a large number of events from the lowest slice into the highest slice to the extent that “reverse” leakage became a problem: after testing this, the $m_{jj} > 1000 \text{ GeV}$ slice contained a very large number of events with truth $m_{jj} < 1000 \text{ GeV}$.

Due to the potential complexity and unclear impact of the maximum m_{jj} option, it was decided to not use it and make no changes to the slicing setup for the full run 2 analysis. Samples were prepared for production using Sherpa 2.2.7 with k_t merging for $Z \rightarrow \nu\nu$, as well as $W \rightarrow l\nu$ and $Z \rightarrow ll$ with all three charged lepton flavours⁹⁰. Events were generated in four p_T^V slices, 100 – 140, 140 – 220, 220 – 280, and 280 – 500, each split in three m_{jj} slices. Each p_T^V slice was split roughly 40%-20%-40% into the three m_{jj} sub-samples using the optimization calculations shown above. A total of 55 million events was generated across all of the processes and samples, as it was determined that this was approximately the size of a request that could be supported by available computing resources.

7.3.5 Limitations and Future Developments

These 55 million events were used to model the QCD V+Jets backgrounds in the preliminary version of the 139 fb^{-1} analysis. Unfortunately, the statistical uncertainties from Monte Carlo were still found to be relatively large even after introducing them⁹¹. In part, this was traced due to reconstruction and pile-up effects. The optimization studies presented in this section were done entirely at truth level, but additional complications arise when reconstructing events. Despite the

⁹⁰2.2.4 had been the latest release when the studies started, but Sherpa developers recommended we move to 2.2.7 to pick up various bugfixes and improvements before beginning production. Validation was done to confirm that no significant difference in filter efficiency was seen between the two minor releases of the generator.

⁹¹Appendix C contains the results from the preliminary analysis, where the relative impact of each source of uncertainty on the final limit on $\mathcal{B}_H \rightarrow \text{inv.}$ can be seen.

cuts introduced to reduce the impact of pile-up, it is still possible for a pile-up jet to be confused for one of the two leading jets that comprise m_{jj} , and of course reconstruction can further change the energies and momenta of the truth jets. Also, even ignoring reconstruction effects, the truth-level selection used for these studies did not fully match the signal region definition, in part because the analysis was still being reoptimized while this work was underway. To further mitigate the Monte Carlo statistical uncertainties, we received permission from ATLAS management to request an extension of these samples for the final version of the 139fb^{-1} analysis. An additional 120 million events were generated using the same m_{jj} -sliced setups, increasing the size of the QCD V+Jets dataset by a factor of 3. With this extension, Monte Carlo uncertainties were considerably reduced, as will be seen later in the results shown in Chapter 9.

For future iterations of this analysis for run 3 and beyond, it may make sense to continue using the k_t -merged, m_{jj} slicing approach, perhaps using the “maximum m_{jj} ” filtering to try and further improve the modelling. Other improvements to Sherpa might make it unnecessary, though, as work is being done upstream to try and improve both the performance and modelling. If events could be generated faster, there would be no need to introduce slicing. Alternatively, a new method of increasing statistics that’s been developed is “phase space biasing”, in which instead of slicing, a smooth, continuous function is applied to reweight the events when calculating matrix elements to artificially increase statistics without the need for slicing. This biasing approach has several advantages, such as removing the potential for discontinuities at slice boundaries when recombining samples. Another potential improvement might be to use another generator altogether, provided it can also be used to generate NLO V+Jets Monte Carlo. The CMS version of this analysis, for instance, uses MadGraph instead of Sherpa [187]. Future analyzers should pay close attention to developments in this area and consider the best combination of tools for the next set of samples.

After generating the samples, some additional modelling problems were discovered with them. A small section of phase space was found to have been accidentally excluded, as can be seen clearly in Figure 7.10. This plot compares a sample produced inclusively with no m_{jj} slicing to the combination of samples generated with slicing. After appropriate normalization, a clear difference in the ratio can be seen at high m_{jj} . This issue appears to stem from the difference between Sherpa’s internal definition of what partons are, and the anti- k_t algorithm. Because anti- k_t is used to cluster the partons before computing m_{jj} , an event that had two partons before clustering but does not have two partons *after* clustering will be rejected— even though in principle, it should have been accepted into the lowest m_{jj} slice. In practice, this seems to have a small but non-zero impact, as shown in the plot. This missing cross section should be consistent between the $Z \rightarrow \nu\nu$, $Z \rightarrow ll$, and $W \rightarrow l\nu$

samples. That means that the data-driven procedures used for background estimation presented in Section 8.1 should be able to correct this type of mismodelling, so it is not a major issue.

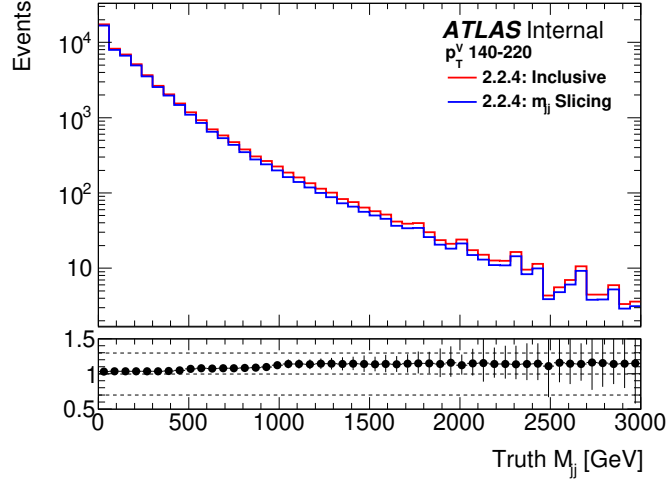


Figure 7.10: Plot comparing a m_{jj} -sliced $Z \rightarrow \nu\nu$ test sample with an inclusive $Z \rightarrow \nu\nu$ test sample, both produced with the same k_t merging criterion. The three m_{jj} slices were weighted by their cross section and recombined and the truth m_{jj} distribution compared with the inclusive. A clear difference can be seen in the ratio at high m_{jj} due to a small amount of missing cross section in the sliced version of the sample.

Another problem was discovered that does involve differences between the Z and W processes; this is presented in Section 7.4 below.

7.4 Jet Veto Efficiency in QCD V+Jets

The V+Jets samples described above were generated and used in the 139 fb^{-1} analysis to estimate both Z and W backgrounds. While developing the analysis, however, a question arose: are the W+Jets and Z+Jets predictions compatible with each other? If so, it might be possible to use one of the V+Jets processes to help model the other⁹². The two weak bosons are quite similar (although not identical), and in principle we would expect to see minimal differences in variables like the dijet invariant mass m_{jj} or the azimuthal separation $\Delta\phi_{jj}$ between the two processes. As a first step to begin answering this question, studies were launched using the V+Jets Monte Carlo to try and identify any potential differences at truth level between the $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ samples. If there were any differences found between the samples, we would then need to understand whether these

⁹²More information on this work can be found in the next chapter, in Section 8.2. Since this study is a truth-level exploration of the differences between the samples generated in the previous section, it is also included in this chapter.

are real, and arise due to electroweak physics, or whether they are a simulation artifact due to a bug somewhere in the event generation pipeline.

One variable of particular interest is the third jet veto efficiency, defined as the percentage of events with exactly two jets with transverse momenta $p_T > 25$ GeV, relative to the number of events with at least two 25 GeV jets. As Section 6.3 explained, the core vector boson fusion process involves exactly two jets, and while some of the signal region bins allow for events to have additional forward jets, most do not. While the core W and Z Feynman diagrams included in the matrix element calculation should be similar, the number of jets in a Monte Carlo event is not just determined by the matrix element but also by the parton shower and hadronization processes. Therefore, it is important to validate that this part of event generation behaves consistently between W and Z when comparing the two processes.

To study this question, the k_t -merged, m_{jj} -sliced samples generated above were used. A minimal VBF-like event selection was defined, using truth jet cuts:

- Events must have at least two truth jets with $p_T > 25$ GeV, as determined using the anti- k_t algorithm with $R = 0.4$.
- The leading and subleading truth jets must have $p_T(j_1) > 80$ GeV and $p_T(j_2) > 50$ GeV, respectively.
- The invariant mass of the dijet system must be $m_{jj} > 800$ GeV.
- The two leading truth jets must not be back-to-back, with $|\Delta\phi_{jj}| < 2.0$.
- The two leading truth jets must be well separated in pseudorapidity, with $\Delta\eta_{jj} > 3.8$.

In addition to the above cuts, a requirement on the truth-level transverse momentum of the weak boson, p_T^V , is used as a proxy for the reconstructed missing transverse momentum requirement. As explained above, p_T^V is calculated from the two leading final-state truth leptons, because the matrix element leptons may change considerably during the event generation process. For $Z \rightarrow \nu\nu$, this is simply the sum of the two leading truth neutrinos. For $W \rightarrow l\nu$, this is now the sum of the leading charged lepton of the appropriate flavour (an electron for $W \rightarrow e\nu$, a muon for $W \rightarrow \mu\nu$) with the corresponding leading neutrino. Note that the charged lepton can emit photons as final state radiation during the event generation process, however, and so it must be “dressed” with any photons that are within $\Delta R(l, \gamma) < 0.1$ of the lepton. Studies were then performed with a variety of p_T^V cuts, including 160 GeV and 200 GeV to match the E_T^{miss} cuts used in the analysis.

7.4.1 Z/W Comparison

With the above selection applied, the jet veto efficiency was then computed. Figure 7.11 shows the jet veto efficiency for $Z \rightarrow \nu\nu$ compared to $W \rightarrow l\nu$ (with $e\nu$ and $\mu\nu$ samples included), with both $p_T^V > 200$ GeV and $p_T^V > 160$ GeV applied. As can be seen from these plots, while the jet veto efficiency appears to agree quite well in the first few m_{jj} bins, there is a clear slope in the Z/W ratio at high m_{jj} . In the highest $m_{jj} > 3000$ GeV bin, there is nearly a 30% difference between the veto efficiency in the two processes. Also, the jet veto efficiency appears to decline overall as a function of m_{jj} , from about 30% at $m_{jj} = 800$ GeV to only around 5-10% in the last bin. There is not an immediately obvious electroweak physics explanation of why this discrepancy between the two processes would occur. This means that, if we were to use the $W \rightarrow l\nu$ MC to try and predict the $Z \rightarrow \nu\nu$ background, a 30% systematic uncertainty would need to be assigned to cover this difference in the highest m_{jj} bins. As these are the most sensitive bins in the analysis, as shown in Figure 6.6, this would be unfortunate.

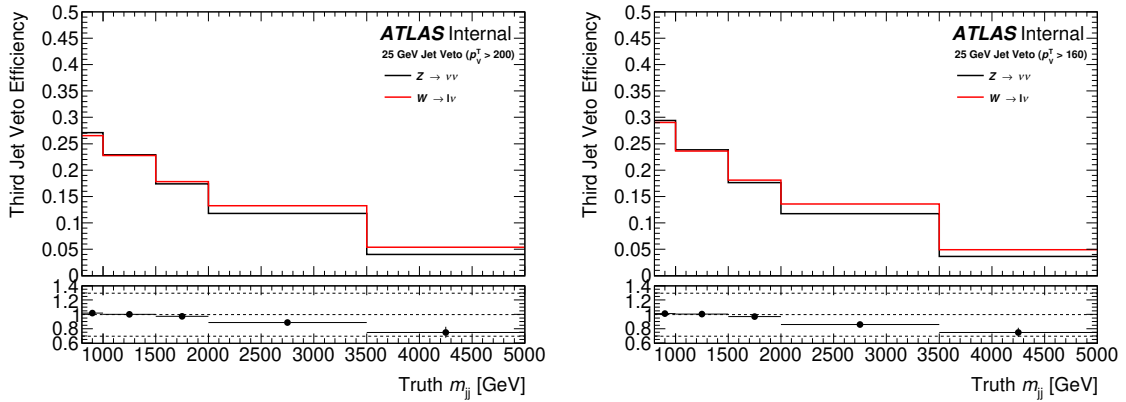


Figure 7.11: Third jet veto efficiency for 25 GeV jets calculated as a function of m_{jj} with the above selection applied, plus $p_T^V > 200$ GeV (left) and $p_T^V > 160$ GeV (right). Jet veto efficiency is calculated using the NLO Sherpa $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ samples, where $W \rightarrow l\nu$ contains both e and μ events, and then compared; the ratio in the lower panel is defined as Z/W . A significant discrepancy can be seen in this ratio at the highest m_{jj} bins.

Additional checks were then performed to try and understand where this difference in the jet veto efficiency might be coming from. The k_t merged, m_{jj} sliced samples also have a matrix element filter on the transverse momentum of the weak boson applied, with $100 < p_T^V < 500$ GeV. Since the p_T^V cut used here does not have an upper cutoff, one possibility is that samples with $p_T^V > 500$ GeV also need to be included in order to remove any discontinuities or boundary effects due to the slicing. Unfortunately, the only available $p_T^V > 500$ GeV Sherpa samples were produced with a different

m_{jj} (GeV)	$p_T^V > 200$		$p_T^V > 160$	
	k_t -Merged	All Samples	k_t -Merged	All Samples
800-1000	1.02 ± 0.01	1.02 ± 0.01	1.01 ± 0.01	1.01 ± 0.01
1000-1500	1.00 ± 0.01	1.00 ± 0.01	1.01 ± 0.01	1.01 ± 0.01
1500-2000	0.97 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	0.97 ± 0.01
2000-3500	0.89 ± 0.02	0.89 ± 0.02	0.86 ± 0.02	0.86 ± 0.02
3500+	0.77 ± 0.08	0.75 ± 0.07	0.76 ± 0.07	0.75 ± 0.06

Table 7.3: Ratio of jet veto efficiency between $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$, as shown in Figure 7.11, calculated separately with $p_T^V > 200$ and $p_T^V > 160$ GeV requirements applied. The “ k_t -Merged” columns show the ratio computed with only the m_{jj} -sliced, k_t -merged Sherpa 2.2.7 samples, which have a matrix element cut at $p_T^V < 500$ GeV. The “All Samples” column shows the ratios calculated with $p_T^V > 500$ GeV samples included as well; no significant difference is seen from the inclusion or exclusion of these samples.

generator configuration: no m_{jj} slicing and with the default merging criterion and Sherpa 2.2.1. Fortunately, it was found that the p_T^V distribution appears smooth when combining these samples with the k_t -merged ones, so it should not be a problem to include them when calculating the jet veto efficiency. Table 7.3 shows the Z/W ratio in efficiency in each m_{jj} bin with and without these higher p_T^V samples included. Their inclusion or exclusion does not appear to affect the jet veto efficiency in any way, which is not unexpected given their small cross section and overall negligible impact on the analysis. Since this seems to have no effect, the rest of these studies were continued with only the k_t merged samples.

Another simple test that can be done is to try breaking up the $W \rightarrow l\nu$ distribution into electron and muon samples, and comparing them separately against $Z \rightarrow \nu\nu$. Figure 7.12 shows the result of this: the jet veto efficiency ratio appears roughly the same between the electron and muon processes. Because there seems to be no difference between the two lepton flavours, some of the future studies in this section were performed with just $W \rightarrow e\nu$ or just $W \rightarrow \mu\nu$ samples to save computation time. Additionally, to make sure that the analysis m_{jj} binning is not masking some important feature of the ratio, Figure 7.12 also contains the jet veto efficiency calculated in fixed 500 GeV-width bins. This second set of plots shows that the Z/W ratio appears quite stable for both electron and muon samples until $m_{jj} = 2500$ GeV (partially into the $2000 < m_{jj} < 3500$ GeV bin), where it begins to decline dramatically.

Next, a similar test was done in which the $W \rightarrow l\nu$ sample was split into W^+ and W^- events, and the ratio evaluated separately. The results of this test are shown in Table 7.4: while broadly similar patterns are seen, it is interesting to note that the efficiency is consistently shifted up in the W^- events and consistently down in the W^+ events away from the $W^\pm$ inclusive values. This might

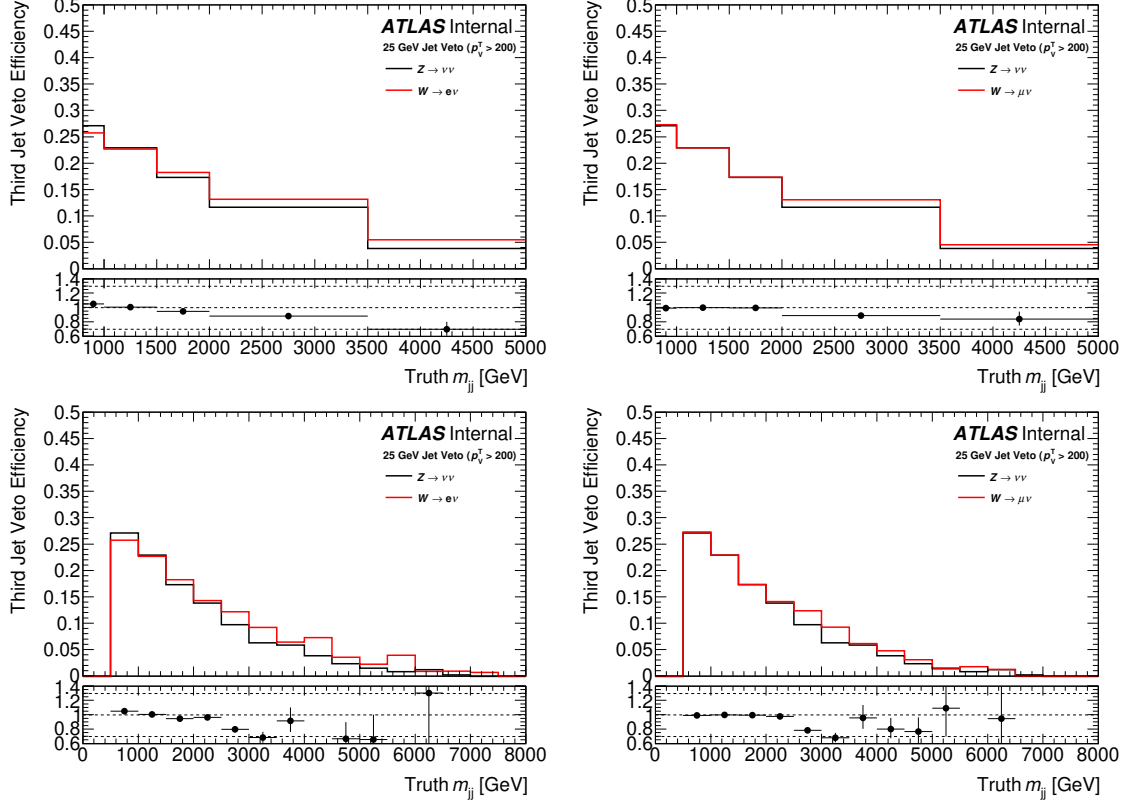


Figure 7.12: Jet veto efficiencies calculated and compared between $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ as shown in Figure 7.11. Here, the $W \rightarrow l\nu$ sample has been divided into $W \rightarrow e\nu$ on the left and $W \rightarrow \mu\nu$ on the right; the same general trend is seen for both. The plots on the bottom show the ratio calculated in fixed, 500 GeV m_{jj} bins rather than the variable bin widths in the above plot. The discrepancy between processes appears independent of the binning strategy used.

suggest that there is some real, fundamental physics difference involved in this discrepancy. Finally, another test was performed in which the jet veto efficiency was calculated separately for different $\Delta\phi_{jj}$ ranges: $0 < \Delta\phi_{jj} < 1$, $1 < \Delta\phi_{jj} < 2$, and $2 < \Delta\phi_{jj} < 2.5$. No significant impact was seen from varying the $\Delta\phi_{jj}$ selection, suggesting that this issue will be consistent between the low- $\Delta\phi_{jj}$ and high- $\Delta\phi_{jj}$ bins of the analysis.

As part of these studies, questions arose over the truth-level object definitions used to identify jets and calculate p_T^V . Variations on the “dressing” scheme explained above when “reconstructing” the truth-level weak boson appeared to have minimal impact, as did using the matrix-element level p_T^V (defined as the vector sum of the two leptons in the matrix element) instead of the final state version. Additionally, the truth jets used here are allowed to include truth neutrinos and muons in a jet, so a test was done in which the truth jets were recalculated without them. This also had no

m_{jj} (GeV)	$Z/W^\pm$	Z/W^+	Z/W^-
800-1000	1.02 ± 0.01	1.00 ± 0.02	1.03 ± 0.01
1000-1500	1.00 ± 0.01	0.98 ± 0.01	1.02 ± 0.01
1500-2000	0.97 ± 0.01	0.94 ± 0.02	0.99 ± 0.02
2000-3500	0.89 ± 0.02	0.85 ± 0.02	0.91 ± 0.02
3500+	0.77 ± 0.08	0.69 ± 0.10	0.80 ± 0.09

Table 7.4: Jet veto efficiency ratios calculated as described above in Figure 7.11 and Table 7.3. Here, the $W \rightarrow l\nu$ sample was split according to the charge of the visible lepton into W^+ and W^- events, and the ratio evaluated separately. The same general trend is seen, but the ratios are shifted consistently up when only taken with W^+ , and consistently down when only taken with W^- .

impact.

7.4.2 Other Differences Between Z and W

So far, we have only compared the jet veto efficiency as a function of m_{jj} . In order to get a better understanding of the problem, truth-level comparisons of other variables between the $W \rightarrow l\nu$ and $Z \rightarrow \nu\nu$ samples were performed with the same selection applied. One obvious distribution to check is the jet multiplicity, N_{jets}^{25} . Figure 7.13 shows the number of jets compared between W and Z samples with both $m_{jj} > 800$ GeV and $m_{jj} > 2500$ GeV cuts applied. 2500 GeV was used as a “high m_{jj} ” threshold since this appears to be where the discrepancy in the veto efficiency ratio emerges, as shown above in Figure 7.12. As the plots show, while the jet multiplicity between the two processes appears the same at low m_{jj} , there are significant differences at high m_{jj} . The $Z \rightarrow \nu\nu$ sample appears to have more events with 4 and 5 jets, while the $W \rightarrow e\nu$ sample has more events with 2- and 3- jet events. This difference will of course lead to a higher jet veto efficiency for the W , as shown above.

Other variables like $\Delta\phi_{jj}$, p_T^V , the p_T and η of the leading jet, the rapidity of the vector boson η_V , and the mass of the vector boson m_V were also checked. Of these variables, the rapidity of the vector boson appeared to show the largest discrepancy between W and Z at high m_{jj} , as shown in Figure 7.14. Here, we see a comparison of η_V plotted with and without the jet veto applied. Before applying the jet veto, the W boson appears to be consistently more forward than the Z , while the Z appears consistently more central. After applying the jet veto, these differences are somewhat suppressed. Note that these differences in boson rapidity also appeared suppressed at low m_{jj} , indicating that this might also be related to the difference in jet veto efficiency. Similar differences at high m_{jj} before and after the jet veto were also seen in the rapidity of the leading jet, $\eta(j_1)$.

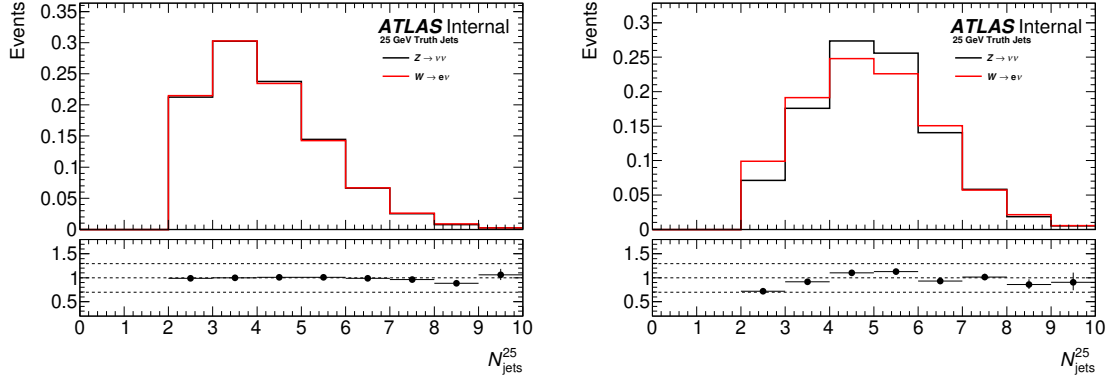


Figure 7.13: Comparison of N_{jets} between $Z \rightarrow \nu\nu$ and $W \rightarrow e\nu$ with $p_{\text{T}}^V > 200$ GeV and the jet veto efficiency study selection applied. On the left, a $m_{\text{jj}} > 800$ GeV cut is applied and the two processes agree quite well; on the right, a higher cutoff $m_{\text{jj}} > 2500$ GeV is applied and significant disagreement is seen, analogous to the differences in jet veto efficiency.

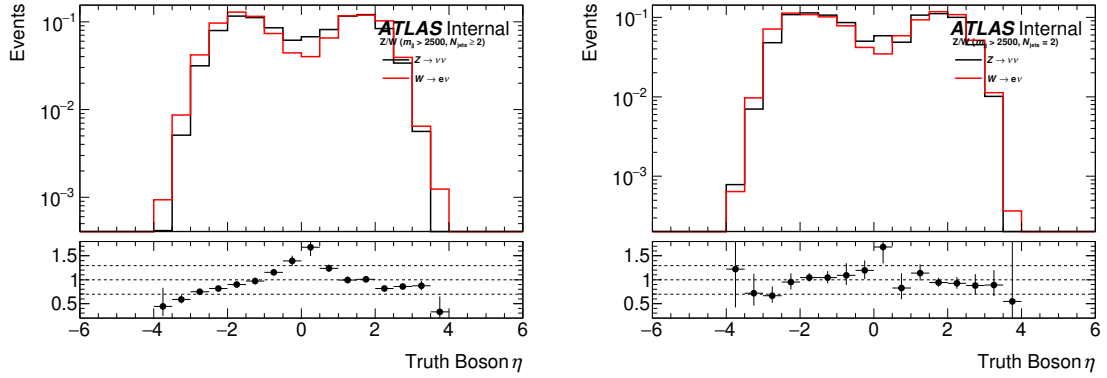


Figure 7.14: Comparison of the η^V of the truth boson between $Z \rightarrow \nu\nu$ and $W \rightarrow e\nu$ with $p_{\text{T}}^V > 200$ GeV, $m_{\text{jj}} > 2500$ GeV, and the jet veto efficiency study selection applied. The plot on the left is before applying the $N_{\text{jets}}^{25} = 2$ veto, where significant difference are seen between the processes. The veto is then applied in the plot on the right, where the differences appear suppressed.

7.4.3 Parton-Level Issues

From the above studies, it is clear that there is some difference between these W and Z samples at high m_{jj} : the rapidity distributions of the weak boson and jets appear different, and there appear to be more jets in the $Z \rightarrow \nu\nu$ samples than in the W . It is still unclear, however, why this might be occurring. If this is a generator level issue, it could be an issue with the parton shower model, or with the matrix element calculation. By examining the full truth record of the events and looking at parton-level quantities, we can determine whether or not this discrepancy is also present in the

matrix element or whether it only appears after running the parton shower⁹³. As a first test, we used the matrix element information to classify the various W and Z diagrams by the flavour of their incoming partons: i.e. whether events involve two quarks (qq), a quark and an anti-quark ($q\bar{q}$), a quark and a gluon (qg), or two gluons (gg)⁹⁴. Figure 7.15 shows the fraction of each type of diagram in both the W and Z samples as a function of m_{jj} . Interestingly, the distributions look somewhat similar at low m_{jj} but quite different at high m_{jj} , where the $Z \rightarrow \nu\nu$ is dominated by qq but the W is more evenly divided between qq and qg . This might indicate that different types of diagrams are contributing to the observed discrepancies.

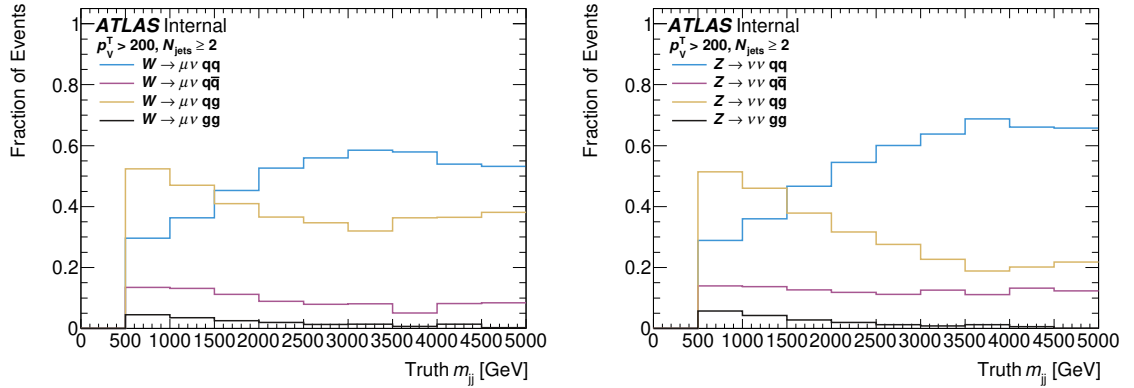


Figure 7.15: Plots showing the $W \rightarrow \mu\mu$ (left) and $Z \rightarrow \nu\nu$ (right) samples divided by the flavour of their incoming partons into qq , $q\bar{q}$, qg , and gg events. The fraction of each type of event is plotted as a function of m_{jj} ; while the samples appear similarly comprised at low m_{jj} , by 2500 GeV the Z sample becomes dominated by qq events while the W is more evenly split between qq and qg .

An additional test was done in which the ratio of W to Z events was calculated separately for only qq , $q\bar{q}$, qg , and gg events as a function of dijet mass. The ratio was found to be stable as a function of m_{jj} for qg events, but differ considerably for both qq and $q\bar{q}$ events at high m_{jj} by around a factor of 2. This suggests that the problem might be entirely due to qq events, and so another test was done in which the jet veto efficiency was calculated separately for qq and qg . As Figure 7.16 shows, while the qq ratio has the same familiar shape as shown above in Figure 7.11, the qg ratio does not. It appears noticeably more stable at high m_{jj} , although it is possible this is partially a statistical effect, as the ratio in the $m_{jj} > 3.5$ TeV bin has a large uncertainty. Still, it seems to confirm that the differences between the two processes might be more concentrated in diagrams

⁹³The matrix element partons are chosen as described in Section 7.3: status 3 partons are selected for MC@NLO “H” events, and status 20 partons for MC@NLO “S” events.

⁹⁴For this calculation: qg and $\bar{q}g$ are considered identical; it would probably also be reasonable to combine qq and $q\bar{q}$ together.

with two incoming quarks than those involving gluons.

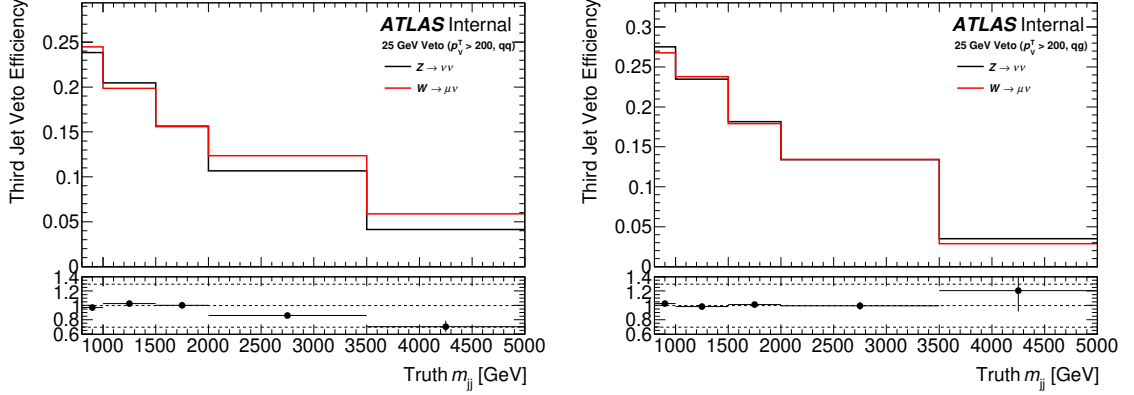


Figure 7.16: Jet veto efficiencies calculated and compared between $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ as shown in Figure 7.11. Here, both W and Z samples were split by the flavour of their incoming partons, and the veto efficiency evaluated separately for quark-quark (qq) and quark-gluon (qg) events. On the left, Z/W is evaluated just for qq events, where the similar disagreement is seen, while on the right, Z/W is evaluated just for qg events, where the ratio appears more stable.

Finally, another variable that can be considered is the invariant mass of the *incoming* partons, rather than the outgoing jets, which we label m_{qq} ⁹⁵. This can be thought of as the energy of the incoming system, in the same way that m_{jj} is in some sense the energy of the outgoing (hadronic) system for vector boson fusion. Figure 7.17 shows this variable calculated in this selection with a $m_{jj} > 2500$ GeV cut applied separately for qq events and for qg events. Clear differences are seen between the W and Z distributions for qq events– the W appears noticeably more boosted relative to the Z – but these differences are suppressed for qg events. From these results, it is clear that whatever differences exist between the two processes at high m_{jj} , they are present at the matrix element level as well. This means that the parton shower is *not* at fault, and there is either a bug in the matrix element calculation or some real physics difference.

7.4.4 Parton Multiplicity Tests

An additional parton-level quantity that can be inspected is the number of matrix element partons in each event: the parton or matrix element multiplicity. While the parton shower merging process will add additional jets, this is still somewhat related to the jet multiplicity. Figure 7.18 shows the parton multiplicity for W and Z events, computed separately at low and high m_{jj} and before and

⁹⁵Technically, of course, this is only correct for qq events– for a qg event this should be m_{qg} , and so on. But I used m_{qq} anyway to refer to the mass of the incoming system.

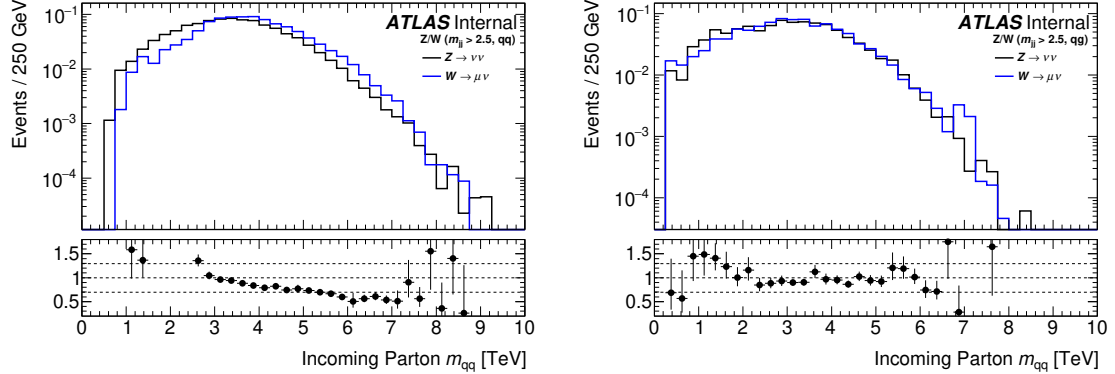


Figure 7.17: Comparison of the invariant mass of the vector sum of the incoming partons, labelled here as m_{qq} , between $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ at high $m_{jj} > 2500$ GeV. The comparison is performed separately for just qq events (left) and just qg events (right).

after the $N_{\text{jets}}^{25} = 2$ veto. The samples both appear to be dominated by four-parton events, especially at high m_{jj} , and even after the jet veto there are still large numbers of 3- and 4- parton events. This is partially a consequence of the use of k_t merging, which has the effect of classifying more forward partons as part of the matrix element, making it more likely that the correct jets will be selected when imposing a m_{jj} filter, as explained above. However, there are clearly dramatic differences between the parton multiplicity distributions at high m_{jj} , especially after applying the jet veto.

At this point, we presented these results to theorists and Sherpa experts in order to get their opinion on what might be going on here. They suggested that, since there are major mismatches in parton multiplicity, there might be a generator-level problem in Sherpa involving higher-multiplicity matrix elements. As a reminder, these samples were produced with up to four partons, two of which are up to NLO in α_s , a configuration labelled below as the “baseline” or “NLO2+LO2”. Following their suggestion, therefore, we prepared a sequence of test samples with fewer partons: a “NLO2+LO0” configuration with *exactly* two NLO partons and no additional LO diagrams, and a “NLO2+LO1” configuration that allowed for one additional LO parton and three total. Additionally, to understand whether this might have something to do with higher order NLO diagrams, leading order only test samples were also prepared with two (LO2), three (LO3), and four (LO4) partons for comparison purposes. k_t merging and m_{jj} slicing was used for the tests, and 2.5 million events each generated in a $m_{jj} > 1000$ GeV, $p_T^V > 200$ GeV slice of phase space for both $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$. A NLO2+LO2/baseline test sample was also regenerated with the same parameters and statistics in order to make a direct comparison.

The jet veto efficiency was then calculated for each of these test samples: the results are shown

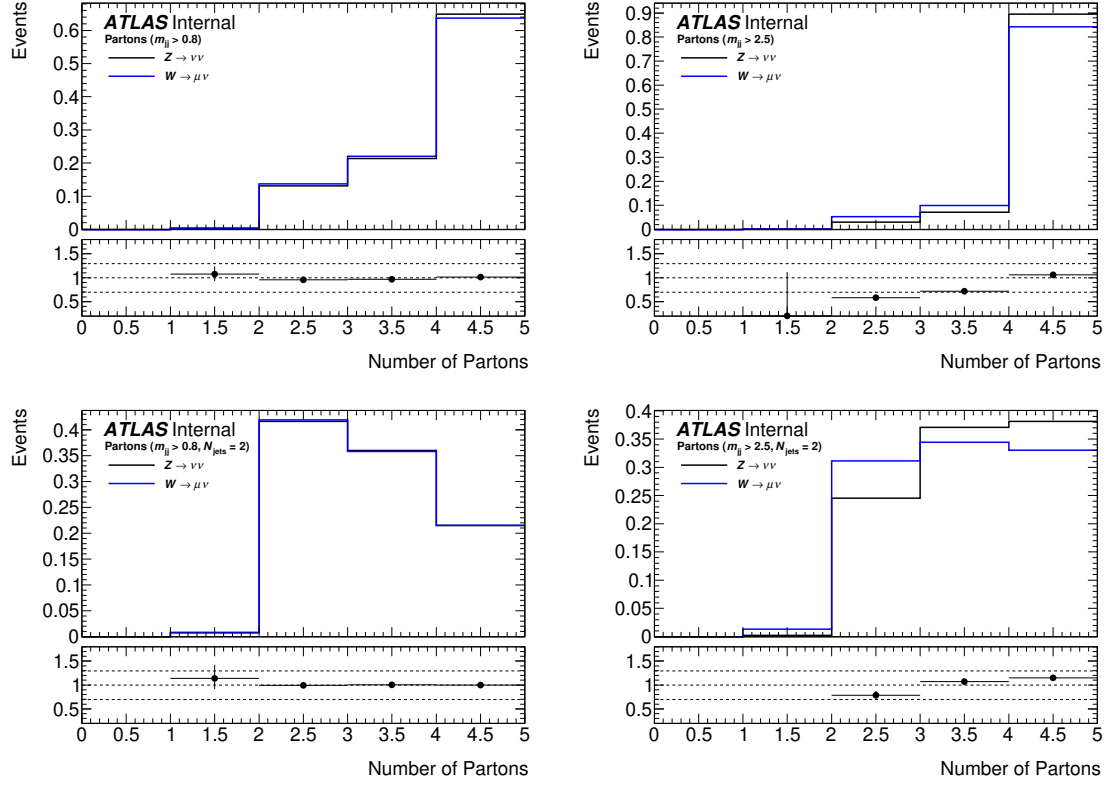


Figure 7.18: Comparison of the matrix element parton multiplicity between the $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ samples. On the left, comparisons were done with a low $m_{jj} > 800$ GeV cut, while on the right they were done with a high $m_{jj} > 2500$ GeV cut. The top shows the comparison before applying a jet veto, while on the bottom, events must have exactly $N_{\text{jets}}^{25} = 2$ jets.

below. Figure 7.19 shows the jet veto efficiency ratio compared from the 4-parton LO-only and NLO configurations: as can be seen from the plots, there is no real difference in behavior. The same m_{jj} dependence and the same discrepancy at $m_{jj} > 3.5$ TeV is observed in both setups.

Next, we looked at the other test samples: the NLO and LO-only versions of the two- and three- parton configurations. These results are presented below in Figure 7.20. Strikingly, the same discrepancy is *not* seen in either the NLO2+LO0 or LO2 configurations: whether the partons are leading order or not does not seem to matter, what matters is the overall number of partons. There is still a small discrepancy in the highest bin, but it appears to be on the order of 5-10%, which is substantially reduced from the 30-40% disagreement seen before. The LO3 and NLO2+LO1 configurations also appear very similar, and quite strange compared to both the two-parton and four-parton setups. The Z jet veto efficiency is consistently higher than the W , except in the final

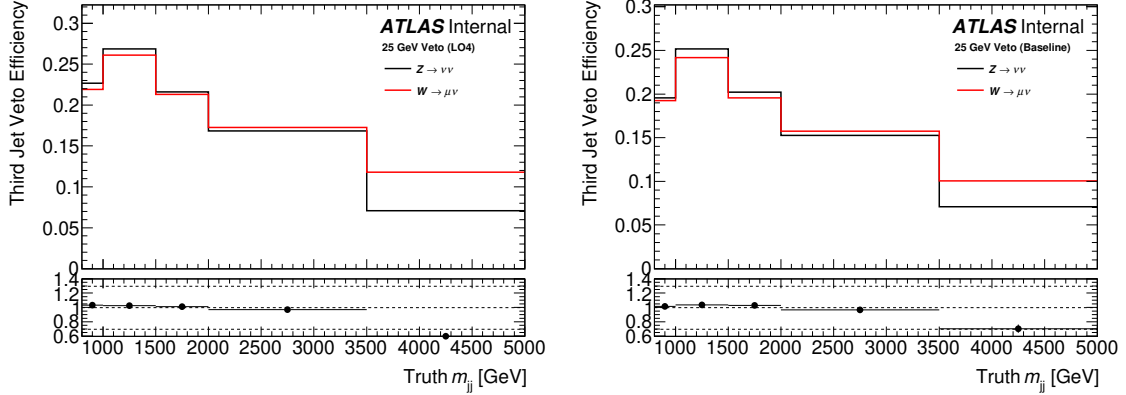


Figure 7.19: Jet veto efficiencies calculated and compared between $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ as shown in Figure 7.11, using the test samples with different matrix element configurations. The left shows the four-parton LO4 configuration, which can have up to four leading order partons, while the right shows the four-parton NLO2+LO2 baseline configuration, where two of the four partons can be NLO. No significant difference is seen between the two setups.

m_{jj} (GeV)	NLO2+LO2	NLO2+LO0	LO2	LO4
800-1000	1.02 ± 0.02	1.01 ± 0.01	1.00 ± 0.01	1.03 ± 0.01
1000-1500	1.04 ± 0.01	1.00 ± 0.01	1.01 ± 0.00	1.03 ± 0.01
1500-2000	1.03 ± 0.01	0.99 ± 0.01	1.00 ± 0.01	1.01 ± 0.01
2000-3500	0.97 ± 0.02	0.95 ± 0.01	1.01 ± 0.01	0.97 ± 0.01
3500+	0.71 ± 0.05	0.94 ± 0.07	0.92 ± 0.06	0.60 ± 0.04

Table 7.5: Jet veto efficiency ratios calculated from the two-parton and four-parton test samples, at both NLO and LO, for the NLO2+LO2 (baseline), NLO2+LO0, LO2, and LO4 configurations. In general, good agreement is seen between the two two-parton configurations across all m_{jj} bins.

m_{jj} bin, where there is still a large disagreement.

These results were presented to our theorist collaborators, as well as Sherpa experts and developers. The conclusion was that there is very likely a bug involving the merging of higher multiplicity parton matrix elements, which is why strange behavior is seen in the three- and four- parton setups. While naively, one would expect the matrix element calculation to be *more* accurate than the parton shower, it seems that allowing additional jets to be modelled by the parton shower instead leads to better overall agreement between the W and Z processes. The conclusion then was that, since the two-parton NLO configuration is trusted more than the four-parton baseline configuration, these are the samples that should be used when calculating an uncertainty to cover the jet veto efficiency difference. An additional 10 million events were generated for this sample in order to increase statistics, and the jet veto efficiency evaluated in each m_{jj} bin is shown below in Table 7.5.

Meanwhile, this matrix element calculation bug in Sherpa is still being investigated by the

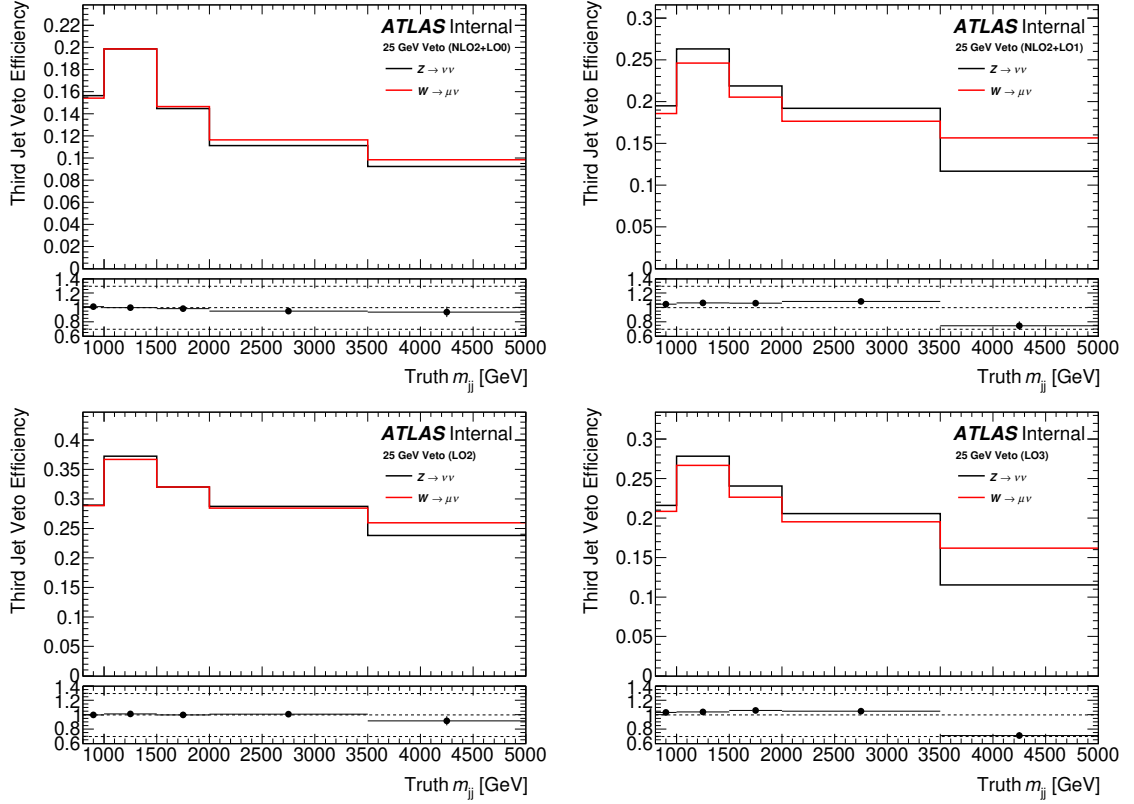


Figure 7.20: Jet veto efficiencies calculated and compared between $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\nu$ as shown in Figure 7.11, using the test samples with different matrix element configurations. The top plots show NLO configurations, while the plots on the bottom show LO-only configurations. The left plots show samples with only two-parton matrix elements, NLO2+LO0 (top) and LO2 (bottom), where in both cases the Z/W ratio appears much more stable. The right plots show samples with up to three partons, NLO2+LO1 (top) and LO3 (bottom), where in both cases the ratio does not appear stable.

theorists and experts, and remains unsolved as of this writing. The jet veto differences between W and Z were seen in other, more recent V+Jets test samples produced with different configurations, including in samples that did not use k_t merging and m_{jj} slicing, so it is definitely not an artifact of those choices⁹⁶. Using the two-parton samples is a reasonable workaround, but it would be good to solve this issue for future samples and future analyses.

⁹⁶Most recently, in a set of test samples generated with Sherpa 2.2.11 with various improvements to stability and performance. Unfortunately, these improvements did not fix this particular issue.

7.5 Issues with EWK V+Jets

Most of this chapter has focused on issues involved in producing Monte Carlo for the QCD V+Jets processes. However, problems were encountered in producing Monte Carlo at high accuracy for the electroweak V+Jets as well. As noted above, during the preliminary analysis, Sherpa 2.2 was used to generate electroweak V+Jets sample at leading order. Because Sherpa used for the QCD V+jets and multi-boson (VV+jets, etc.) samples, it was desired to also produce a NLO electroweak V+Jets sample using Sherpa. Unfortunately, a bug involving quark color flow was discovered that made this impossible. Sherpa 2.2 hardcodes the flow of color from incoming quarks to outgoing quarks for specific Feynman diagrams, and it turns out that vector boson fusion production of a weak boson at NLO is not one of those diagrams. This would have led to the generation of events where the color of outgoing quarks was not assigned correctly relative to the color of incoming quarks, a serious potential problem.

Therefore, a NLO sample generated with Herwig 7 was used instead. During the preliminary analysis, the Sherpa LO prediction was reweighted upward by 20% to the Herwig NLO cross section. This is a relatively large effect, and since the two generators are quite different, questions were raised about whether this approach was acceptable, especially when it came to assign systematic uncertainties. For the final version of the 139 fb^{-1} analysis, the NLO Herwig 7 sample was used directly. However, a modelling problem was discovered in the $Z \rightarrow ll$ electroweak samples when comparing predictions from Herwig 7 (at NLO) to Sherpa (at LO). The $Z \rightarrow ll$ samples also contain diagrams involving $\gamma \rightarrow ll$, generally with cuts on the invariant mass of the vector sum of the leptons, m_{ll} , to exclude all but offshell photons. Some $\gamma^* \rightarrow ll$ diagrams with a mass around $m_{ll} = 60\text{ GeV}$ appeared to be missing from the NLO Herwig 7 samples that were present in the Sherpa LO samples.

The largest impact from this can be seen when looking at the missing transverse momentum at truth level, when calculated with the visible leptons marked as an invisible particle as introduced in Section 6.3.2. slope can be seen in Figure 7.21 when comparing the ratio of $Z \rightarrow ll$ and $Z \rightarrow \nu\nu$ between Herwig and Sherpa. This issue was reported to and acknowledged by the Herwig 7 developers, but is still unresolved as of this writing. In the absence of any better options, it was decided to reweight the Herwig 7 $Z \rightarrow ll$ prediction as a linear function of the missing transverse momentum at truth level, calculated using the above plot. The reweighting function was found to be $1.02 + 0.00041 E_T^{\text{miss}}$, so compared to the 20% reweighting from LO to NLO mentioned above, this is a relatively small impact that only affects the $Z \rightarrow ll$ (and not the $Z \rightarrow \nu\nu$ or $W \rightarrow l\nu$). Still, hopefully this issue will be fixed upstream for future versions of the analysis.

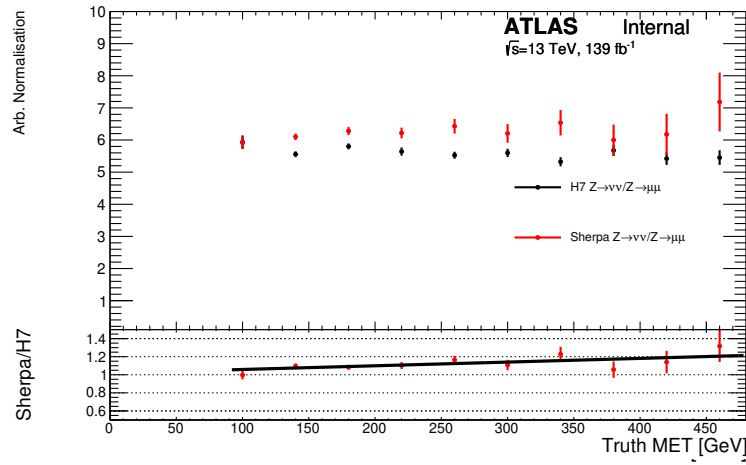


Figure 7.21: Ratio of $Z \rightarrow \nu\nu$ and $Z \rightarrow \mu\mu$, taken from electroweak V+Jets MC simulated with Herwig 7 at NLO (black) and Sherpa 2.2.1 at LO (red), as a function of the truth-level E_T^{miss} . The visible leptons have been marked invisible, meaning that they are vectorially added to the E_T^{miss} . The difference between the two generators appears to be due to missing $\gamma^* \rightarrow ll$ diagrams in the Herwig sample.

CHAPTER 8

Background Estimation

The definition of a signal region for the VBF+MET analysis— a series of requirements or “cuts” on different variables that the invisible Higgs signal process would satisfy— was given in Chapter 6. One of the goals when defining a signal region selection is to minimize the contamination from other Standard Model background processes. However, this background contamination cannot be fully removed; in this case, for instance, the $Z \rightarrow \nu\nu$ background in particular is irreducible and cannot be distinguished from the invisible decay of a Higgs boson. An important part of the analysis is then to carefully estimate the size of this background contribution. Then, the estimate can be compared to the observed data in the signal region to determine whether or not the observation is consistent with the background prediction or if there is any sign of new physics.

An overview of the various background processes this analysis is concerned with was given in Section 6.5.3. As discussed in Chapter 7, Monte Carlo simulations are then used to model these processes and generate simulated datasets. If our MC generators perfectly mirrored reality, it would be possible to just use the simulations to estimate the number of expected background events. In practice, that is not the case: for one thing, we know of technical problems and limitations with the event generators themselves, as described in Chapter 7. But that should perhaps not be too surprising, given that the electroweak and QCD physics being modelled here is extremely complex. Performing matrix element calculations to NLO is certainly better than LO, but it would be even better to go to NNLO, and so on. Parton shower models also introduce considerable uncertainties. And given that we also need to take into account detector effects, uncertainties also arise when performing a simulation of the ATLAS detector over this generated MC. It is therefore important to use *data-driven* approaches where possible, and ensure that any prediction from MC is validated or rescaled using real data from the experiment.

This chapter goes into detail as to how background estimation is done for the VBF+MET analysis, building on the work described in the last two chapters. Sections 8.1 and 8.2 explain how the V+Jets background estimate is performed using the dedicated W and Z control regions defined in Chapter 6. The next three sections then outline the techniques used to estimate a different background: “multijet” QCD events, where mismeasurement leads to a jet being improperly identified as either a lepton (Section 8.3) or E_T^{miss} (Sections 8.4 and 8.5).

8.1 V+Jets Estimate

The careful and precise estimate of the $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ lost lepton backgrounds is the main challenge of this analysis. Dedicated MC simulations of V+Jets processes are produced as described in Section 7, with increased statistics in the analysis’s high- m_{jj} phase space. These simulated events are then rescaled in a partially data-driven process in order to predict the expected number of background events in the signal region. The rescaling is performed using dedicated one-lepton and two-lepton control regions for W and Z processes, respectively. As the Higgs to invisible process contains no leptons, extremely low signal contamination will be seen in these regions. Because $Z \rightarrow \nu\nu$ and $Z \rightarrow l^+l^-$ are fundamentally the same process, the two-lepton control region can be used to estimate the $Z \rightarrow \nu\nu$ background.

These one- and two-lepton control regions are defined identically to the signal region definition given in Section 6.3, except that the lepton veto is replaced with a requirement that there be exactly one or two charged electrons or muons. They are quite pure in the W+Jets and Z+Jets processes, with around 90%+ of the events observed in this region appearing to come from both strong and electroweak V+Jets processes. Subtracting contributions from the other backgrounds (top processes and multiboson events), we can then compare the number of observed data events to the number of events predicted by the MC simulations. Given the purity of the regions, any difference between data and MC here is highly likely to involve the V+Jets process. Therefore, we can take that ratio between data and simulation from the control region and use it as a normalization factor β in the signal region, in order to correct the estimate from Monte Carlo there. This technique should work provided any MC mismodelling is independent of the region definitions.

Instead of thinking about normalizing the MC estimate in the signal region, we could instead imagine that we are “transferring” the measurement in the control region(s) to the signal region. The ratio of V+Jets events between the two regions should be the same for both data and Monte Carlo. Therefore, the ratio of MC between the two regions can be taken as a *transfer factor*, α , which

is applied to the data recorded in each control region and used to predict the expected number of background events in the signal region. Mathematically, these two techniques are completely equivalent, as shown in Equation 8.1 below⁹⁷.

$$V_{\text{SR}} \approx \frac{V_{\text{SR}}^{\text{MC}}}{V_{\text{CR}}^{\text{MC}}} V_{\text{CR}}^{\text{data}} = \alpha_V V_{\text{CR}}^{\text{data}} = \frac{V_{\text{CR}}^{\text{data}}}{V_{\text{CR}}^{\text{MC}}} V_{\text{SR}}^{\text{MC}} = \beta_V V_{\text{SR}}^{\text{MC}} \quad (8.1)$$

The transfer (or normalization) factors are computed separately for each bin used in the analysis, and applied simultaneously during the fit. Therefore, the normalization between data and MC can vary independently across different m_{jj} , $\Delta\phi_{\text{jj}}$, $E_{\text{T}}^{\text{miss}}$, and N_{jets} regions. Note that in past versions of the analysis, as well as in the preliminary 139fb^{-1} result, separate transfer factors were computed for the W and Z backgrounds, with only $Z \rightarrow l\bar{l}$ used to estimate $Z \rightarrow \nu\nu$. In the current version of the analysis, the one-lepton control region is now also used to help constrain the Z background, as explained below in Section 8.2. However, separate transfer factors from the two control regions are still computed and input into the fit.

8.1.1 W Control Regions

The one-lepton W control regions are mostly identical to the signal region, except that the lepton veto is removed and events must have a single electron or muon instead. More stringent quality requirements are applied to these leptons than are used in the signal region’s lepton veto, as explained in Section 6.3.2. As noted there, electrons must pass the “tight” identification and muons the “medium” identification working points for an event to be accepted into the W control regions. Additionally, both electrons and muons must pass requirements on their isolation and on the longitudinal and transverse impact parameters of their associated track to ensure that they are well separated from other particles and were produced from the primary vertex. The lepton is then required to have transverse momentum $p_{\text{T}} > 30\text{ GeV}$. Also, in order to avoid having to change the $E_{\text{T}}^{\text{miss}}$ cut, the missing transverse momentum is modified by marking the lepton (and any finite state radiation emitted from it) as “invisible” in this region. This means that it is not included when we sum the four-momenta of all observed objects to calculate $E_{\text{T}}^{\text{miss}}$, and so the unchanged missing transverse momentum cut is then applied to this modified $E_{\text{T}}^{\text{miss}}$. Another way to think of this is that the “real” $E_{\text{T}}^{\text{miss}}$ has been corrected by taking the vectorial sum of it with the visible lepton, so that the modified value continues to estimate the transverse momentum of the vector boson, p_{T}^V .

⁹⁷While technically, the normalization factors β are used in the fit model described in Chapter 9, the technique is still commonly referred to as a transfer factor technique.

This region is split into separate electron and muon control regions in order to deal with contamination from a misidentified multijet background. The multijet background here arises when a jet in a pure-QCD event is improperly identified as an electron or muon due to detector mismeasurement. The impact of these misidentified or “fake” leptons is assessed separately for electrons and muons. The calculation of this fake lepton estimate is presented in more detail in Section 8.3 below, but it involves applying an additional cut to both electron and muon regions in order to exclude a small area of fake-enriched phase space from which the estimate can be derived. In the $W \rightarrow e\nu$ control region, we therefore impose an additional cut on the missing transverse momentum significance, $S_{\text{MET}} > 4\sqrt{\text{GeV}}$ [188]. In the $W \rightarrow \mu\nu$ control region, this is instead done with a requirement on the transverse mass, $m_T > 20 \text{ GeV}$. These variables are defined in Section 8.3 below.

Figures 8.1 and 8.2 show pre-fit distributions from both the $W \rightarrow e\nu$ and $W \rightarrow \mu\nu$ control regions for m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and E_T^{miss} (corrected as described above by adding the lepton four-momentum). Because these are control regions, real data can be included before unblinding the full analysis; as mentioned in Section 6.3.1, the data included here was recorded using lepton triggers rather than the missing transverse momentum trigger. As can be seen from these plots, at low m_{jj} strong V+Jets processes dominate, but as m_{jj} increases contributions from electroweak V+Jets processes gradually become more and more important. Contributions from the fake lepton backgrounds are not shown, as these will be applied during the fit.

8.1.2 Z Control Regions

The two-lepton Z control region, like the W control region, is mostly identical to the signal region except for the lepton requirement. Here, exactly two leptons are required, which must have the same flavour and opposite signs. The Z control region also uses more stringent lepton identification requirements than the lepton veto, however these are slightly looser than definitions used in the W control region above: electrons and muons must pass their “loose” working points, as opposed to the tight and medium working points, respectively. One of the leptons must have transverse momentum $p_T > 30 \text{ GeV}$, while the other only needs to have $p_T > 4.5 \text{ GeV}$. The E_T^{miss} is then calculated in this region with both leptons marked as invisible. To ensure the region is relatively pure in Z events, an additional requirement on the invariant mass of the vector sum of both leptons, $|M_{ll} - M_Z| < 25 \text{ GeV}$, is imposed⁹⁸. Additionally, unlike the W CRs, $Z \rightarrow ee$ and $Z \rightarrow \mu\mu$ events are not separated and are grouped together into a single inclusive $Z \rightarrow ll$ control region.

⁹⁸Given that $M_Z = 91.2 \text{ GeV}$, this amounts to requiring $66.2 < M_{ll} < 116.2 \text{ GeV}$.

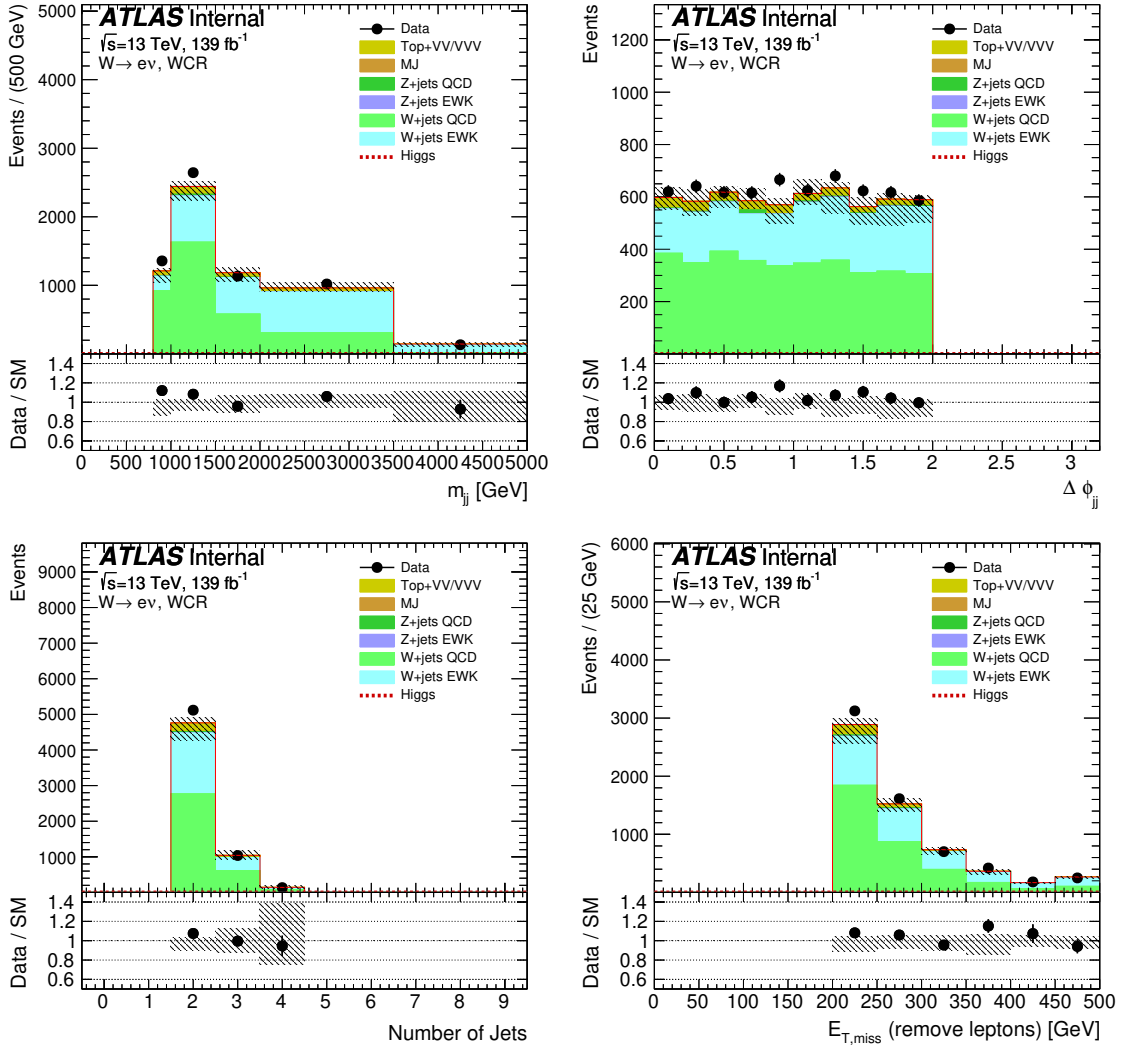


Figure 8.1: Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and the transverse momentum of the vector sum of E_T^{miss} and the visible lepton in the $W \rightarrow e\nu$ control region. The error band includes both statistical and systematic uncertainties. Note that while these plots show the inclusive $W \rightarrow e\nu$ control region, they were made for the preliminary full run 2 analysis, meaning that only events $E_T^{\text{miss}} > 200 \text{ GeV}$ are included.

While the Z control region is relatively pure in $Z \rightarrow ll$ events, it was discovered to have a not insignificant amount of contamination from multi-boson processes, especially at high m_{jj} . VV , VVV , $\text{VBF } H \rightarrow WW$, and $\text{VBF } H \rightarrow \tau\tau$, and other similar processes were found to contribute about 33% to the overall background in the highest $m_{jj} > 3500 \text{ GeV}$ bin. The amount of multiboson was found to increase as a function of the uncorrected $(E_T^{\text{miss}})_{\text{uncorrected}}$; that is, without adding the

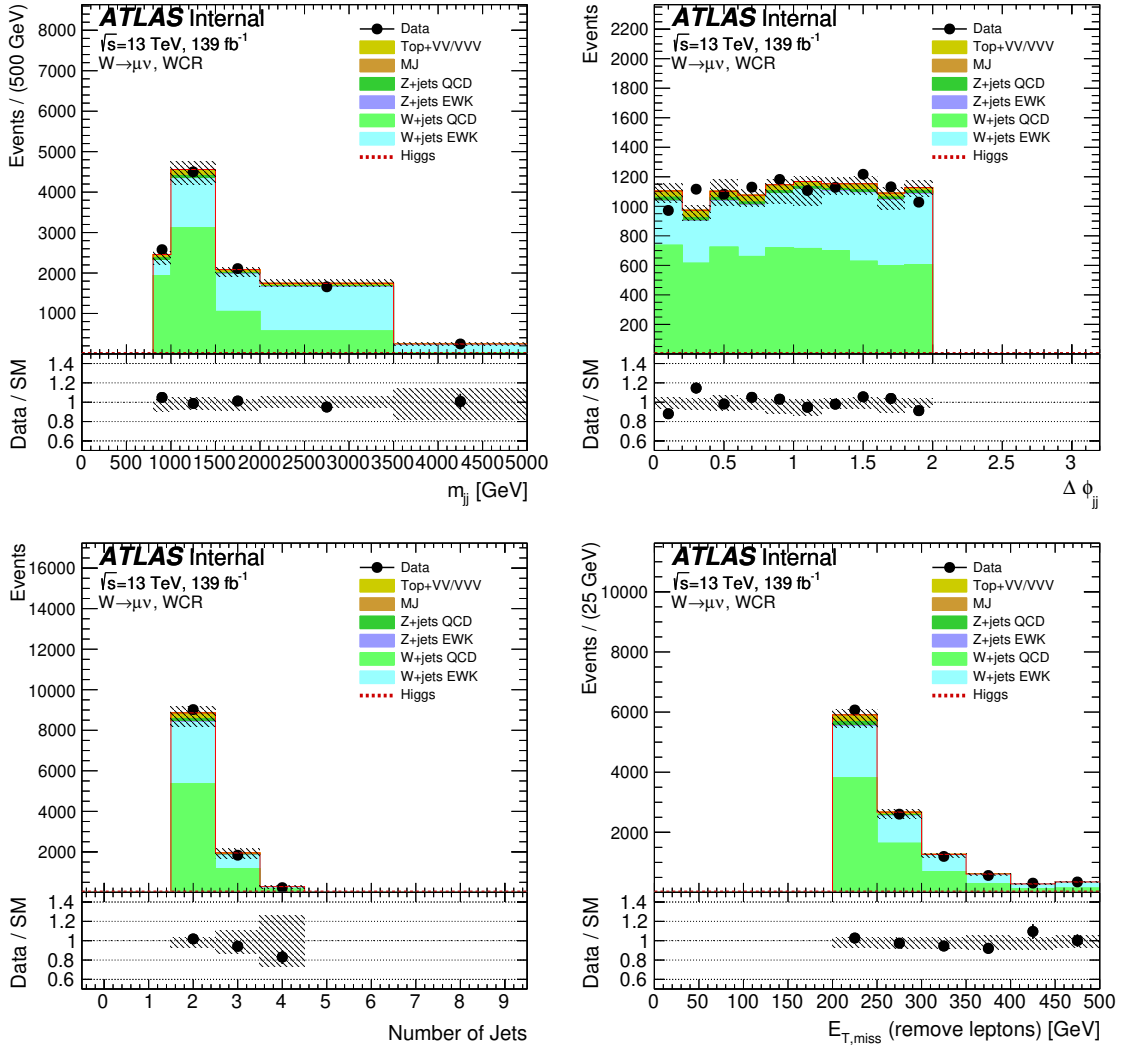


Figure 8.2: Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and the transverse momentum of the vector sum of E_T^{miss} and the visible lepton in the $W \rightarrow \mu\nu$ control region. See Figure 8.1 for more details.

four-momenta of the two leptons to the observed missing transverse momentum. This can be seen in Figure 8.3, where after about 100 GeV, the multiboson background becomes dominant. Due to the poor agreement between data and MC in this region, a normalization factor is computed by taking the ratio of data to MC. An uncertainty on that normalization factor is assigned to be the difference between the ratio and unity: therefore, the multiboson normalization was found to be 0.56 ± 0.44 . With this normalization factor applied to the multiboson background in the Z control region, a cut on $(E_T^{\text{miss}})_{\text{uncorrected}} < 70 \text{ GeV}$ was then added. This reduced the contamination in the highest m_{jj}

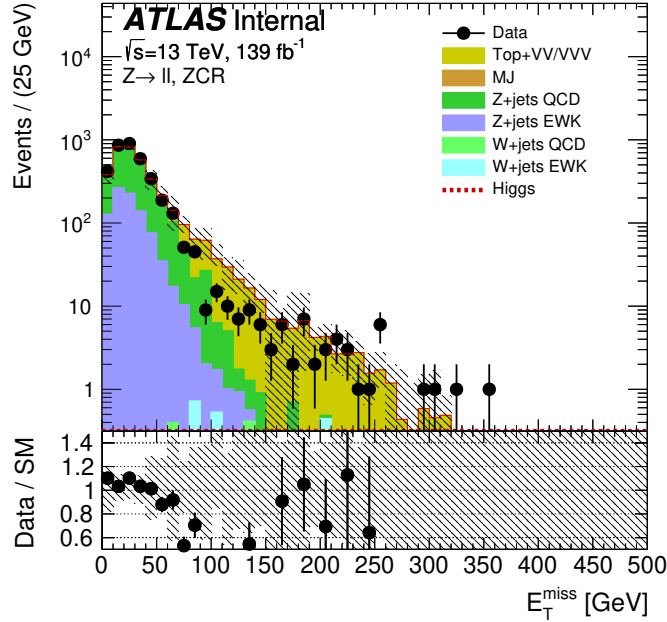


Figure 8.3: Pre-fit plot showing the missing transverse momentum, E_T^{miss} , in the $Z \rightarrow ll$ control region. Below around $E_T^{\text{miss}} < 70$ GeV, this region is pure in strong and electroweak Z+jets events. However, by 100 GeV and above, an increasingly large fraction of the background appears to come from multi-boson (VV, VVV, etc.) events. This background also appears to be poorly modeled, as the data to MC agreement in this region becomes significantly worse.

bin from 33% to around 9% with a minimal loss in Z+jets statistics.

Figure 8.4 shows pre-fit distributions from both the $W \rightarrow e\nu$ and $W \rightarrow \mu\nu$ control regions for m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and E_T^{miss} , with this additional $(E_T^{\text{miss}})_{\text{uncorrected}} < 70$ GeV requirement applied.

8.1.3 V+Jets Uncertainties

There are two main sources of uncertainty on the V+Jets background estimate that need to be taken into account. The first is the statistical uncertainties in the control regions, both from real data and from the V+Jets Monte Carlo samples. Work to reduce the Monte Carlo statistical uncertainties was documented in Section 7.3. The second are the theoretical systematic uncertainties on those MC samples, including the scale and PDF uncertainties introduced in Section 7.1. Separate systematics are determined for the QCD and electroweak V+Jets samples, and are not correlated between the two.

The QCD V+Jets samples were generated with Sherpa, using the MEPS@NLO merging and matching technique outlined in Section 7.1.5. In addition to the factorization and renormalisation

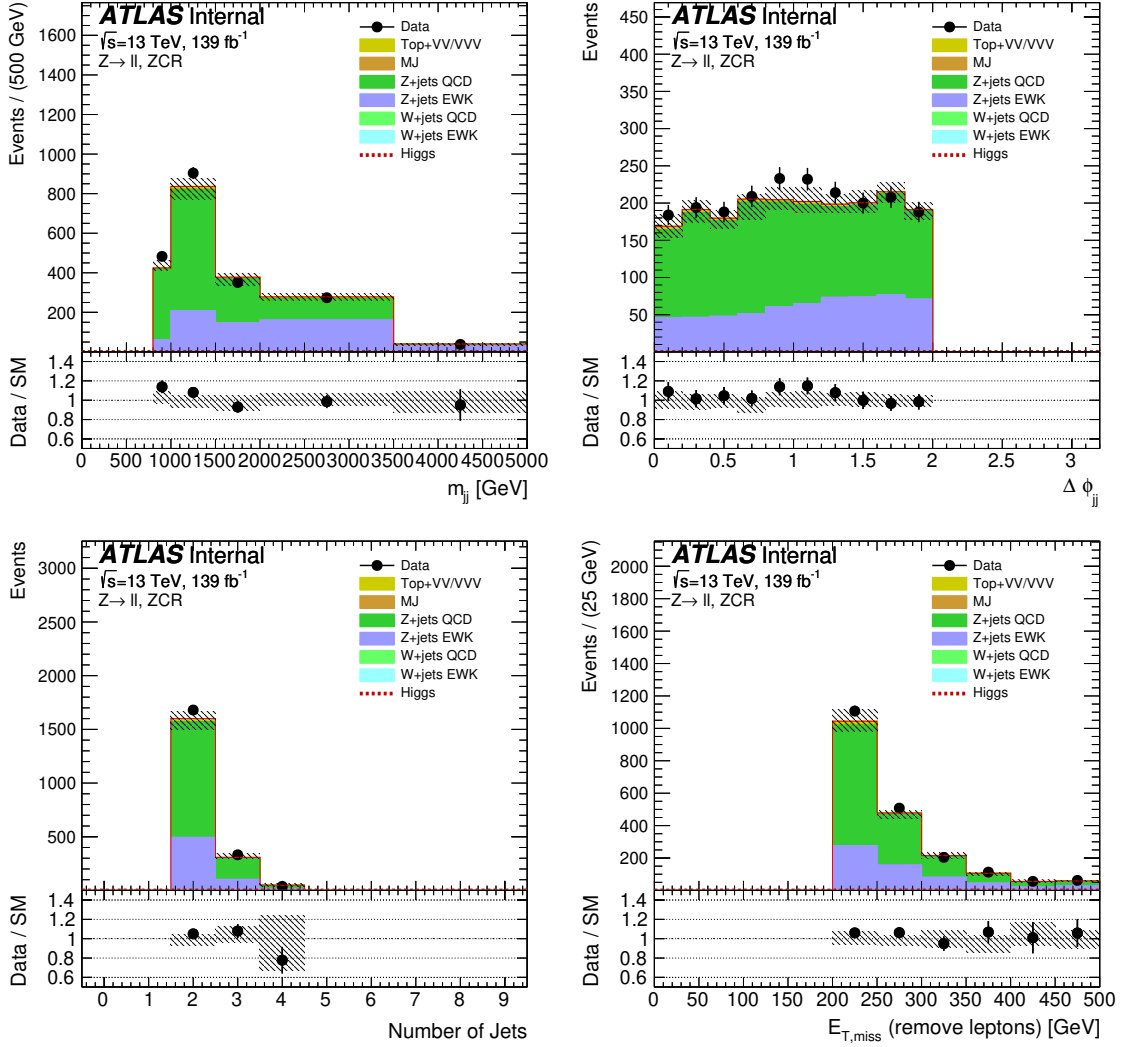


Figure 8.4: Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and the transverse momentum of the vector sum of $E_{\text{T}}^{\text{miss}}$ and the visible lepton in the $Z \rightarrow ll$ control region. Both electron and muon events are included here. See Figure 8.1 for more details.

scale variations, this means that there are also resummation (μ_Q) and merging (Q_{cut}) scales which need to be varied. The factorization and renormalisation scales are varied during the event generation process up and down by factors of two, both independently and together, and the result saved as an event weight. Combined with the nominal value, this gives seven different data points for each event, and so an envelope of these seven points is constructed and compared with the nominal value to calculate a systematic uncertainty. These variations range from $^{+27}_{-18}\%$ at low m_{jj} to $^{+43}_{-26}\%$ on the event yield at high m_{jj} . For the resummation and merging scales, the complex matching

and merging procedure means that it is not possible to vary these scales during event generation. Instead, separate samples were generated with μ_Q shifted up and down by a factor of 2, and Q_{cut} shifted from 20 GeV to 15 GeV and 30 GeV. A relative uncertainty was then defined, by comparing the up and down variations as shown below in Equation 8.2:

$$\text{rel. unc.} = \frac{r_{\text{up}} - r_{\text{down}}}{r_{\text{up}} + r_{\text{down}}} \quad (8.2)$$

Because it's very expensive to produce Sherpa V+Jets events, it was not possible to generate these samples with high statistics or reconstruct them. These relative uncertainties were instead evaluated using a loose truth-level selection, with $m_{jj} > 800$ GeV, $\Delta\eta_{jj} > 2.5$, $E_T^{\text{miss}} > 150$ GeV, and leading, subleading jet $p_T > 50$ GeV. The uncertainty was calculated independently for each of the five m_{jj} bins, as shown below in Figure 8.5, but due to poor statistics, the actual uncertainty in each bin was taken from a linear fit of all five points. The same procedure was also used to calculate the uncertainties separately for $N_{\text{jets}} > 2$ events (in bins 11, 12, and 13) and $160 < E_T^{\text{miss}} < 200$ GeV events (for bins 14, 15, and 16). Across all bins, these uncertainties were found to vary from 4% to 8%.

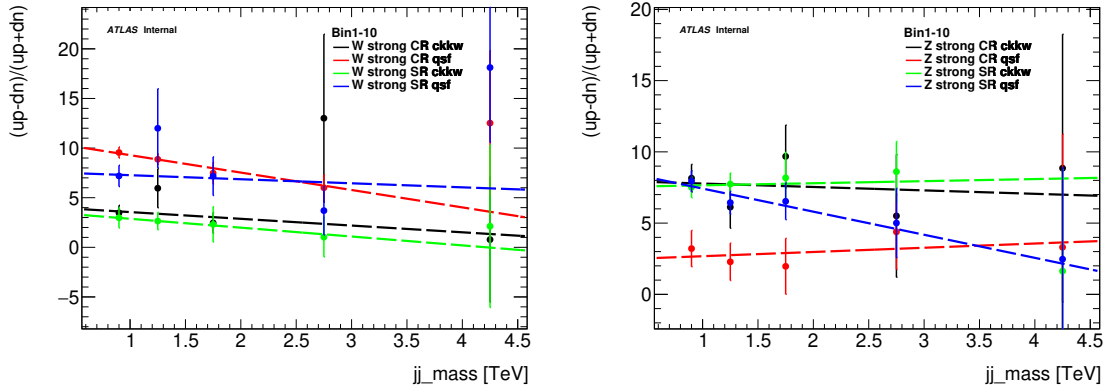


Figure 8.5: Relative resummation (μ_Q , labelled “QSF”) and merging scale (Q_{cut} , labelled “CKKW”) uncertainties calculated for the Sherpa QCD V+Jets samples with a loose truth-level selection applied. The plot on the left shows $W \rightarrow l\nu$ events in the one-lepton control region and zero-lepton signal region, while the plot on the right shows $Z \rightarrow ll$ events in the two-lepton control region and $Z \rightarrow \nu\nu$ events in the zero-lepton signal region. The dashed lines are linear fits to the five m_{jj} points, from which the actual uncertainties are taken due to the limited statistics. These plots show the values used for the first ten bins, which require $E_T^{\text{miss}} > 200$ GeV, $N_{\text{jets}} = 2$: the same values were used for low- and high $\Delta\phi_{jj}$.

PDF uncertainties were also calculated for the QCD V+Jets samples by varying the choice of PDF set. They were found to vary between 1 and 2% across all bins.

Similar uncertainties were also computed for the electroweak V+Jets samples. In the preliminary 139fb^{-1} analysis, LO Sherpa samples were used, but NLO Herwig 7 samples were produced for the final 139fb^{-1} analysis instead. Scale and PDF variations were used from both sets of samples. Herwig varies its scales together internally up and down by a factor of 2, and so Equation 8.2 above is used to calculate a systematic variation. With the LO samples, their scale variations were on the order of 20%, while the Herwig variation is much reduced to around a 5% impact on the event yield. As for the PDF variations, as noted in Section 7.5, the Herwig samples do not have separate PDF variations, and so the ones from LO Sherpa were used and reweighted to NLO in both iterations. These variations had an impact on the order of 1-2% on the total event yield.

An important point is that the impacts quoted above for the different theory systematics are the impact on the nominal event yield in the signal and control regions. But the way that the Monte Carlo prediction in the control region will affect the overall sensitivity of the analysis is through the transfer factor, Equation 8.1. The transfer factor contains the ratio of the background prediction in the signal region to that in the control region. The systematics on these predictions are correlated between signal and control regions, so they should mostly cancel when the ratio is taken. Transfer factor uncertainties are then computed for each systematic by taking ratio of the signal region prediction with the systematic variation applied, $N_{\text{SR}}^{\text{syst}}$, to the control region prediction with the systematic prediction applied, $N_{\text{CR}}^{\text{syst}}$. The uncertainty itself is defined as $N_{\text{SR}}^{\text{syst}}/N_{\text{CR}}^{\text{syst}} - 1$ so that if the two predictions agree perfectly, the systematic will have fully cancelled. This is done separately for each variation and each bin. For the merging and resummation variations in the QCD samples, the transfer factor uncertainty can effectively be read off of Figure 8.5 as the difference between the CR and SR values in each bin. As shown in that plot, for many bins this difference is very small. For the transfer factor uncertainty from the renormalisation and factorisation scales, Figure 8.6 shows that for most bins, it is around 1-2%, even though the impacts on the event yield are much larger. This cancellation illustrates the power of the transfer factor method.

8.2 Reweighting W+Jets to Constrain Z+Jets

The extent to which the uncertainties on the V+Jets transfer factors will cancel is driven by the statistics of the Monte Carlo simulations used to compute them. In addition to MC simulation statistics, the data statistics in the control region also impact the overall sensitivity and uncertainty on the background prediction. If a control region has significantly fewer events than the signal region, extrapolating from that to predict a background in the signal region will be statistically

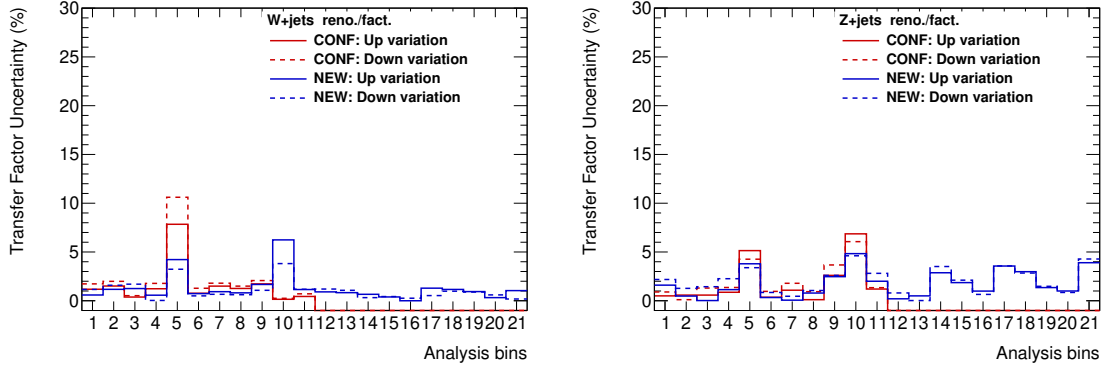


Figure 8.6: Transfer factor uncertainties calculated separately for W+Jets (left) and Z+Jets (right) samples on the renormalisation and factorisation scale variations, evaluated separately in each of the 16 analysis bins (as labelled in Section 6.3.4) and in five bins of a $2 < \Delta\phi_{jj} < 2.5$ validation region (see Section 9.2.2). The red line shows the uncertainties from the preliminary analysis, while the blue shows the uncertainties from the final version.

limited. Ideally, the control region should have a comparable or large number of events in order to avoid limiting the background estimate. In past versions of this analysis, including the preliminary 139 fb^{-1} version, separate transfer factors for the W+Jets and Z+Jets processes were computed from the one- and two-lepton control regions. More $W \rightarrow l\nu$ events will have a reconstructed lepton than a lost lepton, so this background estimate will not be limited. For the $Z \rightarrow \nu\nu$ background estimate to also not be limited, the Z would need to have a roughly even probability of decaying to a pair of charged leptons or a pair of neutrinos, leading to equivalent statistics between the SR and two-lepton control region.

Unfortunately, this is not the case. As described in Chapter 2, the Z boson couples differently to left- and right-handed chiral states. This difference results in a Z boson decaying to a neutrino pair twice as often as it decays to a pair of charged leptons. For a specific lepton flavour l , the $Z \rightarrow l^+l^-$ decay has been measured to occur roughly 3.3% of the time, while the corresponding $Z \rightarrow \nu_l\bar{\nu}_l$ decay occurs about 6.6% of the time [12]. A further issue arises because we only consider electron and muon decays in the 2-lepton control region, but the $Z \rightarrow \nu\nu$ background in the signal region contains decays to all three lepton flavours. That means that the total branching ratio for the $Z \rightarrow \nu\nu$ background is around 20%, compared to just 6.7% for the $Z \rightarrow ee$ and $Z \rightarrow \mu\mu$ processes used to estimate it. We therefore expect the $Z \rightarrow ll$ control region to have significantly fewer events than the real $Z \rightarrow \nu\nu$ background in the signal region, statistically limiting our ability to estimate that background. This can clearly be seen by comparing the Z+Jets predictions in the pre-fit signal region plots shown in Figure 6.7 with those in Figure 8.4 from the Z control region above. Both

plots are normalized appropriately, and there is roughly an order of magnitude difference in the number of events in each⁹⁹.

One possible improvement might be to find a way to use the one-lepton W control region to help constrain the $Z \rightarrow \nu\nu$ background as well. The W coupling does not suffer from the same limitation as the Z : the $W^+ \rightarrow l^+\nu$ branching ratio has been measured to be about 10.86% for each lepton flavour, or about a factor of 3 more than the equivalent $Z \rightarrow l^+l^-$ process [12]. The total Z and W widths, Γ , are not identical, but they are quite close: $\Gamma_Z = 2.4952 \pm 0.0023 \text{ GeV}$ vs $\Gamma_W = 2.085 \pm 0.042 \text{ GeV}$ [12]. And the inclusive W production cross section at $\sqrt{s} = 13 \text{ TeV}$, 190.1 nb, is about 3.25 times as large as the Z production cross section, 58.43 nb [126] [189]. Therefore, we would expect to see around ten times as many events across both $W \rightarrow e\nu$ and $W \rightarrow \mu\nu$ control regions relative to the $Z \rightarrow ll$ region. The selections are not quite identical, so this will not be exactly the case, but around an order of magnitude difference can be seen when comparing Figures 8.1, 8.2, and 8.4 above¹⁰⁰.

This suggests that using the $W \rightarrow l\nu$ control region to derive an estimate for the $Z \rightarrow \nu\nu$ background would be a major improvement. While the two electroweak bosons are very similar, they are not identical thanks to electroweak symmetry breaking: most obviously, the Z is slightly heavier and neutral, while the slightly lighter W is charged. But, as discussed above, there are more subtle differences in the weak flavour-changing and neutral interactions that lead to different coupling strengths. Deriving a $Z \rightarrow \nu\nu$ prediction from $W \rightarrow l\nu$ requires a careful understanding of the differences between the two processes. We do not fully trust the weak boson Monte Carlo to provide this understanding. Among other issues, there are known differences between the jet multiplicity in the W and Z QCD V+Jets samples, as presented in Section 7.4. Those jet veto studies indicated that at high m_{jj} , MC simulations of the two processes begin to behave differently due to apparent technical issues in the generator. Figure 8.7 shows a comparison between both MC and data for events in the Z and W control regions; as can be seen in these plots, the W/Z ratio computed from MC does not always agree with the ratio computed from actual data.

Given these issues, and the limitations of the MC simulation, it is clear that the W simulation alone cannot be relied upon to estimate the Z background. Therefore, we collaborated with a

⁹⁹The impact of this can also be seen in Table 7.1 when discussing MC production; the $Z \rightarrow \mu\mu$ cross section is an order of magnitude smaller than both the $Z \rightarrow \nu\nu$ and $W \rightarrow \mu\mu$ processes in the $140 < p_T^V < 220 \text{ GeV}$ regime, which is where this analysis is most sensitive.

¹⁰⁰In particular, the $S_{\text{MET}} < 4\sqrt{\text{GeV}}$ cut removes a large number of events from the $W \rightarrow e\nu$ control region, which is why there appear to be fewer events than in the $W \rightarrow \mu\nu$. And additional requirements are applied to the $Z \rightarrow ll$ control region to remove multi-boson contamination, reducing the statistics there. Finally, these plots are all pre-fit, meaning that multijet and fake lepton contamination has yet to be taken into account.

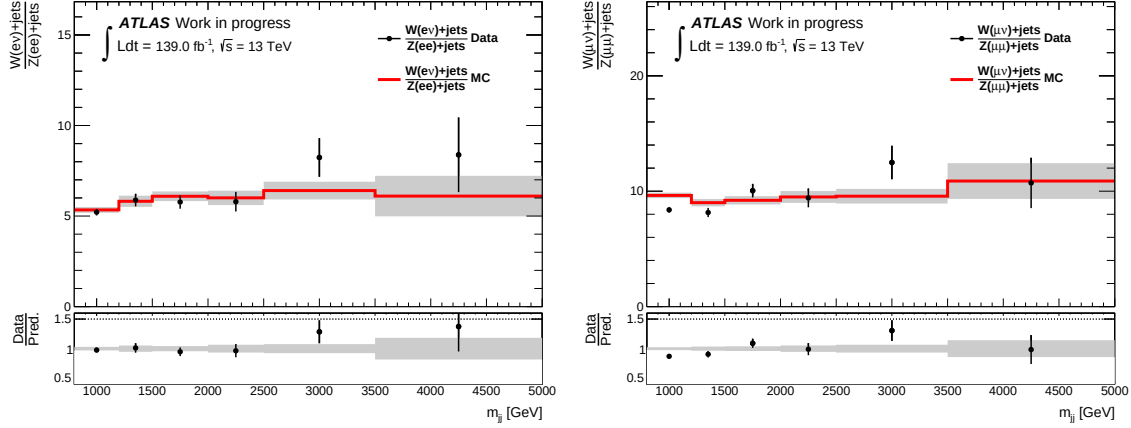


Figure 8.7: Comparison between $W \rightarrow l\nu$ and $Z \rightarrow ll$ yields in the one- and two-lepton control regions, respectively, as a function of m_{jj} . Ratios are taken between the MC prediction in each region as well as the number of data events observed. The left plot shows a comparison of the electron regions, while the right shows a muon comparison. The ratios do not agree within statistical uncertainties for all bins, especially at $m_{jj} > 2.5$ TeV, implying that the MC does not properly model differences between the two processes.

group of theorists, Jonas Lindert, Marek Schöenherr, and Stefano Pozzorini [190], who developed an analytic procedure for reweighting the W Monte Carlo in order to make it Z -like¹⁰¹. To perform this reweighting, the W/Z ratio is calculated perturbatively at next to leading order in both the strong and electroweak forces: that is, considering contributions from $\alpha_s^3\alpha^2$, $\alpha_s^2\alpha^3$, $\alpha_s\alpha^4$, and α^5 diagrams for the V +Jets process. This fixed-order calculation is performed as a function of a single variable x , which in this case is $x = m_{jj}$ due to the observed m_{jj} dependence on the ratio from MC. Equation 8.3 shows how this reweighting calculation is applied. The ratio $\mathcal{R}_{MC}$ is the ratio of Z to W events as computed from the Monte Carlo simulation (as shown in Figure 8.7). The term $\mathcal{R}_{TH}$ is computed by the theorists as a function of m_{jj} , and used to correct that ratio.

$$\frac{d}{dx} \frac{d}{d\vec{y}} \sigma^Z = \frac{\mathcal{R}_{TH}^{Z/W}(x)}{\mathcal{R}_{MC}^{Z/W}(x)} \sigma_{MC}^Z = \frac{1}{\mathcal{R}_{MC}^{Z/W}(x)} \left(\frac{\frac{d}{dx} \sigma_{(N)LO}^Z}{\frac{d}{dx} \sigma_{(N)LO}^W} \right) \frac{d}{dx} \frac{d}{d\vec{y}} \sigma_{MC}^Z \quad (8.3)$$

This reweighting is applied to the Monte Carlo at truth level (before reconstruction) in a minimal selection shown below, with a VBF-like topology that's much looser than the analysis signal region. The ratio $\mathcal{R}_{TH}/\mathcal{R}_{MC}$ is computed in this selection for each bin and applied during the fit process in order to reweight the W to the Z . This ratio is computed and applied separately to the QCD and EW simulations.

¹⁰¹A paper from these authors describing this calculation in detail for the VBF+MET analysis is forthcoming; the referenced paper above describes similar analytic reweighting of V +Jets MC performed for the ATLAS mono-jet analysis [191].

- The leading and subleading jets must have transverse momentum $p_T > 50 \text{ GeV}$ and pseudorapidity $|\eta| < 4.5$.
- The jets must be separated in pseudorapidity by at least $\Delta\eta_{jj} > 2.5$.
- The (truth) weak boson must have a transverse momentum $p_T^V > 150 \text{ GeV}$.

To enable this collaboration, the theorists were designated ATLAS [Analysis Consultants and Experts \(ACEs\)](#) and given access to the V+Jets MC simulations developed for this analysis in Chapter 7. Not all of the NLO diagrams listed above are included in the ATLAS V+Jets samples, but they provided us with corrections to the samples that included them, as well as an inclusive reweighting for the cross section up to [NNLO](#). The theorists then computed the reweighting function $\mathcal{R}_{\text{TH}}(m_{jj})$, as well as several uncertainties on that ratio, and provided these results for the final full run 2 version of the analysis. Figure 8.8 shows the final reweighting function, as well as the uncertainties on it, which are described in more detail in the sections below. As can be seen in the plots, both QCD and EW reweighting functions appear to have non-trivial dependence on m_{jj} . The ratio in the low- m_{jj} EW bins, in particular, is relatively far from unity, which illustrates the necessity of these calculations in endeavouring to use the W to constrain the Z .

8.2.1 Theoretical Uncertainties

One major advantage of reweighting the W to the Z is that it allows the theoretical uncertainties on the V+Jets processes described in Section 8.1.3 to be fully correlated. This means that when the ratio is taken, these uncertainties will maximally cancel, which significantly reduces their impact. Still, there are other systematic uncertainties on the reweighting itself which need to be considered as part of this procedure, some of which are computed by the [ACEs](#) while others are computed by the analysis team. The first two sources of uncertainty on the reweighting are theoretical uncertainties on the fixed-order QCD and QCD-EW mixing terms, labelled $\delta R_{\text{QCD}}^{Z/W}$ and $\delta R_{\text{mix}}^{Z/W}$. These terms were both computed directly by the [ACEs](#) for both the strong and electroweak V+Jets processes.

8.2.2 Parton Shower Uncertainties

The third source of uncertainty on the reweighting arises from the parton shower model(s) used in the Monte Carlo generators. The reweighting calculations were done using fixed-order matrix element calculations for the W and Z processes. As described in Section 7.1, the V+jets MC is generated using a mix of matrix element calculations and a parton shower process, which adds additional

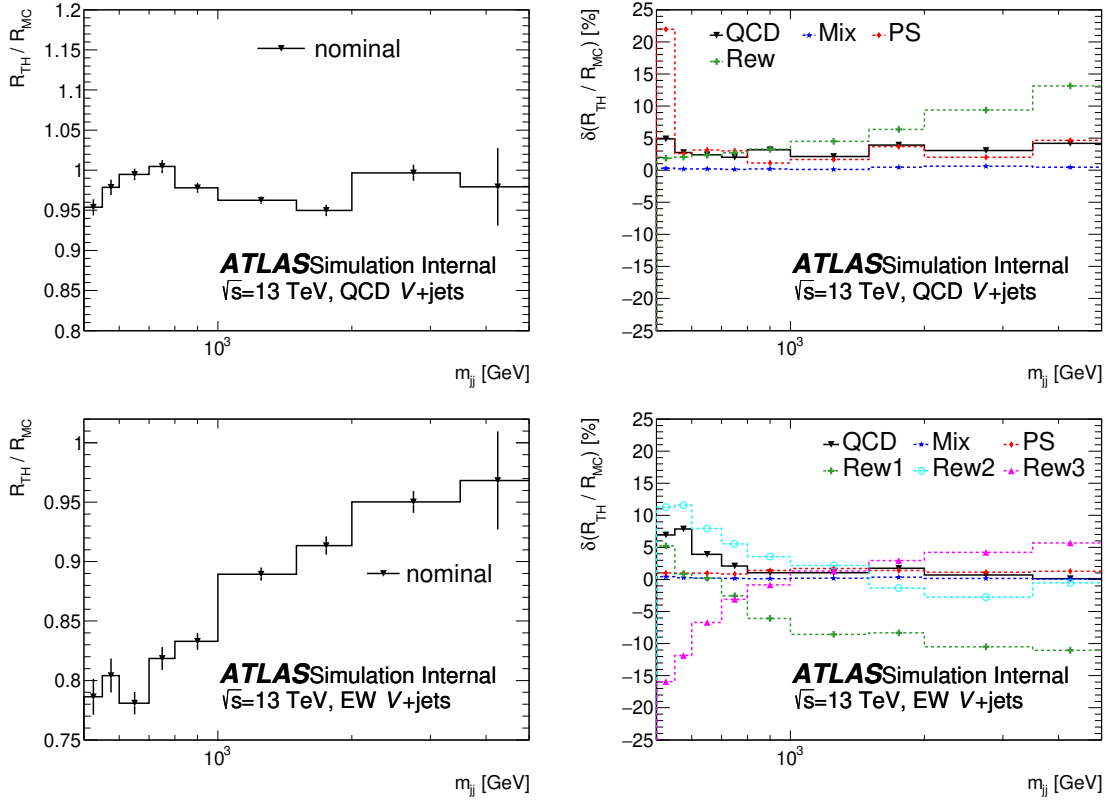


Figure 8.8: Plots of the theory calculations used to “reweight” W +jets MC simulation to Z +jets, performed separately for QCD (top) and EW (bottom) processes, performed as a function of m_{jj} . The left set of plots show the ratio of the reweighting function $\mathcal{R}_{TH}$, computed by the ACEs, with $\mathcal{R}_{MC}$, the Z/W ratio from MC. The right plots show the systematic uncertainties on this calculation: the QCD, QCD-EW mixing, parton shower, and reweighting uncertainties, definitions of which can be found in Sections 8.2.1, 8.2.2, and 8.2.3.

hadronic decays (and therefore jets) to an event beyond what is modelled directly in the core matrix element process. The choice of parton shower model generally leads to a systematic uncertainty, because individual generators implement the parton shower process differently. In this case, there may be differences between Z and W processes in the parton shower model used by a particular generator, and these will not be accounted for in the fixed-order analytic theory calculations.

The QCD V+Jets samples use the Sherpa generator and parton shower model [155], and an uncertainty on the parton shower model is not available. However, the generator configuration, as well as the samples, were shared with the ACEs who did compute a parton shower uncertainty for the QCD processes¹⁰². For the electroweak V+Jets samples, which were generated using Herwig, a

¹⁰²As noted in Chapter 7, Marek Schöenherr is also a Sherpa author, and had assisted in developing the k_t -based

parton shower systematic uncertainty was calculated by varying the parton shower model used in the event generation process from the default dipole recoil-based model with an Angular ordered model [182]. The difference between the two parton shower models was assigned as the systematic.

8.2.3 Reweighting Uncertainties

The final class of uncertainties arises due to the fact that the loose truth-level selection used for the reweighting differs from the VBF signal region used by the analysis. If the W and Z processes behave differently in the reweighting selection than they do in the signal (or control) region(s), the reweighting function $R_{\mathcal{T}\mathcal{H}}$ will not be entirely correct when it is applied in the fit. Studies were done to understand any such differences, and separate QCD and EW reweighting uncertainties assigned to cover the different issues that were observed. For the QCD V+Jets samples, the main difference seen between W and Z concerns the third jet veto efficiency, as presented in Section 7.4. Due to issues merging events with 3- and 4- parton matrix elements at event generation, the jet multiplicity varies as a function of m_{jj} between the processes. After the jet veto is applied, the two processes appear to agree well. Since this is a nonphysical effect, it was agreed with the ACEs to apply the third jet veto to the QCD Monte Carlo before performing the reweighting. The jet veto efficiency is then computed using W and Z samples generated with only 2-parton matrix elements, and assigned as a systematic uncertainty.

In the electroweak V+Jets samples, the reweighting uncertainties cover a different phenomenon altogether. Nontrivial contamination from electroweak diboson production was observed with the reweighting selection applied. This contamination grows with $\Delta\phi_{jj}$, becoming significant once $\Delta\phi_{jj} > 2$. While this region is not directly used in the analysis, the correction is done inclusively in $\Delta\phi_{jj}$, meaning there is a potential impact which an uncertainty is needed to quantify. In addition to the inclusive correction, the ACEs also produced the electroweak $\mathcal{R}_{\text{TH}}$ in $0 < \Delta\phi_{jj} < 1$, $1 < \Delta\phi_{jj} < 2$, and $\Delta\phi_{jj} > 2$ bins. For each event, the electroweak reweighting uncertainty is assigned as the difference between the inclusive correction and the binned correction for the $\Delta\phi_{jj}$ of each event. This leads to the three separate EW reweighting uncertainties seen in Figure 8.8.

8.3 Misidentified Leptons in W Control Region

The transfer factor procedure described in the previous two sections makes one crucial assumption: that any differences between data and MC simulation in the control regions will be mirrored in

m_{jj} -slicing configuration for Sherpa.

the signal region. If the one- and two- lepton control regions only contain V+Jets events, this will of course be the case. However, in reality, no region will ever be 100% pure in a single process. Therefore, any contamination from other backgrounds in the control regions must be understood and estimated, in order to use those control regions for the V+Jets background estimation.

One source of contamination in the one-lepton W control region arises from misidentified or “fake” lepton events: a QCD-like multi-jet event where a jet is incorrectly identified as a charged lepton when reconstructing data from the detector. The contamination from these misidentified lepton events is small, on the order of a few percent. However, it arises from detector effects and lepton (mis)-identification, and therefore is not modelled directly using Monte Carlo. That means that if no attempt to estimate this contamination is made, a small excess of data events will be seen in the one-lepton control regions that will be due to misidentified multijet events. This excess will not be a result of any V+Jets modelling problem, and therefore will not be expected in the signal region, as leptons are vetoed there. Therefore, in order to prevent the V+Jets transfer factors from being affected by this contamination, an estimate of the misidentified multijet background must be made.

The misidentified multijet estimate is performed using a transfer factor approach that is very similar to the one used for V+Jets estimation overall. The technique was briefly described in Section 8.1.1, and is expanded upon here. Leptons must pass an identification requirement to be considered real, as mentioned above and in Section 6.3.2. In order to select events with leptons that are likely to have been misidentified, this identification can be inverted to find leptons that pass a looser identification but fail the tighter definition used to define the control regions. The rest of the one-lepton control region selection is then applied to create an “anti-ID” version of the W control region, as opposed to the normal or “ID” region. Then, a variable is identified which can be used to divide both the anti-ID and ID regions into fake-enriched and non-fake-enriched sections of phase space. This creates four regions; the amount of fake lepton contamination in each is determined as the difference between data and MC. A transfer factor can be computed as the ratio of fake lepton contamination between the enriched and non-enriched parts of the anti-ID region. This transfer factor is then used to normalize the fake lepton estimate from the enriched part of the ID region, in order to predict the amount of contamination in the non-enriched part of the ID region.

This section describes in detail how this is done and what variables are used for both the $W \rightarrow e\nu$ and $W \rightarrow \mu\nu$ control regions. While a jet can be improperly identified as either an electron or muon, as described in Chapter 3.4, electron and muon identification and reconstruction are quite different, and so the anti-ID definitions and variables used are different. The fake electron and fake muon

estimates are described separately in detail below.

8.3.1 Fake Electron Estimate

The fake electron estimate is performed using the missing transverse momentum significance, labelled S_{MET} . This variable, defined in Equation 8.4 below, can be used to quantify the likelihood of whether or not a given event's $E_{\text{T}}^{\text{miss}}$ is really caused by an undetected neutral particle escaping the detector. The square root of the observed total transverse momentum in the event approximates the $E_{\text{T}}^{\text{miss}}$ resolution, and therefore a low value of S_{MET} implies the $E_{\text{T}}^{\text{miss}}$ is unlikely to be real [188].

$$E_{\text{T}}^{\text{miss}} \text{ sig.} = S_{\text{MET}} = \frac{E_{\text{T}}^{\text{miss}}}{\sqrt{p_{\text{T}}(l) + \sum_{i=0}^{N_{\text{jets}}} p_{\text{T}}(j_i)}} \quad (8.4)$$

A multi-jet event with a misidentified lepton that does not contain a real leptonic W boson decay will also not contain a real, prompt neutrino. Therefore, both the $E_{\text{T}}^{\text{miss}}$, and the significance of the $E_{\text{T}}^{\text{miss}}$ as defined above is likely to be much lower than events with a real charged lepton (and real neutrino). As a result, we can use this variable to separate the $W \rightarrow e\nu$ control region into a fake-enriched low- S_{MET} region and a non-fake-enriched high- S_{MET} region. As mentioned in Section 8.1.1, a cut on $S_{\text{MET}} > 4\sqrt{\text{GeV}}$ is therefore applied to the $W \rightarrow e\nu$ CR, while events with $S_{\text{MET}} < 4\sqrt{\text{GeV}}$ are used for the fake-enriched low- S_{MET} region.

For the fake electron estimate, an anti-ID region is constructed by inverting the electron ID working point. In the one-lepton control region, it was mentioned above that electrons must pass the “tight” working point for an event to be accepted. To define the corresponding anti-ID region, events must contain an electron which fails the “tight” ID requirement but passes the “loose” ID requirement. Since a “loose” electron is enough to veto an event from the signal region, this is equivalent to saying that events must have enough evidence of an electron to be vetoed from the SR but not enough to be accepted into the $W \rightarrow e\nu$ CR. All other cuts from the $W \rightarrow e\nu$ control region are then applied. Figure 8.9 shows plots of data and MC simulation for this electron anti-ID region. As can be seen in these plots, a very large excess between data and MC is seen in the low- S_{MET} region, exactly as expected since this area of phase space should be especially fake-enriched. Subtracting the MC simulation from data can be used to estimate the number of multi-jet events with fake electrons, as shown in the bottom set of plots.

Transfer factors are then computed by taking the ratio of fake electron events with $S_{\text{MET}} > 4\sqrt{\text{GeV}}$ to those with $S_{\text{MET}} < 4\sqrt{\text{GeV}}$. These transfer factors are labelled R_S , and are passed as parameters to the fit. There, they will be applied simultaneously with the transfer factors for the

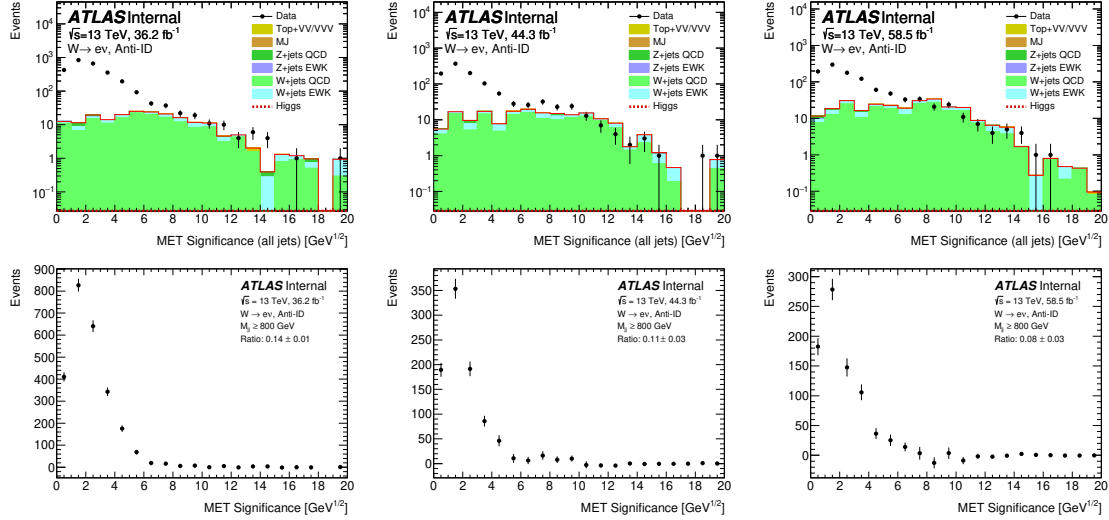


Figure 8.9: Plots of the inclusive missing transverse momentum significance, also referred to as S_{MET} , for the $W \rightarrow e\nu$ anti-ID region. The top set of plots show the (dis-)agreement between data and Monte Carlo prediction, while the bottom set of plots show the template shape produced by subtracting MC from the data in the top plots. The fake electron transfer factor is then evaluated by taking a ratio in the template shape around $4\sqrt{\text{GeV}}$. Data from 2015/16, 2017, and 2018 is plotted separately; while the statistics vary, in general the three templates appear quite similar.

V+Jets background estimate in order to determine the true number of fake lepton events in the $W \rightarrow e\nu$ control region. The plots in Figure 8.9 show separate calculations for the 2015/16, 2017, and 2018 datasets, as pile-up conditions (and the amount of QCD multi-jet events) varied from year to year. However, as can be seen in these plots, the three fake electron shapes appear consistent from year to year despite these conditions. Additionally, these plots show inclusive fake electron distributions, ignoring any potential dependence on either m_{jj} or $\Delta\phi_{jj}$ dependence, the variables in which the analysis is binned. A study was done to explore whether or not the fake electron transfer factor R_S depends on either the data-taking period, or any of these quantities.

The results of this study are shown in Figure 8.10. The fake electron transfer factors were computed separately for each dataset as a function of m_{jj} (inclusive in $\Delta\phi_{jj}$), as well as separately as a function of $\Delta\phi_{jj}$ (inclusive in m_{jj}). Only events with $N_{\text{jets}} = 2$ were considered here, due to limited statistics. Several conclusions can be seen from these plots: first, that R_S does not appear to have any dependence on the data-taking period in any of the bins. Second, there does not appear to be any observable $\Delta\phi_{jj}$ dependence, as the ratios appear roughly equal between the two $\Delta\phi_{jj}$ bins. And third, despite poor statistics in $m_{jj} > 3.5 \text{ TeV}$, there *does* appear to be some m_{jj} dependence in the first four bins, with the ratio appearing to increase with m_{jj} . The source of this dependence was

traced to the transverse momentum of the (fake) electron, $p_T(l)$. As m_{jj} increases, the number of misidentified events with high “electron” p_T decreases, which from Equation 8.4 leads to an increase in the relative fraction of events with high S_{MET} . Finally, an additional study verified that there is no significant difference in the fake electron transfer factors between $W \rightarrow e^+\nu$ and $W \rightarrow e^-\nu$ anti-ID events.

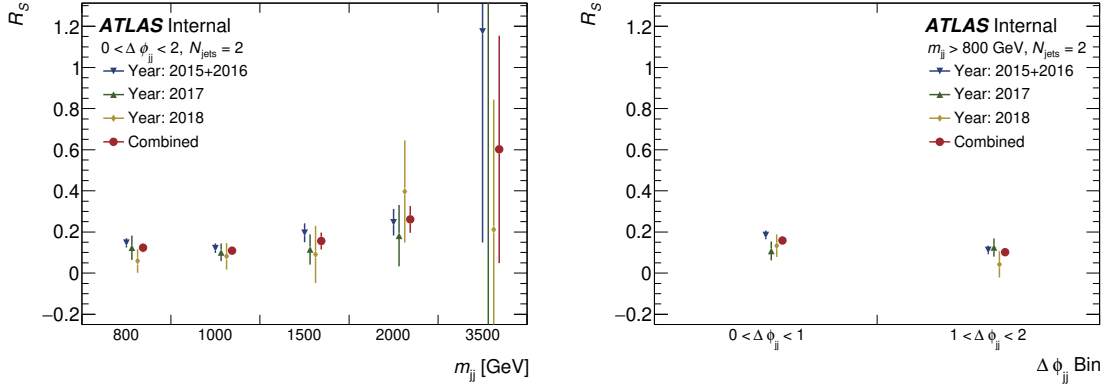


Figure 8.10: Plots of the fake electron transfer factor, R_S , shown as a function of m_{jj} (left) and $\Delta\phi_{jj}$ (right). While $W \rightarrow e\nu$ anti-ID statistics are quite limited at high m_{jj} , in general, the ratio R_S seems to increase as a function of m_{jj} . On the other hand, the ratio appears consistent between the low and high $\Delta\phi_{jj}$ bins. Transfer factors from the 2015/16, 2017, and 2018 datasets all appear to be in good agreement with each other.

As can be seen in these plots, the $W \rightarrow e\nu$ anti-ID region has relatively low statistics. Therefore, since the ratio R_S only appears to depend on m_{jj} , the decision was made to compute just six transfer factors from the entire full run 2 dataset: one for events with $N_{\text{jets}} > 2$, and then one for each m_{jj} bin that is inclusive in $\Delta\phi_{jj}$. Figure 8.11 shows the inclusive $W \rightarrow e\nu$ anti-ID region with data and MC with the full run 2 dataset that was used for this calculation. Note that, because different $E_{\text{T}}^{\text{miss}}$ requirements were experimented with over the course of the analysis, these transfer factors were computed separately in Table 8.1 for both $E_{\text{T}}^{\text{miss}} > 160 \text{ GeV}$ and Table 8.2 for $E_{\text{T}}^{\text{miss}} > 200 \text{ GeV}$; the latter were ultimately used. The statistical uncertainties in these tables are the uncertainty on the efficiency of the $S_{\text{MET}} < 4 \text{ GeV}$ cut, and are therefore computed using binomial statistics. The statistical uncertainties on each transfer factor from this calculation are then assigned as a systematic during the fit procedure.

A final study was performed in which alternate definitions of the significance S_{MET} were considered. In recent years, ATLAS has moved to a more sophisticated “object-based” definition of missing transverse momentum significance. Instead of just using the scalar sum of the transverse

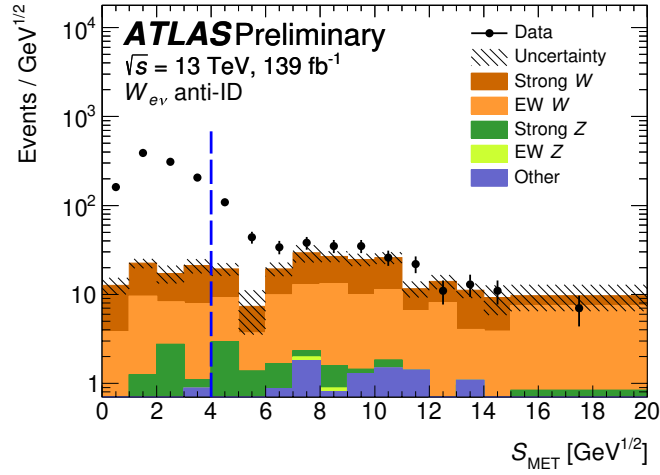


Figure 8.11: Plot of the inclusive missing transverse momentum significance, S_{MET} , in the $W \rightarrow e\nu$ anti-ID region. Unlike Figure 8.9, S_{MET} here is plotted with all three data-taking periods combined (so the full run 2 dataset), as the fake electron transfer factor was observed to be consistent from year to year. The blue line indicates the $4\sqrt{\text{GeV}}$ cut, about which the transfer factor is calculated.

Bin	2015/16 R_S	2017 R_S	2018 R_S	Inclusive R_S
$0.8 < m_{jj} < 1.00 \text{ TeV}$	0.15 ± 0.02	0.12 ± 0.06	0.06 ± 0.06	0.12 ± 0.02
$1.0 < m_{jj} < 1.50 \text{ TeV}$	0.12 ± 0.02	0.10 ± 0.04	0.08 ± 0.06	0.11 ± 0.02
$1.5 < m_{jj} < 2.00 \text{ TeV}$	0.20 ± 0.05	0.12 ± 0.07	0.09 ± 0.14	0.16 ± 0.04
$2.0 < m_{jj} < 3.50 \text{ TeV}$	0.25 ± 0.06	0.18 ± 0.15	0.40 ± 0.25	0.26 ± 0.06
$m_{jj} > 3.50 \text{ TeV}$	1.18 ± 1.03	-0.43 ± 3.28	0.21 ± 0.63	0.60 ± 0.55
$N_{\text{jets}} > 2$	0.19 ± 0.04	0.07 ± 0.09	0.16 ± 0.08	0.15 ± 0.03
Unbinned	0.16 ± 0.01	0.11 ± 0.03	0.10 ± 0.04	0.13 ± 0.01

Table 8.1: Pre-fit fake lepton estimate transfer factors, R_S , computed using all three datasets and with a $E_{\text{T}}^{\text{miss}}$ cut at 160 GeV. Ratios were evaluated separately for the five m_{jj} bins, inclusive in $0 < \Delta\phi_{jj} < 2$, as well as for all events with $N_{\text{jets}} > 2$. The final row shows a transfer factor computed inclusively for all data taken in 2015/16, 2017, 2018, and across the full run 2 dataset.

Bin	2015/16 R_S	2017 R_S	2018 R_S	Inclusive R_S
$0.8 < m_{jj} < 1.00 \text{ TeV}$	0.30 ± 0.06	0.21 ± 0.05	0.08 ± 0.12	0.23 ± 0.05
$1.0 < m_{jj} < 1.50 \text{ TeV}$	0.21 ± 0.05	0.03 ± 0.11	0.17 ± 0.10	0.16 ± 0.04
$1.5 < m_{jj} < 2.00 \text{ TeV}$	0.16 ± 0.09	-0.03 ± 0.06	-0.10 ± 0.28	0.07 ± 0.08
$2.0 < m_{jj} < 3.50 \text{ TeV}$	0.36 ± 0.15	0.01 ± 0.14	-0.12 ± 0.34	0.19 ± 0.11
$m_{jj} > 3.50 \text{ TeV}$	2.76 ± 3.49	8.29 ± 68.78	-0.27 ± 0.48	0.17 ± 0.69
$N_{\text{jets}} > 2$	0.14 ± 0.06	0.34 ± 0.12	0.16 ± 0.16	0.21 ± 0.06
Unbinned	0.22 ± 0.03	0.14 ± 0.05	0.11 ± 0.07	0.18 ± 0.02

Table 8.2: Pre-fit fake lepton estimate transfer factors, R_S , computed using all three datasets and with a $E_{\text{T}}^{\text{miss}}$ cut at 200 GeV; see Table 8.1 for more details.

momentum of the event to approximate the E_T^{miss} resolution, the object-based definition considers each object which enters each event's E_T^{miss} individually and computes an event-by-event likelihood function [188]. This more sophisticated approach has been found to be more powerful in determining whether or not the E_T^{miss} in an event is real; therefore, it was also evaluated for the fake electron estimate. However, it was found that requiring $S_{\text{MET}}^{\text{obj}} > 4\sqrt{\text{GeV}}$ had significantly lower efficiency than using the definition in Equation 8.4, reducing the statistics of the $W \rightarrow e\nu$ control region. Figure 8.12 shows how the efficiency of the $S_{\text{MET}}^{\text{obj}}$ cut varies with the threshold; lowering the cut to $2.6\sqrt{\text{GeV}}$ would give comparable efficiency to the Equation 8.4 approach. Unfortunately, a cut of $2.6\sqrt{\text{GeV}}$ also leads to lower rejection power, as the ratio R_S increases, meaning that there is significant fake contamination in the non-enriched part of the anti-ID region. Given this, the decision was made to stick with the Equation 8.4 approach presented above.

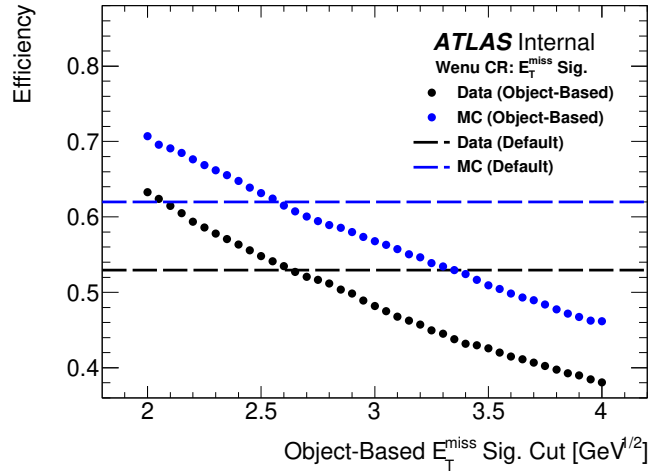


Figure 8.12: Study demonstrating the impact of using the object-based definition of S_{MET} for the fake electron estimate instead of the definition in Equation 8.4. The dashed lines indicate the efficiency of the $S_{\text{MET}} < 4\text{GeV}$ cut in the $W \rightarrow e\nu$ control region on both data (black) and Monte Carlo (blue). The efficiency of this cut was found to be significantly lower when using the object-based definition; the blue and black data series show the effect on cut efficiency of continually lowering the object-based cut from 4 to $2\sqrt{\text{GeV}}$. Requiring $S_{\text{MET}}^{\text{obj}} > 2.6\sqrt{\text{GeV}}$ was found to be equivalent to $S_{\text{MET}} > 4\sqrt{\text{GeV}}$.

8.3.2 Fake Muon Estimate

A jet can be misidentified as a muon as well as an electron, leading to multijet contamination of the $W \rightarrow \mu\nu$ control region. Most jets are absorbed in ATLAS's hadronic calorimeter layers, and do not make it into the muon system, and so this effect is expected to be small compared to the

fake electron background. As a result, past versions of this analysis did not include a dedicated estimate of this contamination. To verify that this is actually the case, the full run 2 background now includes a dedicated fake muon estimate.

First, an attempt was made to estimate the misidentified muon contribution using the S_{MET} -based approach described above for electrons. By inverting the muon identification requirements to select muons which would pass the “very loose” selection working point but fail the “medium” working point, a $W \rightarrow \mu\nu$ anti-ID control region was defined¹⁰³. Figure 8.13 shows plots of the significance S_{MET} , and the difference between data and MC simulation, in this region. Unfortunately, as can be seen in the plots, no noticeable excess between data and MC is seen. This suggests that, unlike the electron version, this anti-ID region is not fake-enriched and therefore cannot be used to calculate a transfer factor. Instead, to get a very rough estimate of the size of the fake muon background, the difference between data and MC was measured in the $W \rightarrow \mu\nu$ control region. The CR was normalized to unity in the $S_{\text{MET}} > 4\sqrt{\text{GeV}}$ region (where the contribution from multijet events should be small), and then the difference between data and MC taken only for events with $S_{\text{MET}} < 4\sqrt{\text{GeV}}$. This procedure approximated the fake muon contamination to be on the order of 2 (when requiring $E_{\text{T}}^{\text{miss}} > 200 \text{ GeV}$) to 3% (when requiring $E_{\text{T}}^{\text{miss}} > 160 \text{ GeV}$) of data in the low-significance part of the $W \rightarrow \mu\nu$ control region. That suggests the contamination in the entire CR from fake muon events is quite small.

A more robust procedure was developed to estimate the fake muon contribution for the final 139fb^{-1} result. The two-lepton $Z \rightarrow \mu\mu$ region has a looser muon definition than the one-lepton $W \rightarrow \mu\nu$ region; it accepts “loose” muons while the $W \rightarrow \mu\nu$ CR requires “medium” leptons. Instead of inverting the W muon definition, the Z muon definition was inverted in order to produce a tighter muon anti-ID region. This resulted in selecting only events with less isolated and more jet-like muons, leading to a region with worse statistics but one that is considerably more fake-enriched than the first attempt shown in Figure 8.13. A variety of variables were then inspected in order to determine what could best be used to compute a transfer factor, including the missing transverse momentum $E_{\text{T}}^{\text{miss}}$, the significance S_{MET} , the muon transverse momentum p_{T} , as well as a measure of the muon’s isolation. In the end, the quantity that was determined to be the most useful was the transverse mass, m_{T} . The transverse mass, defined below in Equation 8.5, is the invariant mass of the sum of the muon and the missing transverse momentum.

¹⁰³As with the $W \rightarrow e\nu$ anti-ID definition, this $W \rightarrow \mu\nu$ anti-ID region would contain events with enough evidence of a muon to be vetoed from the zero-lepton signal region, but not enough to be accepted into the one-lepton control region.

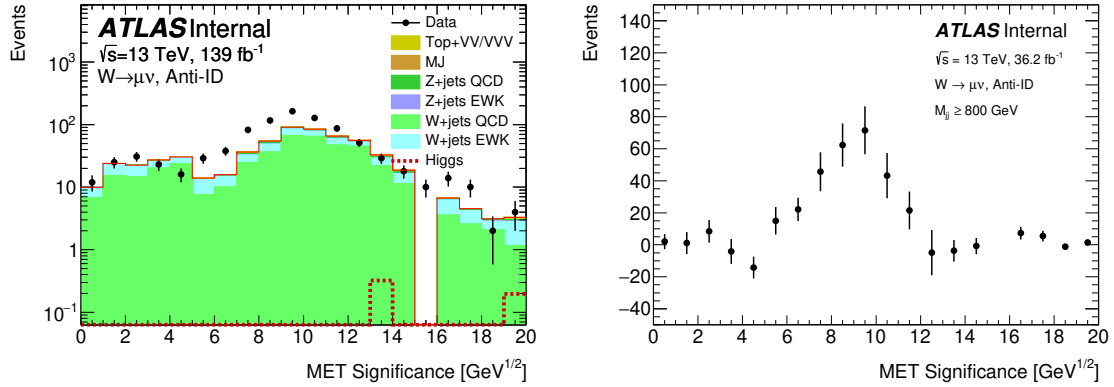


Figure 8.13: Plots of the inclusive muon missing transverse momentum significance, S_{MET} , in a hypothetical $W \rightarrow \mu\nu$ anti-ID region, defined by inverting the muon identification requirement in much the same way as the $W \rightarrow e\nu$ anti-ID region. The plot on the left shows both data and Monte Carlo simulation, while the plot on the right shows a template shape produced from subtracting MC from data. Unlike Figure 8.9, good agreement between data and Monte Carlo is seen in this region, indicating that it is not fake-enriched.

$$m_T = \sqrt{2p_T(l)E_T^{\text{miss}}(1 - \cos \Delta\phi_{l,\text{miss}})} \quad (8.5)$$

For events with real muons, and therefore real W boson decays, m_T should be large (and close to the real mass of the W boson). For multi-jet events with fake muons (and fake E_T^{miss}), the m_T should be quite lower, in much the same way as the significance. Figure 8.14 shows plots of m_T in the updated, tighter muon anti-ID region. A small but significant excess can be seen in the region $m_T < 20 \text{ GeV}$ due to the presence of fake muon events. Muon transfer factors, labeled R_M^{104} can then be computed by subtracting from MC from data and taking the ratio around 20 GeV in analogy to what was done for the fake electron transfer factor. An inclusive muon transfer factor, for the full run 2 dataset, was measured to be 0.29 ± 0.15 . This very large statistical uncertainty arises due to the low statistics in the anti-ID region. An attempt was then made to understand if the fake muon contribution depends on either m_{jj} , $\Delta\phi_{jj}$, N_{jets} , or the data-taking period, using the same analysis strategy described above for the fake electron estimate. Figure 8.15 shows R_M computed as a function of m_{jj} and $\Delta\phi_{jj}$ for all three data-taking periods. However, the already-poor statistics get worse once a binning is applied to the anti-ID region, and this means that we cannot conclude one way or another whether the fake muon contribution has any dependence on these variables. Therefore, the inclusive transfer factor was used for all bins as an input to the fit process, with its

¹⁰⁴Since this is derived from the transverse mass, instead of significance, and is for muons.

large uncertainty assigned as a systematic.

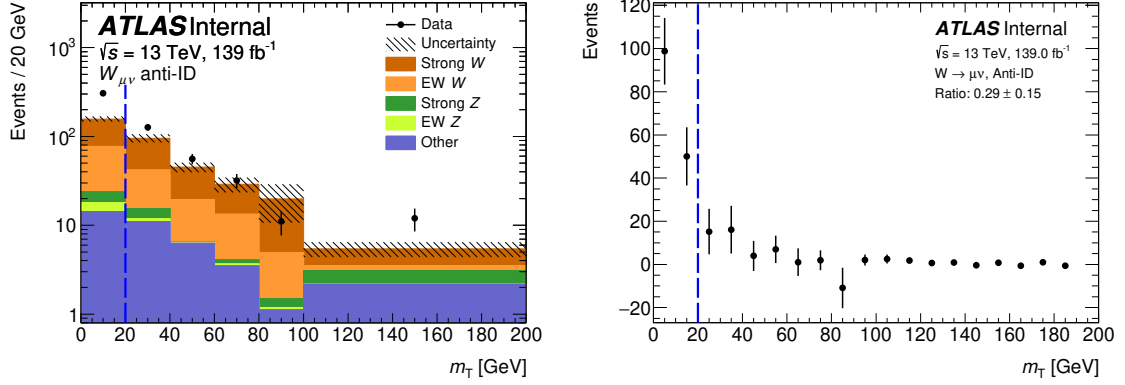


Figure 8.14: Plots of the inclusive transverse mass, m_T , in the $W \rightarrow \mu\nu$ anti-ID region. The plot on the left shows the agreement between data and Monte Carlo, while the plot on the right shows the template shape produced from subtracting MC from data. While the statistics are quite low, an excess can be seen in the $m_T < 20$ GeV region on the left of the dashed blue lines, indicating the presence of fake muon events.

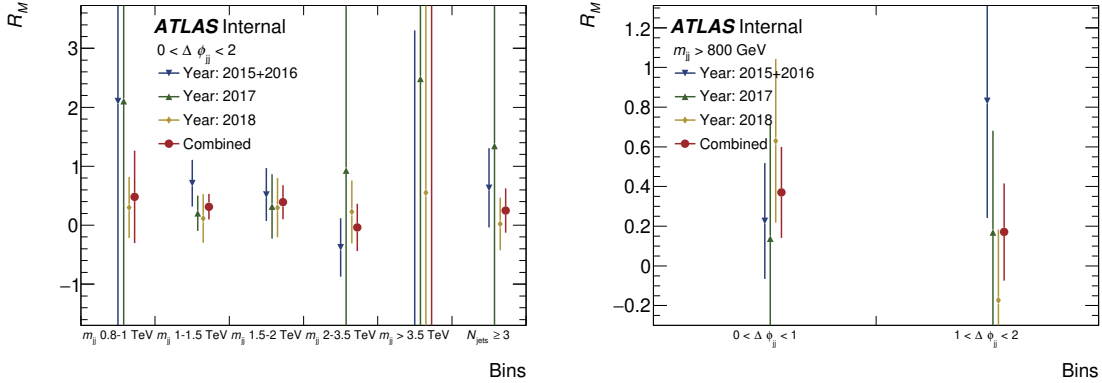


Figure 8.15: Plots of the fake muon estimate transfer factor, R_M , shown as a function of m_{jj} (left) and $\Delta\phi_{jj}$ (right), in analogy to the fake electron plots shown in Figure 8.10. Due to poor statistics, it is difficult to make a definitive conclusion about any dependence on either variable.

8.4 Multijet Estimate: Rebalance and Smear

QCD-like multi-jet events can also contaminate the zero-lepton signal region. Unlike the fake lepton contamination described in Section 8.3, contamination in the signal region occurs due to events with fake missing transverse momentum, often due to jet energy resolution mismeasurement or other detector effects. Because this E_T^{miss} is not real, as the missing transverse momentum cut is

increased, the size of the multijet contamination is significantly reduced. Further, requiring that the two leading jets not be back-to-back ($\Delta\phi_{jj} < 2$) also removes a large fraction of the multijet background. Still, as with the fake leptons, this multijet background must be measured carefully using a partially data-driven approach in order to ensure it is properly constrained during the fit.

Two independent procedures were developed for this analysis in order to estimate the multijet background. The first method, known as “rebalance and smear”, is briefly described in this section¹⁰⁵, and was also used in past versions of the VBF+MET analysis [135] and further expanded upon for the full run 2 iteration. While it is quite sophisticated, the method is also quite complex, and results in relatively large systematic uncertainties for such a small background. Therefore, for the final version of the full run 2 analysis, a second procedure using transfer factors was developed and is described below in Section 8.5. Both methods are used together in different parts of the signal region, and in general, were found to agree quite well with each other despite being very different.

The idea behind rebalance and smear is to take reconstructed QCD-like events that may or may not contain real E_T^{miss} , and simulate the effects of the jet energy resolution mismeasurement that gives rise to multijet contamination in order to produce a population of events with fake E_T^{miss} . This is accomplished by first “rebalancing” the momenta of an event’s jets, shifting each jet p_T within its uncertainty in order to bring that event’s E_T^{miss} to 0. Then, the jets are “smeared” according to the jet response and re-evaluated in order to produce a population of events with fake E_T^{miss} . Because one of the two leading jets is often a pileup jet, this procedure is performed separately for events that have no pileup jets with $p_T > 50$ GeV (non pileup jets are referred to as “hard scatter” jets, and so this is known as the **Hard Scatter (HS)** multijet component) and for those that do (known as the **Hard Scatter and Pile-Up (HS+PU)** component). The relative contribution of each component to the total multijet estimate is then normalized in a fit to data. Finally, the combined estimate is normalized again in two dedicated multijet-enriched validation regions before being used in the fit. These steps, and the uncertainties which are incurred along the way, are all described in more detail below.

8.4.1 Generating Events with Fake E_T^{miss}

The diagram in Figure 8.16 outlines the three main steps of the rebalance and smear procedure prior to normalization. Rebalance and smear can in principle be ran over either data or a MC simulation.

¹⁰⁵Readers with access to internal ATLAS documents should consult Sections 5.6 and 5.7 of the VBF+MET internal note (ATL-COM-PHYS-2020-310, <https://cds.cern.ch/record/2717301>), where the entire procedure is presented in considerably more detail.

In past versions of the analysis, and in the preliminary full run 2 result, single-jet triggered data was used as the input sample for the procedure. Pile-up tagging using JVT and FJVT were used to separate the **HS+PU** and **HS** components. In the final full run 2 result, the procedure was modified to instead use reconstructed di-jet Monte Carlo events generated using Pythia 8. An advantage of using MC is that truth matching can be performed to identify pile-up jets: any reconstructed jet not within $\Delta R < 0.1$ of a truth jet is considered to be a pile-up jet. In this way the input sample can be divided between events with two truth-matched jets that have additional $p_T > 50$ GeV pileup (**HS+PU**) and events that do not (**HS**). These subsamples are then ran separately through the rebalance and smear procedure.

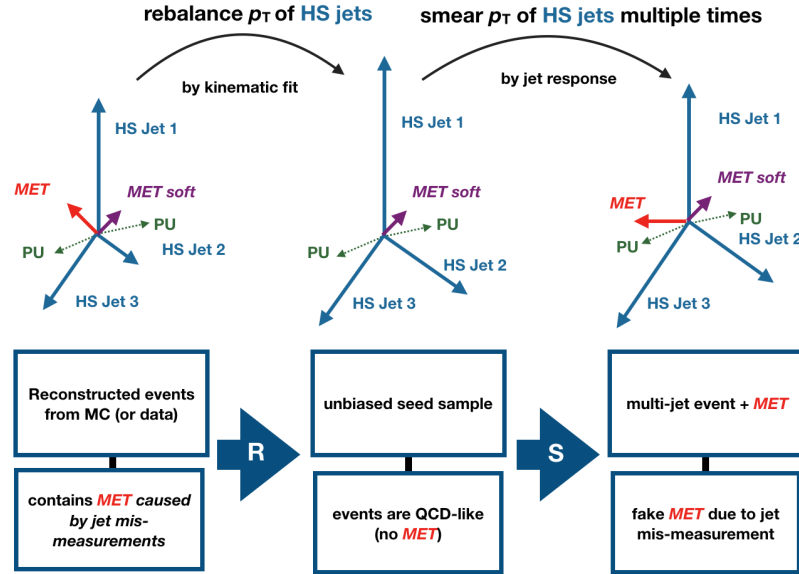


Figure 8.16: Sketch outlining the rebalance and smear procedure used to perform the multijet background estimate. Reconstructed QCD events drawn from Monte Carlo (or single-jet triggered data) are “rebalanced” to produce a population of events with no E_T^{miss} . Then, the momentum of each jet is smeared in order to simulate the kind of jet mis-measurement that can lead to fake E_T^{miss} .

Rebalancing the input samples is then done using the software package “KinFitter”. As the name suggests, this tool is designed to perform fits with external kinematic constraints [192]. For the purposes of the multijet estimate, the absolute value of the scalar sum of the transverse momentum of the truth-matched jets is required to be equal to the soft term E_T^{soft} : that is, the E_T^{miss} computed from visible objects must be zero. This is performed by shifting the p_T and ϕ of each truth-matched jet within their experimental uncertainties while holding the pseudorapidity η constant¹⁰⁶. If the fit

¹⁰⁶Since changing the pseudorapidity will not change the jet’s orientation in the transverse plane, it can be held

does not converge, and this cannot be achieved, the event is rejected and not used. In the version of this procedure used in the preliminary 139 fb^{-1} result, all jets were rebalanced, but now that truth matching is used to identify pile-up jets, they are preserved unchanged and only hard scatter jets are used in the kinematic fit.

Once the events have been rebalanced, they are then smeared. The smearing process also involves adjusting the kinematics of an event’s jets, but whereas the rebalancing process does this using fit constraints, the smearing is done randomly. This is done by randomly sampling the jet energy response function, which is obtained from MC simulation and defined as the ratio of reconstructed jet energy to truth jet energy. For each jet, a random value $r > 0$ is obtained from the jet response distribution and used to scale the jet mass and energy, which causes p_T and ϕ to shift while η is held fixed. A single event can be smeared multiple times in order to artificially increase the statistics of the multijet estimate. The event weight of each smearing is reduced accordingly to represent the fact that multiple smeared events originated from the same seed.

The jet response distribution is non-Gaussian; it can be treated as having a Gaussian “core” and non-Gaussian “tail”. The entire distribution is sampled when performing the smearing, and therefore two systematic uncertainties are assigned based on varying the core and tail separately. When smearing an event, the smearing is performed both separately with the size of the core increased by 15%, and also with the tail increased by 50%. These variations are then taken as shape uncertainties on the multijet estimate.

8.4.2 Pileup vs Hard Scatter Jets

The rebalance and smear procedure generates QCD-like events with fake missing transverse momentum. Some of these events contain a leading hard scatter jet and a subleading pile-up jet, while others only contain hard-scatter jets. While both types of events will contribute to the overall multijet estimate, because these subsamples are generated separately¹⁰⁷, the *relative* size of each component of the total multijet estimate is not yet known. To determine this, a fraction fit is performed using un-prescaled single-jet triggered data. In the fraction fit, the shape of the [HS](#) and [HS+PU](#) templates are held fixed, but the normalization of each is allowed to vary. Because the pileup conditions varied considerably across run 2 between 2015/16, 2017, and 2018, this fraction fit is done separately for each of the three data-taking periods. A loose “multi-jet control region” selection is used for the fit, defined as follows:

constant.

¹⁰⁷And because they are derived from MC, rather than data, in the final version of the 139 fb^{-1} analysis.

- Leading, subleading jets must have $p_T(j_1) > 80 \text{ GeV}$, $p_T(j_2) > 50 \text{ GeV}$.
- Leading jet required to have $f_{\text{JVT}} < 0.2$.
- Events must not have any additional jets with $p_T > 25 \text{ GeV}$.
- Dijet system required to have $m_{jj} > 400 \text{ GeV}$, $\Delta\eta_{jj} > 2.5$, and $\Delta\phi_{jj} < 2$.
- Missing transverse momentum: $100 < E_T^{\text{miss}} < 200 \text{ GeV}$, with the soft term $E_T^{\text{soft}} < 20 \text{ GeV}$.

The fraction fit is performed using $\Delta\phi_{jj}$. Figure 8.17 shows post-fit plots of $\Delta\phi_{jj}$ and m_{jj} in this control region for the full run 2 dataset. After performing the fit, the **HS+PU** component was found to be responsible for about twice as much of the multijet background (68%, 74%, 65% of the total for 2015/16, 2017, and 2018 respectively) compared to the **HS-only** component (32%, 26%, 35% of the total). Therefore, pile-up conditions will clearly have a major impact on the structure and size of the multijet background.

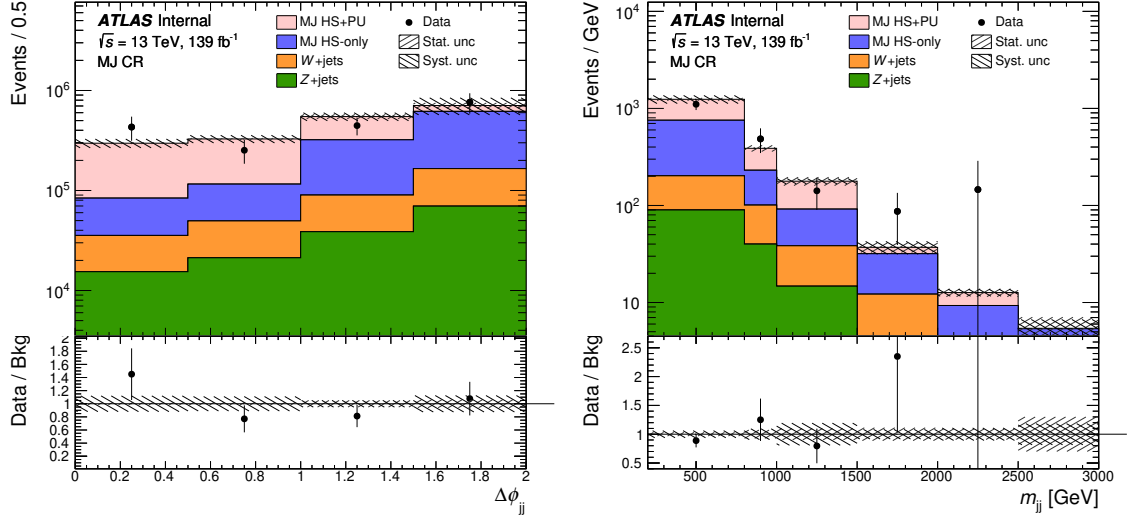


Figure 8.17: Post-fit plots of $\Delta\phi_{jj}$ and m_{jj} in the loose multijet control region. The mixed pileup and hard scatter (HS+PU) and hard scatter only (HS) components of the multijet estimate are shown separately here. A fraction fit to data in $\Delta\phi_{jj}$ is used in this region to normalize the relative fraction of each component. Note that while these plots show the full 139 fb^{-1} dataset, the fraction fit is performed separately for each of the three data-taking periods.

8.4.3 Closure and Normalization

The fit shown in Figure 8.17 generates a multijet estimate, which in principle could be used in the signal region. However, due to the complexity of the method, we perform a second multijet normalization in two validation regions that are much closer to the SR than the loose multijet control region defined above. These two validation regions are defined as follows:

- A “mid- E_T^{miss} , mid- m_{jj} ” region, with $160 < E_T^{\text{miss}} < 200$ GeV and $800 < m_{jj} < 1500$ GeV. This region was excluded from the signal region because the multijet background was too large; it therefore makes a good validation region for checking and normalizing the multijet estimate¹⁰⁸.
- A “low- m_{jj} ” region, where $200 < m_{jj} < 800$ GeV. Since the VBF Higgs to Invisible signal primarily consists of high m_{jj} events, and since even the $800 < m_{jj} < 1000$ GeV bin has a very low signal acceptance, it is safe to use this region to normalize the multijet estimate.

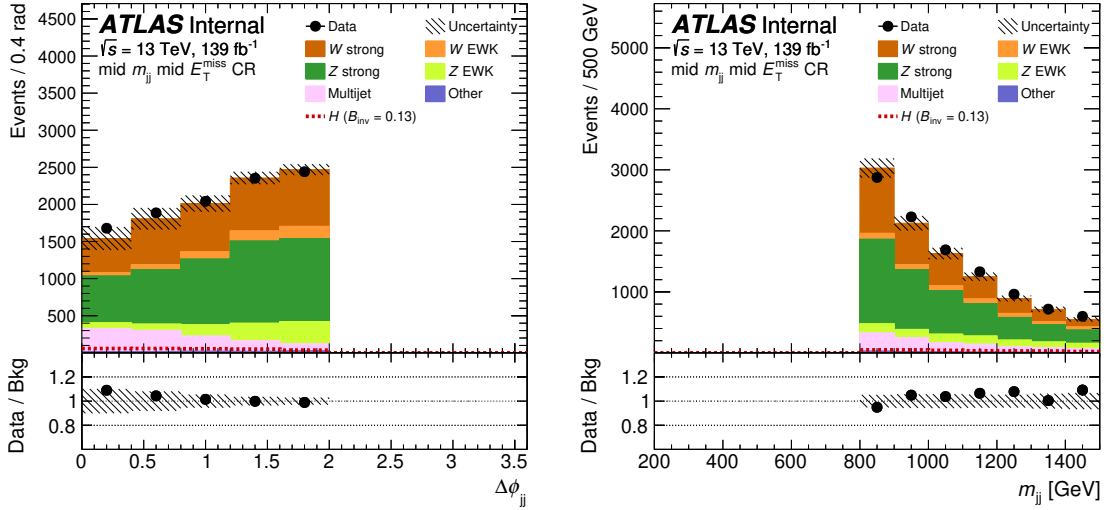


Figure 8.18: Plots of $\Delta\phi_{jj}$ and m_{jj} in one of the two multijet normalization regions. All cuts from the zero-lepton signal region are applied, except the missing transverse momentum requirement is changed to $160 < E_T^{\text{miss}} < 200$ GeV and the dijet invariant mass is changed to $800 < m_{jj} < 1500$ GeV. This region is used to validate the shape of the multijet estimate produced using rebalance and smear; additionally, a normalization factor is computed for the entire shape by subtracting the non-multijet backgrounds from data and then comparing to the multijet estimate.

Plots of data vs background (including both backgrounds modelled using MC, as well as the multijet estimate) in these regions are shown in Figures 8.18 and Figure 8.19. A normalization factor

¹⁰⁸For the preliminary analysis, a low- E_T^{miss} region ($100 < E_T^{\text{miss}} < 150$ GeV) was used instead before the E_T^{miss} cut was raised to 200 GeV (and then partially lowered back to 160 GeV).

is computed inclusively for each of the three data-taking periods by subtracting the non-multijet backgrounds from data, and then taking the ratio with the multijet prediction. This produces the multijet normalization factors seen in Table 8.3. The average of the two normalization factors is used to normalize the multijet estimate in the fit. Half the difference between the two normalization factors is taken as a systematic uncertainty: this is assigned as a “non-closure” systematic, which represents the extent to which the two closure tests did not agree with each other.

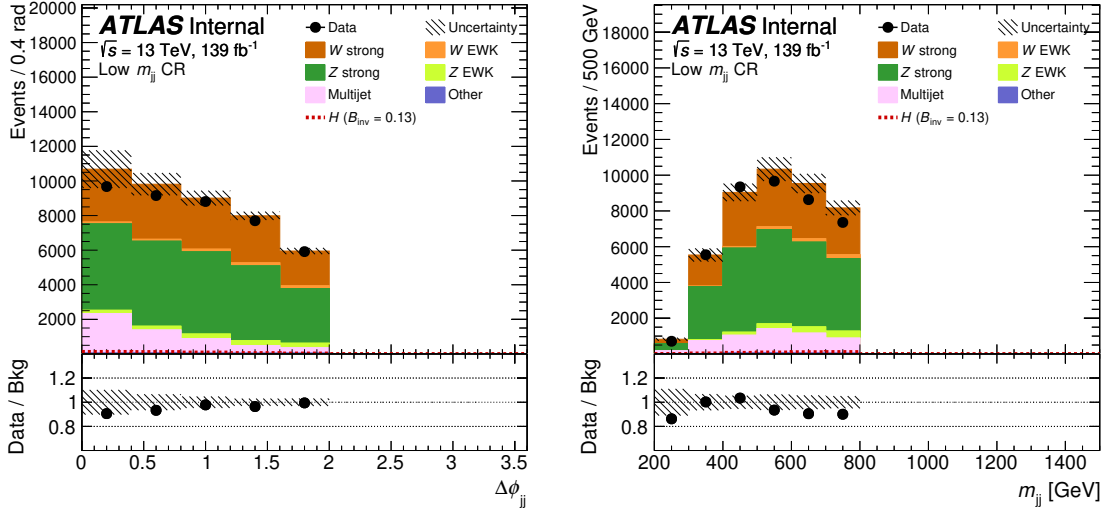


Figure 8.19: Plots of $\Delta\phi_{jj}$ and m_{jj} in one of the two multijet normalization regions. All cuts from the zero-lepton signal region are applied, except the m_{jj} requirement is changed to $200 < m_{jj} < 800$ GeV: the E_T^{miss} requirement is unchanged. See Figure 8.18 for details on the use of these two normalization/validation regions.

Year	Low m_{jj}	Mid E_T^{miss} , Mid m_{jj}	Average
2015/16	0.17	0.38	0.27 ± 0.11
2017	0.55	1.54	1.04 ± 0.49
2018	0.48	0.97	0.73 ± 0.25

Table 8.3: Normalization factors computed for the multijet estimate for the final 139 fb^{-1} analysis, from the regions shown in Figures 8.18 and Figure 8.19. The average of the two normalization factors is used, and half the difference between the two is taken as a systematic uncertainty.

A fourth systematic uncertainty is assigned in addition to the core and tail systematics and the non-closure normalization uncertainty. It was mentioned previously that the multijet estimate was performed using data for the preliminary 139 fb^{-1} result. While the final version of the analysis switched to using MC, in order to help validate that change, the two sets of results were compared. The difference between the two predictions was taken as another systematic uncertainty in order

to cover any changes in pile-up modeling or jet energy scale that may have arisen as a result of the switch to MC. The estimates are calculated separately for each data-taking period and for each year; to give a sense of the impact each uncertainty has on the multijet estimate, Table 8.4 shows the inclusive predictions for each data-taking period and the uncertainties on each. Overall, the multijet background appears to vary from approximately 2.8% of the total background in 2016 to 8.8% in 2017 and 7.0% in 2018, due to the increased pile-up conditions in those years.

Year	Yield	Stat (%)	Core (%)	Tail (%)	Data vs MC (%)	Non-Closure (%)
2015/16	112.8	3.7	3.9	2.5	2.4	40
2017	385.9	4.9	4.5	5.8	7.0	47
2018	414.6	3.2	8.7	5.9	-0.4	34

Table 8.4: Normalized inclusive predictions of the multijet background from 2015/16, 2017, and 2018, along with the statistical uncertainty on that estimate, as well as the core/tail, data vs MC, and non-closure systematic uncertainties.

8.5 Multijet Estimate using FJVT

The rebalance and smear procedure described above is quite complex, and as can be seen in Table 8.4, the systematic uncertainties on the prediction are not small. During the development of the preliminary 139 fb^{-1} analysis, these multijet systematics were briefly the leading source of systematic uncertainty before the E_T^{miss} cut was raised from 160 GeV to 200 GeV. Since the multijet background is much smaller than the V+Jets background, the systematics on the multijet should, in principle, not be more important than the systematics on the V+Jets when it comes to setting a limit on the invisible Higgs branching ratio. Therefore, a considerable amount of work was performed in order to improve the multijet estimate for the final 139 fb^{-1} result¹⁰⁹. Part of that work involved developing a second, independent procedure for performing the multijet estimate that does not use rebalance and smear. Instead, a transfer factor approach similar to that described in Sections 8.1 and 8.3 was adopted.

The transfer factor approach makes use of the fact that most of the multijet background appears to arise due to events with pile-up jets (the [HS+PU](#) component). To reduce the impact of pileup jets in the signal region, a requirement is imposed on FJVT, the forward jet vertex tagger score: both the leading and subleading jet must have FJVT less than 0.5 (or FJVT less than 0.2 for events with $160 < E_T^{\text{miss}} < 200\text{ GeV}$). Therefore, in order to produce a region enriched in pileup events–

¹⁰⁹Part of that result involved modifying the rebalance and smear procedure to use MC rather than data, as already described previously.

and therefore, multijet contamination– this FJVT requirement can be inverted to instead require a jet with FJVT *larger* than 0.5 (or 0.2 for events with $160 < E_T^{\text{miss}} < 200 \text{ GeV}$). To ensure this multijet “FJVT control region” is pure in multijet events, only the leading jet FJVT requirement is inverted. As shown in Figure 8.20, there are huge excesses seen in this region not explained by Monte Carlo, indicating a large multijet presence. To increase the statistics of this region, no N_{jets} veto is applied to any of the bins.

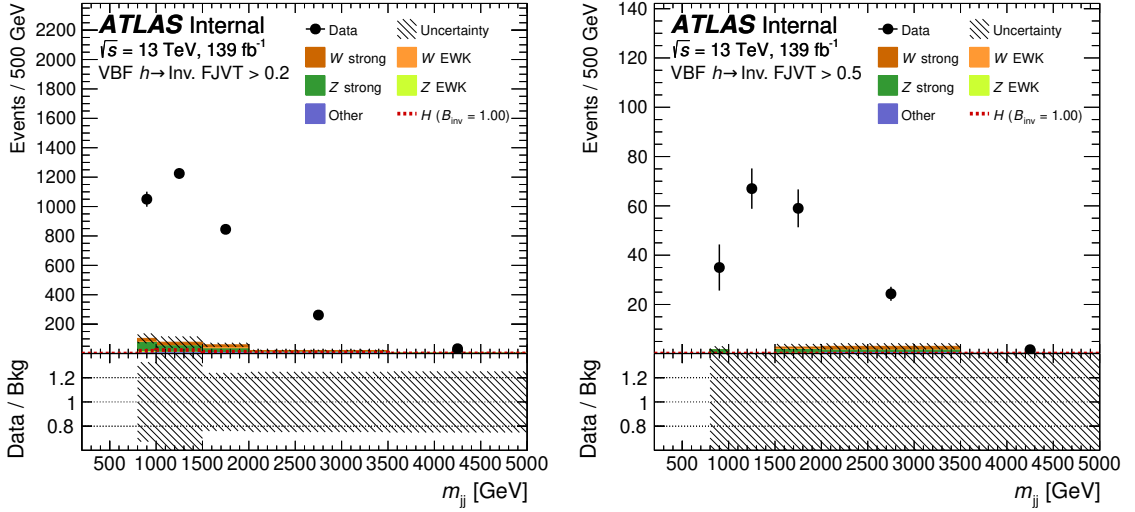


Figure 8.20: Plots showing the multijet control region, in which the FJVT requirement on the leading jet is inverted. The left plot shows events with $160 < E_T^{\text{miss}} < 200 \text{ GeV}$, where the leading jet must have FJVT greater than 0.2, while the plot on the right shows events with $E_T^{\text{miss}} > 200 \text{ GeV}$, where the leading jet must have FJVT greater than 0.5. As can be seen from these plots, these regions are very pure in multijet events– the MC simulation of other backgrounds represent only around 5-10% of the data in both cases.

As previously explained, the transfer factor approach requires four regions. Since the lower the missing transverse momentum requirement, the easier it is for a multijet event to have fake E_T^{miss} , so the transfer factor itself is derived from a low- E_T^{miss} version of the SR, in which events must have $100 < E_T^{\text{miss}} < 160 \text{ GeV}$ but is otherwise identical to the 0 lepton signal region. The size of the multijet contamination here is computed by subtracting the non-multijet MC backgrounds from the data. The transfer factor is then computed by taking the ratio of events which pass the leading jet FJVT requirement and have exactly two jets to those that fail it; the denominator here, like the FJVT control region, includes events that may have more than two jets. This ratio is labelled R_{MJ} in analogy to the fake lepton transfer factors explained above. Two sets of transfer factors are computed: one using a FJVT threshold of 0.2, for use in the $160 < E_T^{\text{miss}} < 200 \text{ GeV}$ bins,

and another using a FJVT threshold of 0.5 for use in the $E_T^{\text{miss}} > 200 \text{ GeV}$ bins. Studies were done to demonstrate that varying the E_T^{miss} cut has minimal impact on the transfer factor, and that therefore it is possible to use the low- E_T^{miss} region to constrain the multijet background in the high- E_T^{miss} signal region.

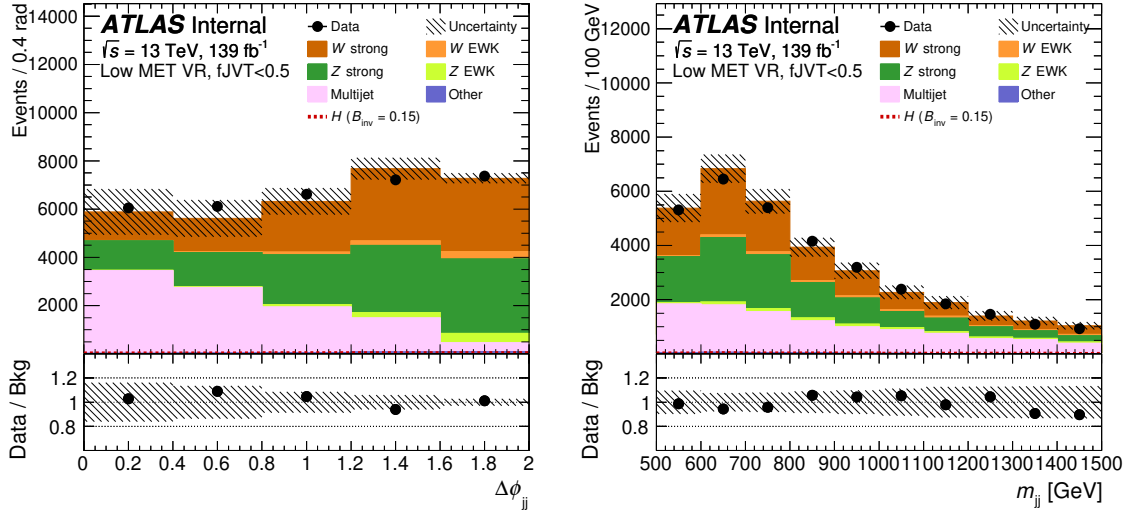


Figure 8.21: Pre-fit plots showing the multijet low- E_T^{miss} validation region ($110 < E_T^{\text{miss}} < 150 \text{ GeV}$) used to check the transfer-factor-based multijet estimate. The pink multijet background appears to do a reasonable job of covering the differences between data and non-multijet backgrounds (simulated using MC) in this region, which due to the lower missing transverse momentum requirement has more multijet events than the signal region.

The transfer factor R_{MJ} is computed for every bin and used to correct the multijet background measured in the FJVT control region, which is again defined as the difference between data and the non-multijet events. This produces the FJVT multijet estimate. As a test of this procedure, the FJVT multijet estimate can be applied back to the low- E_T^{miss} region to see if it explains the excess between data and MC. This is done using a reduced $110 < E_T^{\text{miss}} < 150 \text{ GeV}$ selection, to avoid performing the validation in exactly the same region that was used to compute the transfer factor. Figure 8.21 shows that the FJVT multijet estimate does appear to cover this difference in this validation region. As with the fake lepton estimate, the statistical uncertainties on the transfer factors are assigned as a systematic uncertainty for each bin. Additionally, as studies showed that the transfer factors have some dependence on either the jet p_T or the invariant mass m_{jj} ¹¹⁰, another systematic uncertainty of approximately 20% was assigned to cover these differences. A summary

¹¹⁰Similar to the studies that were done on the fake lepton estimate, described in Section 8.3 above.

E_T^{miss} (GeV)	Year	Multijet	Syst	Stat
200+	2015/16	95	19	15
	2017	207	42	25
	2018	187	38	22
160-200	2015/16	117	24	7
	2017	154	32	8
	2018	132	27	8

Table 8.5: Table summarizing the multijet prediction from the FJVT control region for each data-taking period, along with the systematic and statistical error. Yields are given separately for events with $E_T^{\text{miss}} > 200$ GeV and $160 < E_T^{\text{miss}} < 200$ GeV.

of the multijet prediction from the FJVT control region in each bin is shown below in Table 8.5

8.5.1 Comparison with Rebalance and Smear

At this point, it is necessary to ask how FJVT method compares to the rebalance and smear method described in Section 8.4. As can hopefully be seen by the respective lengths of the sections, the FJVT procedure is much simpler, and conceptually more like the other methods employed in the analysis to estimate other backgrounds. Numerically, the pre-fit multijet prediction for the inclusive signal region from rebalance and smear is 912 ± 383 events, while the FJVT method gives 892 ± 194 events. These predictions agree well, although as can be seen here, the FJVT prediction has considerably smaller systematic uncertainties. Figure 8.22 shows a bin-by-bin comparison between the two methods for the $N_{\text{jets}} = 2, E_T^{\text{miss}} > 200$ GeV bins: as can be seen, in most bins the two predictions agree reasonably well within only their statistical uncertainties. Since the methods are very different, and since the systematic uncertainties on rebalance and smear are not small, this is a good sign.

A combination of the two methods is employed in the analysis. In the low- E_T^{miss} bins, where rebalance and smear gave very large uncertainties, the predictions from the FJVT transfer factors are used directly. The transfer factors are also used directly in the $\Delta\phi_{jj} < 1$ bins, where the statistical uncertainties are reasonable. One exception is the $800 < m_{jj} < 1000$ GeV bin: as this is the one bin with a large discrepancy between the two methods, an extra systematic uncertainty is assigned to cover that difference¹¹¹. In the $1 < \Delta\phi_{jj} < 2$ bins, the statistics in the FJVT control region are poor, as can be seen in Figure 8.22. To augment them, the transfer factor is computed inclusively instead of bin-by-bin and used to scale the multijet estimate derived from rebalance and smear. And finally, because events with $N_{\text{jets}} > 2$ are used to compute the transfer factors for $N_{\text{jets}} = 2$ bins, it is not

¹¹¹The lower m_{jj} bins have an almost negligible impact on the analysis's sensitivity, so assigning a larger systematic uncertainty here has a minimal impact on the expected limit on the invisible Higgs branching ratio.

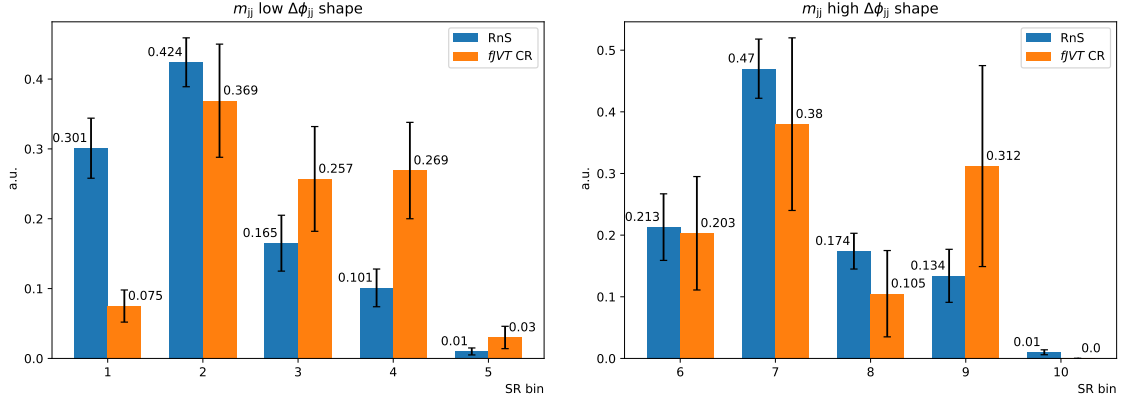


Figure 8.22: Comparisons between the predictions on the multijet background shape between those from rebalance and smear (in blue) and those from the FJVT control region (in orange). The plot on the left shows predictions in the five m_{jj} bins for $0 < \Delta\phi_{jj} < 1$, while the plot on the right shows predictions in the $1 < \Delta\phi_{jj} < 2$ bins. Only statistical uncertainties are included, and the shapes are normalized so that the integral of each distribution is unity. With the exception of the first low $\Delta\phi_{jj}$ bin, the two predictions tend to agree reasonably well within those statistical uncertainties.

safe to use the same events to compute transfer factors for the $N_{\text{jets}} > 2$ bins. Therefore, only the rebalance and smear prediction is used in this region. The combination of these two methods has significantly reduced the impact of multijet-related systematic uncertainties for the final 139fb^{-1} result.

CHAPTER 9

Results and Interpretation

Using the background estimation techniques developed in Chapters 7 and 8, we can at last unblind the signal region and check to see whether any evidence of invisible Higgs decays was observed. Unblinding an analysis like this in ATLAS only occurs once all of the components– the signal region definition described in Chapter 6, the background estimation strategy and inputs, and the statistical framework used to apply those inputs– have been finalized and validated. The signal region is kept blinded until an internal review process performed by members of the ATLAS collaboration not directly involved in the analysis give approval to unblind. Only then do we look at data in the signal region and set a limit on the invisible Higgs branching ratio, $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$.

This chapter describes that process: the statistical framework, known as the likelihood fit, used to apply the background estimation strategy, and the observed and expected limits on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$. The details of how the likelihood fit works are described in Section 9.1. Before unblinding, the fit was tested in a high- $\Delta\phi_{\text{jj}}$ version of the signal region, as described in Section 9.2. Then Section 9.3 shows results from the unblinded signal region before the fit, while Section 9.4 shows the post-fit results, the limit that we set on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$, and the dominant sources of uncertainty on that limit. Next, Section 9.5 presents a few additional interpretations of the limit, most notably including the Higgs portal dark matter model(s) and a comparison to results from dark matter direct detection experiments. And finally, Section 9.6 shows the latest corresponding results from the CMS collaboration and makes a brief comparison between the two.

This chapter (mostly) focuses on results from the final version of the 139fb^{-1} analysis. Results from the preliminary version can be found in Appendix C for comparison purposes, to see how the improvements in V+Jets and multijet background estimation caused the limit to improve over time. Note that the analysis was “re-blinded” between the two iterations of the analysis, meaning that

we did not look at signal region data while implementing changes, again to minimize the possibility of bias. The set of changes between the two iterations were agreed upon ahead of time with our internal ATLAS reviewers, and a second unblinding review was performed before moving ahead with the second version of the analysis.

9.1 Likelihood Fit Details

An analysis like this one is, in some ways, just a counting experiment. The background estimate tells us how many events we expect in the signal region; we then count to see how many events we actually have. A statistical test is needed to understand whether the observed events are consistent with just the background estimate, or the background estimate plus a signal hypothesis, which in this case is the invisible Higgs boson decay. In this analysis, these questions are answered by means of a maximum likelihood fit to the data. Since counting statistics can be modelled using the Poisson probability distribution, we can use the Poisson probability distribution function to construct what's known as the *likelihood* function to determine the likelihood of observing a certain number of events in every bin of every region independently. This function will depend on several parameters, and these parameters are varied during the fit in order to maximize them [193].

Once the likelihood function has been defined, and best-fit values obtained from a fit to the data, the significance of the result needs to be determined. If a large excess of data events over the background prediction appears after the fit, and appears consistent with the invisible Higgs signal model, than this could indicate signs of new physics beyond the Standard Model. On the other hand, if the data and background appear to agree quite well, limits on the existence of the invisible Higgs signal process need to be set instead. By setting an exclusion limit on $\mathcal{B}_{\text{H to inv.}}$, we are saying that the results of this analysis have ruled out an invisible Higgs branching ratio greater than the observed limit, to within some confidence level. In this analysis, these limits are set using frequentist statistics, by means of a technique known as CL_s [194] [195].

These methods are summarized in this section. For this analysis, the fitting and limit-setting were implemented using the software packages HistFactory [196] and HistFitter [197], which use the RooFit [198] and RooStats [199] extensions to ROOT. The fit results in the next few sections and in Appendix C were produced using a framework based on these libraries.

9.1.1 Basic Methods

The Poisson probability of observing N events if M are predicted can be written as follows [200]:

$$\mathcal{P}(N|M) = \frac{e^{-M} M^N}{N!} \quad (9.1)$$

If the analysis only had one bin and one region, then we could write down the probability of observing N events in the signal region using the expression above. However, there are a number of control regions that are used in order to determine the various backgrounds, and these also need to be taken into account. Consider an example in which we have one signal region and one control region, and we observe N^{SR} events and N^{CR} events respectively in each. If M^{SR} and M^{CR} represent the predicted number of events in each region, the total probability could be written as $\mathcal{P}(N^{\text{SR}}|M^{\text{SR}})\mathcal{P}(N^{\text{CR}}|M^{\text{CR}})$. Additional terms could be added for additional control regions as well.

This product of Poisson probabilities for each region of the analysis is known as the *likelihood* function, $\mathcal{L}$ [200] [12]. The likelihood can be written as a function of the observed and expected numbers of background events in each region. Of course, the expected number of events in the signal region could include both background events *or* signal events— an excess of data over the background might be consistent with the signal model. To represent this, the predicted number of signal region events can be written in terms of a parameter μ , known as the signal strength, as follows:

$$M^{\text{SR}} = B^{\text{SR}} + \mu S^{\text{SR}} \quad (9.2)$$

In the VBF+MET analysis, the signal prediction S^{SR} comes from Higgs to invisible Monte Carlo. The signal strength μ can then be thought of as representing the invisible Higgs branching ratio, $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$. The likelihood $\mathcal{L}(\mu)$ then is the probability that the observed data is consistent with a signal strength of μ ; for example, $\mathcal{L}(0.37)$ would be the probability of observing an invisible Higgs branching ratio of $\mathcal{B}_{\text{H} \rightarrow \text{inv.}} = 0.37$ ¹¹². Of course, the likelihood will depend on other parameters besides the signal strength: the background estimate B^{SR} depends on various transfer factors derived from the data/MC agreement in various control regions, for instance. When fitting the background and signal predictions to the data in all the regions, the likelihood function will be used: the fit will try to maximize the likelihood by shifting or “pulling” all these parameters. So while the transfer factors will be initialized to their pre-fit values, they can vary during the fit as needed. The signal strength μ can also be allowed to vary, or it can be held fixed and the other parameters maximized for a given value of μ .

In addition to the transfer factors, and other parameters involving the background estimate, some of the free parameters represent the various sources of systematic uncertainty, known as “nuisance

¹¹²0.37 was the observed limit in the 36.1 fb^{-1} analysis, as mentioned in Chapter 6.

parameters”. These nuisance parameters are generally labelled $\vec{\theta}$. Each nuisance parameter is modelled via a Gaussian distribution, and is defined in such a way that $\theta = 0$ represents the “central value” of the systematic, and normalized so that the uncertainty $\sigma_\theta = 1$. When fitting the likelihood function to the data, each nuisance parameter is able to vary away from the central value within the systematic uncertainty, and the probability of doing so is determined by the Gaussian probability distribution [200]:

$$\mathcal{G}(\theta) = \frac{e^{\frac{-(\theta)^2}{2\sigma_\theta^2}}}{\sqrt{2\pi\sigma_\theta^2}} \quad (9.3)$$

If a systematic uncertainty does shift up or down, it will adjust the predicted number of background or signal events. Therefore, the predicted numbers of events are modified by an exponential response function $(1 + \epsilon)^\theta$ for each nuisance parameter, where ϵ is the fractional uncertainty of the parameter θ . As θ shifts from $0 \rightarrow \pm 1$, this will cause the predicted yields to increase or decrease by an amount proportional to the relative size of the uncertainty in question. Nuisance parameters that represent systematics on the signal prediction will modify S^{SR} , while others will modify the background predictions as appropriate. To simplify the equations shown in this and the next section, this dependence on these exponential response functions is omitted when writing B^{SR} , S^{SR} , etc.

Finally, consider that the signal region (and all the control regions) are actually divided into 16 independent bins. Therefore, the probabilities of observing events in one bin will be independent from each other, and the likelihood function should be written as a product of probabilities across all bins. That is, $\mathcal{P}(N^{\text{SR}}|M^{\text{SR}})$ should really be written as $\prod_i \mathcal{P}(N_i^{\text{SR}}|M_i^{\text{SR}})$, where $1 \leq i \leq 16$ is a product over all sixteen bins.

9.1.2 Likelihood Function

Having introduced the concept of the likelihood function $\mathcal{L}$, we now need to define it. The likelihood function for this analysis is a complicated expression, involving a product over sixteen bins in several different analysis regions, plus the various nuisance parameters. It can be divided into several terms, as shown in Equation 9.4: we will sequentially define each one.

$$\mathcal{L}(\mu, \vec{\beta}_V, \vec{n}, \vec{\theta}) = \mathcal{L}^{\text{SR}}(\mu, \vec{\beta}_V, \vec{n}) \mathcal{L}^{\text{MJ}}(\vec{n}) \mathcal{L}^{\text{V+Jets}}(\vec{\beta}_V, \vec{n}) \mathcal{L}^{\text{fakes}}(\vec{\beta}_V, \vec{n}) \mathcal{L}^{\text{NP}}(\vec{\theta}) \quad (9.4)$$

First, let’s take the signal region likelihood, $\mathcal{L}^{\text{SR}}$, shown below in Equation 9.5. This is a product over all sixteen bins, and the predicted number of events in each bin is the signal region background prediction for that bin, plus the signal S_i^{SR} in that bin scaled by the strength μ .

$$\mathcal{L}^{\text{SR}}(\mu, \vec{\beta}_V) = \prod_i \mathcal{P}\left(N_i^{\text{SR}} | \beta_i \mathcal{R}^{Z/W} B_{Z,i}^{\text{SR}} + \beta_i B_{W,i}^{\text{SR}} + B_{\text{MJ},i}^{\text{SR}} + B_{\text{other},i}^{\text{SR}} + \mu S_i^{\text{SR}}\right) \quad (9.5)$$

As discussed in Chapter 8, there are several backgrounds that enter the signal region and must be estimated: a $W \rightarrow l\nu$ lost lepton estimate (labelled $B_{W,i}^{\text{SR}}$), a $Z \rightarrow \nu\nu$ background estimate (labelled $B_{Z,i}^{\text{SR}}$), a multijet background estimate (labelled $B_{\text{MJ},i}^{\text{SR}}$), and a handful of other backgrounds (labelled $B_{\text{other},i}^{\text{SR}}$, primarily comprised of top-quark and multi-boson processes). These terms collectively make up the expression in Equation 9.5. The predicted multijet background comes either from rebalance and smear or the FJVT control region; see below for the definition of $\mathcal{L}^{\text{MJ}}$. The two V+Jets background estimate are both scaled by the V+jets transfer factor β , which is initially defined as the pre-fit ratio between data and background in the one-lepton control regions, but can vary during the fit. During the final version of the analysis, the reweighting term $\mathcal{R}^{Z/W}$, which is the ratio $\mathcal{R}_{\text{TH}}/\mathcal{R}_{\text{MC}}$ that was introduced in Section 8.2, is used to reweight the one-lepton transfer factor in order to predict the $Z \rightarrow \nu\nu$ background¹¹³.

The transfer factor β comes from the V+jets control regions, and so it is also present in those likelihood terms, shown below in Equation 9.6. In these regions, the predicted number of events arises due to three terms: the Z+Jets and W+jets backgrounds in each bin (labelled $B_{Z,i}$ or $B_{W,i}$ respectively), which are modified by the transfer factors; the non-Z+Jets and non-W+jets backgrounds (labelled $B_{\text{non-W},i}$ and $B_{\text{non-Z},i}$); and, in the one-lepton regions, the numbers of fake electron and fake muon events, $n_{\text{fake-}e,i}$ and $n_{\text{fake-}\mu,i}$. Much like the signal region, the $Z \rightarrow ll$ background prediction is scaled by the reweighting term $\mathcal{R}^{Z/W}$ so that a single V+jets transfer factor can be used. When the fit varies the β transfer factors, it will impact agreement in both the signal and control regions.

$$\begin{aligned} \mathcal{L}^{\text{V+jets}}(\vec{\beta}_V, \vec{n}) = & \prod_i \mathcal{P}\left(N_i^{Z(l)\text{CR}} | \beta_i \mathcal{R}^{Z/W} B_{Z,i}^{Z(l)\text{CR}} + B_{\text{non-Z},i}^{Z(l)\text{CR}}\right) \\ & \prod_i \mathcal{P}\left(N_i^{W(e\nu)\text{CR}} | \beta_i B_{W,i}^{W(e\nu)\text{CR}} + B_{\text{non-W},i}^{W(e\nu)\text{CR}} + R_M n_{\text{fake-}e,i}\right) \\ & \prod_i \mathcal{P}\left(N_i^{W(\mu\nu)\text{CR}} | \beta_i B_{W,i}^{W(\mu\nu)\text{CR}} + B_{\text{non-W},i}^{W(\mu\nu)\text{CR}} + R_{S,i} n_{\text{fake-}\mu,i}\right) \end{aligned} \quad (9.6)$$

The fake lepton predictions come from the low- S_{MET} and low- m_{T} fake electron and fake muon control regions, which were introduced in Section 8.3¹¹⁴. These regions also enter the likelihood

¹¹³In the preliminary version of the analysis, this technique is not used. Instead, a second β_Z transfer factor is computed from the $Z \rightarrow ll$ control region, and allowed to vary independently of β_W . See Appendix C for the results of this version of the fit.

¹¹⁴The preliminary version of the analysis only performed a fake electron estimate, and so no fake muon term was present.

function in the $\mathcal{L}^{\text{fakes}}$ term, which looks very similar to $\mathcal{L}^{\text{V+jets}}$ above. The predicted numbers of W+Jets events in this region ($B^{\text{fake-}e \text{ CR}}$) is also normalized using the same transfer factor β . The fake muon and fake electron estimates, $n_{\text{fake-}e}$ and $n_{\text{fake-}\mu}$, are simply the remaining events after accounting for the other non-W backgrounds. These parameters, which are free to vary during the fit, enter $\mathcal{L}^{\text{V+jets}}$ above, where they are scaled by the transfer factors R_M and $R_{S,i}$, which were determined from the anti-ID regions described in Section 8.3.

$$\begin{aligned} \mathcal{L}^{\text{fakes}}(\vec{\beta}_V, \vec{n}) = & \prod_i \mathcal{P}\left(N_i^{\text{fake-}e \text{ CR}} | \beta_i B_{W,i}^{\text{fake-}e \text{ CR}} + B_{\text{non-W},i}^{\text{fake-}e \text{ CR}} + n_{\text{fake-}e,i}\right) \\ & \prod_i \mathcal{P}\left(N_i^{\text{fake-}\mu \text{ CR}} | \beta_i B_{W,i}^{\text{fake-}\mu \text{ CR}} + B_{\text{non-W},i}^{\text{fake-}\mu \text{ CR}} + n_{\text{fake-}\mu,i}\right) \end{aligned} \quad (9.7)$$

Next, let's consider the multijet background. As discussed in Section 8.4 and Section 8.5, the multijet estimate that enters the signal region comes from two different methods: in the low- $\Delta\phi_{jj}$ bins (1-5) and the low- E_T^{miss} bins (14-16), it comes from the inverted FJVT control regions. For the $N_{\text{jets}} > 2$ bins (11-13), it comes from rebalance and smear. And for the high- $\Delta\phi_{jj}$ bins (6-10), an inclusive prediction from the FJVT control region is scaled by the rebalance and smear estimate in that bin. For bins 1-5, 14-16, and (inclusively) 6-10, the FJVT control region also needs to enter the likelihood function, as shown below in Equation 9.8:

$$\mathcal{L}^{\text{MJ}}(\vec{n}) = \prod_i \mathcal{P}\left(N_i^{\text{FJVT CR}} | B_{\text{non-MJ},i}^{\text{FJVT CR}} + n_{\text{MJ},i}\right) \quad (9.8)$$

The multijet estimate in the signal region is then accordingly defined piecewise as follows, depending on the bin i . The low- E_T^{miss} transfer factor R_{MJ} is used to rescale the number of multijet events $n_{\text{MJ},i}$ in the bins that use the FJVT prediction:

$$B_{\text{MJ},i}^{\text{SR}} = \begin{cases} n_{\text{MJ},i} R_{\text{MJ},i} & i \in [1, 5], [14, 16] \\ n_{\text{MJ}, \text{high-}\Delta\phi_{jj}} R_{\text{MJ}, \text{high-}\Delta\phi_{jj}} f_{\text{R+S},i} & i \in [6, 10] \\ B_{\text{MJ}, \text{R+S},i}^{\text{SR}} & i \in [11, 13] \end{cases} \quad (9.9)$$

Finally, let's consider the nuisance parameter term, $\mathcal{L}^{\text{NP}}$. As explained above, each nuisance parameter is represented using a Gaussian probability distribution centered at a mean of 0, as shown below in Equation 9.10:

$$\mathcal{L}^{\text{NP}}(\vec{\theta}) = \prod_j \mathcal{G}(0 | \theta_j) \quad (9.10)$$

The nuisance parameters representing experimental systematics are treated as correlated across all bins and regions in the fit. The shapes of a small number of systematics were found to be statistically limited, most notably the jet energy response uncertainties. To prevent this from causing problems during the fit, a parabolic smoothing procedure was used for these nuisance parameters to combine neighbouring low-stats bins together. The uncertainty for that specific parameter is then taken from the combined bin. On the other hand, a more sophisticated correlation procedure was used for the theory systematics. While the PDF uncertainties on the V+Jets background were taken as correlated across all bins, the other V+Jets systematics were divided into one uncorrelated component for each bin, plus an additional component that was correlated across all bins. This approach was found to set a slightly less significant limit (using the methods explained below) compared to fully correlating the uncertainties, so it was adopted to be conservative.

9.1.3 Setting Limits

The likelihood can be used to calculate the probability that a given value of μ is consistent with the observed data in all the regions. In a search like this one, however, we are not just interested in the likelihood that a given value of μ happens to explain the data: we want to distinguish between two hypotheses, a “background-only” hypothesis (where $\mu = 0$, and the background prediction alone fully explains the data), and a “signal+background” hypothesis (where $\mu > 0$). Ultimately, we will want to set a limit on the existence of the signal+background hypothesis, that excludes a range of signal hypotheses within some confidence level. To do this, a frequentist approach is used, where we define a test statistic and then compute p -values to assess the statistic’s significance. In frequentist statistics, a p -value is conventionally defined as the probability that the observed data is consistent with the null hypothesis (which, for our purposes, is the background-only hypothesis).

The test statistic used to answer this question is known as the profile likelihood ratio, $q(\mu)$, defined below in Equation 9.11 [193]. This compares the likelihood that a given signal strength μ is consistent with the data to the likelihood of the *best-fit* value $\hat{\mu}$ when the signal strength is allowed to freely float along with the other parameters.

$$q(\mu) = -2 \ln \frac{\mathcal{L}(\mu, \hat{\theta}_\mu)}{\mathcal{L}(\hat{\mu}, \hat{\theta})} \quad (9.11)$$

To calculate p -values from this statistic, the probability distribution function of the statistic $q(\mu)$ is needed. Unfortunately, this is difficult to determine. Monte Carlo methods¹¹⁵ could be used

¹¹⁵Meaning general Monte Carlo methods; i.e. not the specific sense in which particle physicists employ “Monte

to generate datasets with different best-fit parameters, but this rapidly becomes computationally expensive. Fortunately, provided the sample dataset is large, as is the case here, several simplifying approximations can be made. In the asymptotic limit, as the dataset size increases, the profile likelihood ratio can be approximated and its probability distribution found to be a special type of χ^2 distribution known as a non-central χ^2 distribution. The probability distribution functions for this non-central χ^2 distribution are known, and so can be used to calculate p -values [193].

Instead of using p -values directly, we instead make use of a metric referred to as CL_s , defined as the ratio of the p -value for the signal+background hypothesis to 1 - the p -value of the background-only hypothesis [193] [195]:

$$\text{CL}_s = \frac{p_{s+b}}{1 - p_b} \quad (9.12)$$

The 95% confidence level limit on the existence of the signal hypothesis is defined as the signal strength μ which yields $\text{CL}_s = 0.05$; similarly, limits at other confidence levels would be set by adjusting the target CL_s accordingly. This signal strength is then the observed limit on the invisible Higgs branching ratio $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$.

In a search for new physics like this one, we are interested in setting both *observed* and *expected* limits. The observed limit is the limit that was actually found when fitting to real data in the signal region. The expected limit, on the other hand, is the best-possible limit that could be set if the signal region was perfectly predicted by the background. An expected limit is set using what's commonly referred to as an ‘‘Asimov’’ fit¹¹⁶, in which real data in the signal region is not used and N^{SR} is set equal to the background prediction B^{SR} . The maximum likelihood fit is performed as described above, and the value of μ for which $\text{CL}_s = 0.05$ is taken to be the expected limit. This can be thought of as quantifying how sensitive the analysis could be if the data and background agree perfectly. If the observed limit is larger than the expected limit, it indicates the presence of an excess in the data that is not fully explained by the background prediction.

9.2 Validation Fits

Once the signal region is unblinded, the likelihood fit model—like the rest of the analysis—should not be changed. It is therefore important to perform one or more validation fits to ensure that things

Carlo’’ to refer to the generation of simulated collision data described in Chapter 7.

¹¹⁶According to Glen Cowan et al., the Asimov dataset or Asimov fit is so named as a reference to the Isaac Asimov short story Franchise, in which ‘‘elections are held by selecting the single most representative voter to replace the entire electorate’’ [193].

appear stable before proceeding with unblinding. This is especially true for the final version of the 139 fb^{-1} analysis, now that using the W to constrain the Z has introduced additional complexity compared to the preliminary version or even the older 36.1 fb^{-1} version. This section presents two validation fits used to test the fit implementation prior to unblinding: one that just uses the control regions with visible leptons, and one which uses a high- $\Delta\phi_{jj}$ version of the 0-lepton signal region.

9.2.1 Control Region Only

One way to validate the fit is to perform a “control region only” fit. In the control region only fit, both data and MC simulation from the signal region are excluded and the fit only attempts to normalize the control regions. By definition, the control regions should have no signal contamination, so if everything is working properly, we would expect to see very good post-fit agreement between data and MC in these regions. Any excess (or deficit) of data events would indicate poor background modelling of the visible V +jets processes that make up these regions, not the presence of invisible Higgs events. Figure 9.1 shows the results of the control-region-only fit: the post-fit agreement between data and background for each bin in each region, the uncertainty on each bin, and how each bin shifted during the fit.

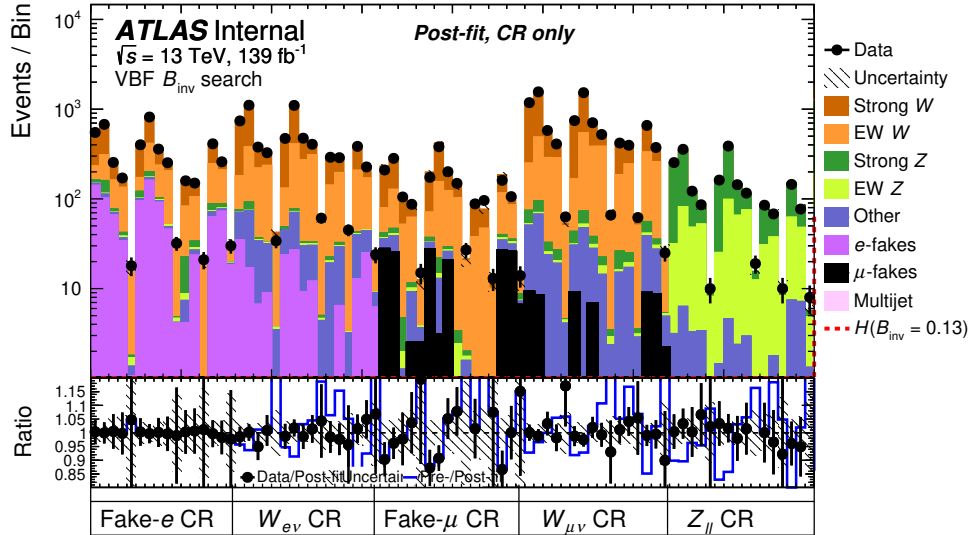


Figure 9.1: Post-fit plot from the “control region only” fit, showing agreement between data and Monte Carlo in all sixteen bins in the W and Z control regions, with both statistical and systematic errors shown. The low- S_{MET} fake electron and low- m_T fake muon estimation regions are also plotted. The blue line in the bottom panel shows the shifts in each bin before and after the fit, while the black dot indicates the post-fit agreement between data and background in each bin.

From this plot, we can see that the control region fit does indeed lead to good agreement in all of the bins in the W and Z control regions within the uncertainties. If we compare these results to the pre-fit distributions for these regions shown in Figures 8.1, 8.2, and 8.4, we can see that the agreement is much better following the fit.

9.2.2 High $\Delta\phi_{jj}$ Validation Region

In the one- and two- lepton control regions, the dominant backgrounds are visible V+jets processes ($Z \rightarrow ll$ and $W \rightarrow l\nu$), not the invisible V+jets backgrounds ($Z \rightarrow \nu\nu$, $W \rightarrow l\nu$ lost lepton) that contaminate the signal region. The control region only fit, by definition, cannot validate that the visible V+jets processes can be used to model the invisible processes: it can only verify that the visible V+Jets processes themselves are modelled consistently across the control regions. To verify that the transfer factor extrapolation really does work, especially now that the W is being used to help constrain the Z , another validation fit was developed: the high- $\Delta\phi_{jj}$ validation region.

The signal region, as described in Section 6.3, requires that all events have two jets that are not back-to-back; i.e. with an azimuthal separation $\Delta\phi_{jj} < 2$. This requirement is imposed because jets from vector boson fusion should be in the same plane, while jets produced in association with Z or W production do not need to be in the same plane. Therefore, this requirement is very useful in distinguishing signal from background. But it also means that if we were to *invert* the cut, and require $\Delta\phi_{jj} > 2$, we would exclude almost all of the invisible Higgs signal and primarily select background V+jets events. That suggests that the high- $\Delta\phi_{jj}$ area of phase space could safely be used for validation purposes, as it should have minimal signal contamination. A validation fit in this region could attempt to measure transfer factors from high- $\Delta\phi_{jj}$ versions of the one- and two-lepton control regions, and then use them to make a background estimate in the high- $\Delta\phi_{jj}$ version of the signal region. This would check to see if the transfer factor approach works in an area of phase space similar, but not identical to, the signal region, and provides a good complement to the control region only fit.

Studies were done to check whether or not this high- $\Delta\phi_{jj}$ region would work for this purpose. Figure 9.2 shows plots of several distributions in the “blinded” 0-lepton high- $\Delta\phi_{jj}$ “signal” region¹¹⁷. As these distributions show, the signal contamination is overall quite low across the full range of m_{jj} . On average, $S/(S+B)$ was found to be 13% across all three data-taking periods, which is low enough to be safe to unblind this region and use it for validation purposes. In addition, as shown in the plot

¹¹⁷As this is *not* really the signal region, it isn’t necessary to keep this region blinded while performing these studies, provided the signal contamination is low enough— which it is, as shown in the plots in Figure 9.2.

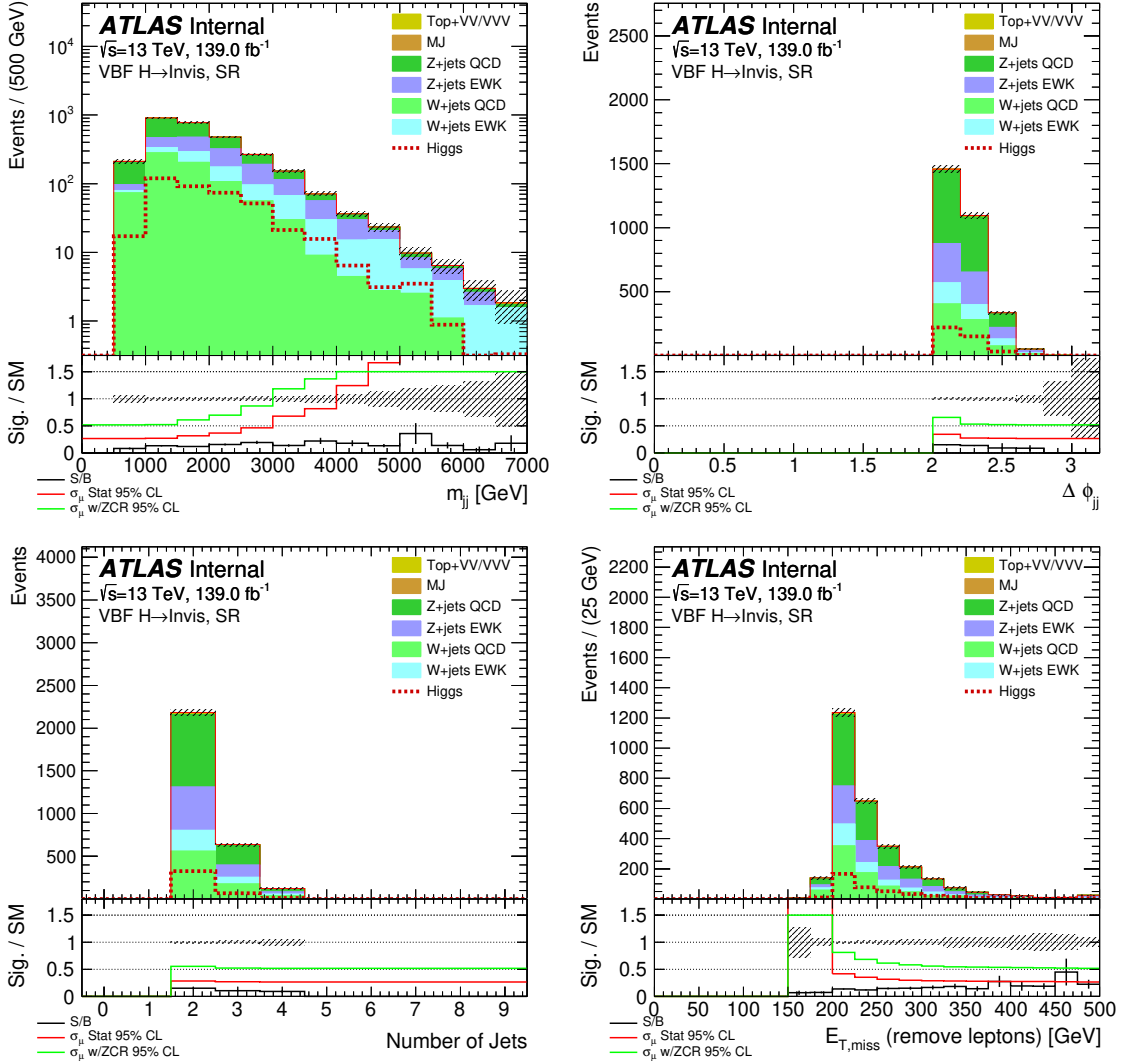


Figure 9.2: Pre-fit plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and E_T^{miss} in the blinded high- $\Delta\phi_{jj}$ validation region. No data is shown: the red dashed line shows a Monte Carlo simulation of the Higgs to invisible signal compared against Monte Carlo simulations of the various background processes. As expected, in this region the signal strength is quite small compared to the $\Delta\phi_{jj} < 2$ signal region shown in Figure 6.7. The error band includes both statistical and systematic uncertainties.

of $\Delta\phi_{jj}$, most of the events in this region appear to have $2 < \Delta\phi_{jj} < 2.5$. Due to limitations in the modelling of the multijet background for $\Delta\phi_{jj} > 2.5$, and due to the lack of statistics above 2.5, it was decided to limit the validation region to $2 < \Delta\phi_{jj} < 2.5$. To be as consistent as possible with the rest of the analysis, the validation region was split into the same five m_{jj} bins: $800 < m_{jj} < 1000$ GeV, $1000 < m_{jj} < 1500$ GeV, $1500 < m_{jj} < 2000$ GeV, $2000 < m_{jj} < 3500$ GeV, and $m_{jj} > 3500$ GeV.

An additional advantage of this validation region is that the 0-lepton high- $\Delta\phi_{jj}$ region will also include non-negligible contamination from multijet events with fake E_T^{miss} . A multijet background estimate was therefore performed using the rebalance and smear method described in Section 8.4, with normalization factors computed in two multijet-enriched low- E_T^{miss} ($100 < E_T^{\text{miss}} < 160$ GeV) regions: one with $2.8 < \Delta\eta_{jj} < 3.8$, and one with $\Delta\eta_{jj} > 3.8$, with the uncertainty taken as half the spread between the two for each data-taking period. These alternate normalization regions were used due to the limited statistics at high $\Delta\phi_{jj}$. The multijet contamination in the high- $\Delta\phi_{jj}$ W control region was also estimated using the same fake lepton estimate procedure introduced in Section 8.3, as no $\Delta\phi_{jj}$ dependence was seen on the fake lepton transfer factor.

m_{jj} Bin	k_V ($2 < \Delta\phi_{jj} < 2.5$)
0.8–1.0 TeV	0.992 ± 0.351
1.0–1.5 TeV	0.956 ± 0.221
1.5–2.0 TeV	0.977 ± 0.180
2.0–3.5 TeV	0.948 ± 0.120
>3.5 TeV	0.885 ± 0.120

Table 9.1: Best-fit values of k_V , the V+jets transfer factors used to normalize the W and Z backgrounds in the high- $\Delta\phi_{jj}$ validation region. Note that $k_V = \beta_V$ in the likelihood function shown above in Section 9.1.

Using these inputs, the likelihood fit was then performed in this $2 < \Delta\phi_{jj} < 2.5$ validation region. Table 9.1 shows the best-fit transfer factors computed during the fit process for the V backgrounds. For the fake leptons, normalization factors β_e and β_μ can be defined as the post-fit data/MC agreement in the low- S_{MET} and low- m_T control regions, respectively. Their inclusive values were found to be $\beta_e = 1.16 \pm 0.53$ and $\beta_\mu = -0.27 \pm 0.46$. Table 9.2 then shows the numbers of data and background events recorded in each region after running the fit. Figure 9.4 shows the post-fit agreement in the five m_{jj} bins for the one- and two-lepton control regions, their corresponding fake lepton control regions, and the 0-lepton “signal” region. Good agreement between data and background prediction is seen in all the bins, and very low signal contamination is seen in the 0-lepton region, as expected. The best-fit signal strength was determined to be 0.37 ± 0.34 , which is almost consistent within one σ with there being no signal at all, again as expected. While this region is clearly not sensitive to the invisible Higgs process, we can attempt to set a 95% confidence level limit using this fit. The observed (expected) limit was found to be 0.94 (0.67), with very large uncertainties as shown below in Table 9.3.

We are of course not trying to set a limit using this validation region: we are only interested in confirming that the fit appears stable. The largest systematic uncertainties, and how they were

Samples	SR	Z $\rightarrow ll$ CR	W $\rightarrow e\nu$ CR	W $\rightarrow \mu\nu$ CR	W $\rightarrow l\nu$ CR	Fake- e CR	Fake- μ CR
Z EWK	1060.4 $\pm$ 116.9	246.1 $\pm$ 27.3	4.3 $\pm$ 0.7	12.1 $\pm$ 0.7	16.4 $\pm$ 1.0	10.2 $\pm$ 0.7	4.2 $\pm$ 0.7
Z strong	2323.6 $\pm$ 135.6	454.3 $\pm$ 33.6	9.1 $\pm$ 2.6	58.3 $\pm$ 4.6	67.5 $\pm$ 5.2	37.0 $\pm$ 6.5	21.6 $\pm$ 4.0
W EWK	543.4 $\pm$ 61.5	0.1 $\pm$ 0.1	708.4 $\pm$ 69.7	1344.5 $\pm$ 133.3	2053.0 $\pm$ 150.4	685.3 $\pm$ 69.5	236.3 $\pm$ 25.8
W strong	1383.5 $\pm$ 105.8	0.2 $\pm$ 0.1	960.3 $\pm$ 71.5	1734.4 $\pm$ 128.7	2694.7 $\pm$ 147.2	872.8 $\pm$ 65.7	384.1 $\pm$ 30.5
Multijet	457.4 $\pm$ 73.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0
μ -fakes	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	-8.4 $\pm$ 1.6	-8.4 $\pm$ 1.6	0.0 $\pm$ 0.0	-25.4 $\pm$ 7.4
e -fakes	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	30.3 $\pm$ 12.7	0.0 $\pm$ 0.0	30.3 $\pm$ 12.7	185.0 $\pm$ 44.6	0.0 $\pm$ 0.0
Other	61.1 $\pm$ 6.8	5.9 $\pm$ 1.2	83.0 $\pm$ 5.4	104.1 $\pm$ 6.4	187.1 $\pm$ 8.3	26.2 $\pm$ 2.2	22.1 $\pm$ 2.5
Data	5849	710	1783	3236	5019	1824	644
Total Bkg	5829.4 $\pm$ 74.6	706.6 $\pm$ 17.5	1795.5 $\pm$ 32.9	3245.0 $\pm$ 48.5	5040.5 $\pm$ 58.6	1816.6 $\pm$ 33.4	642.9 $\pm$ 19.7
Data/Bkg	1.003	1.005	0.993	0.997	0.996	1.004	1.002
VBFH125	786 $\pm$ 70	-	-	-	-	-	-
ggFH125	150 $\pm$ 68	-	-	-	-	-	-
VH125	2.16 $\pm$ 0.18	-	-	-	-	-	-

Table 9.2: Post-fit yields of signal, background, and data events recorded in the high- $\Delta\phi_{jj}$ validation region. Signal yields are normalized to $\mathcal{B}_{H \rightarrow \text{inv.}} = 0.94$, the observed limit from this validation region fit. The “Other” background includes contributions from both single- t , $t\bar{t}$, and multiboson (VV/VVV) backgrounds. These yields are inclusive; the five m_{jj} bins have been combined in each region.

Expected	Observed	+1 σ	-1 σ	+2 σ	-2 σ
0.67	0.94	0.93	0.48	1.26	0.36

Table 9.3: Observed and expected 95% confidence level limits on $\mathcal{B}_{H \rightarrow \text{inv.}}$ from a fit to the high- $\Delta\phi_{jj}$ validation region, with both statistical and systematic uncertainties included.

pulled during the fit process, are shown in Figure 9.3. The three largest uncertainties appear to be the electroweak reweighting uncertainty on the Z/W ratio (described in Section 8.2), followed by two systematics on the multijet estimate. In general, these leading systematics do not appear to be pulled significantly during the fit, suggesting the overall process is stable. This stability and the good agreement between data and MC seen in Figure 9.4 helps to confirm that everything has been implemented correctly for the final 139 fb $^{-1}$ analysis.

9.3 Unblinded Signal Region

The fit model is the last major component of the analysis; having validated it as described in the above section, we can finally unblind¹¹⁸. Figure 9.5 shows the m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and $E_{\text{T}}^{\text{miss}}$ distributions in the unblinded signal region, with data as well as Monte Carlo simulations of both the signal and background. Note that the $\mathcal{R}^{Z/W}$ corrections have already been applied to the Z processes, as described in Section 8.2. The agreement between data and MC is not perfect in every bin, but this is entirely expected, as we have not run the likelihood fit yet. The background prediction

¹¹⁸Once permission was given by the collaboration, as described in the introduction.

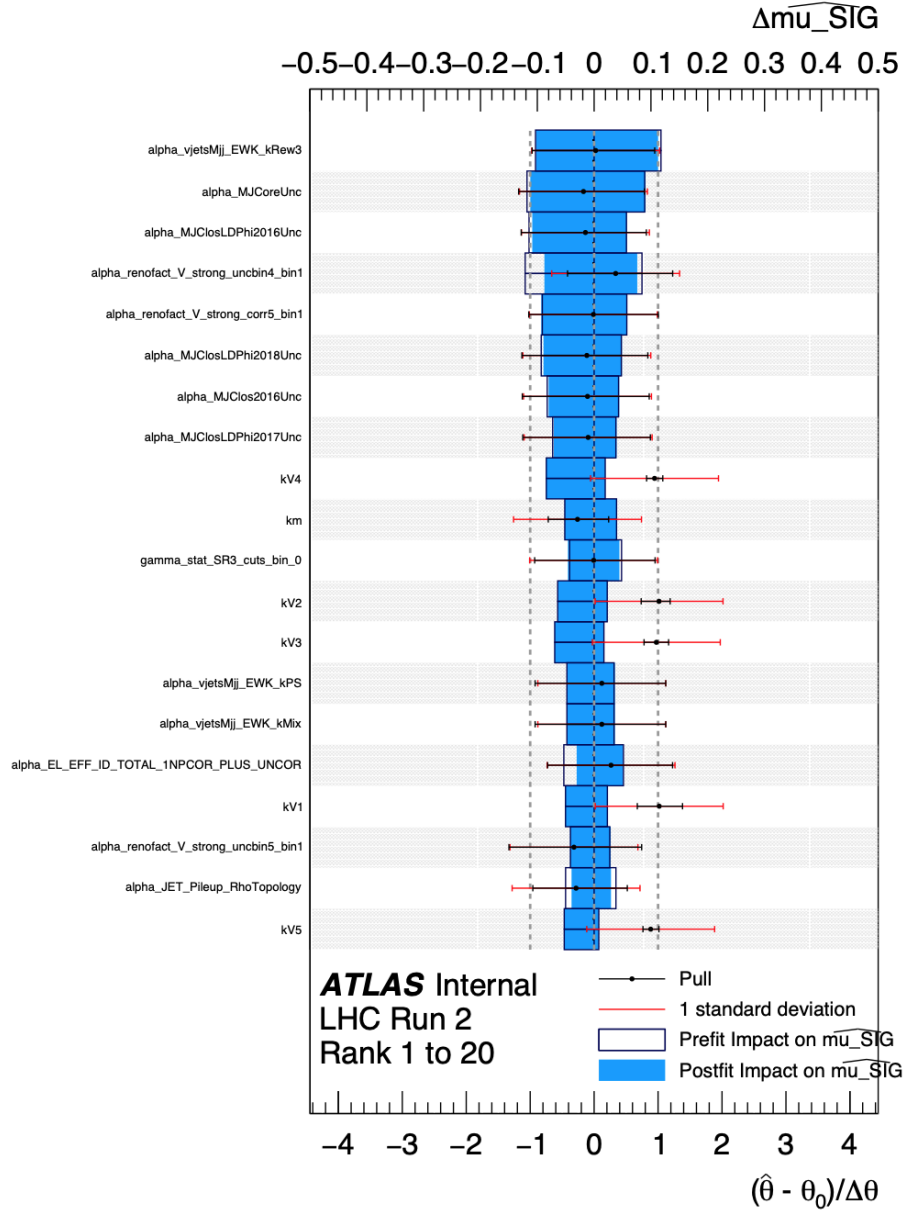


Figure 9.3: Ranking plot showing the twenty nuisance parameters which had the largest impact on the fit in the high- $\Delta\phi_{jj}$ validation region. The black dot shows the “pull” of the parameter, meaning the extent to which it was shifted up or down during the fit, and the blue band shows the impact on the likelihood.

in all the bins will change once that is done. Table 9.4 gives the number of data, background, and signal events measured in each region, and Figure 9.6 shows the yield in each of the bins in each of the regions, again all before applying the fit. These plots and numbers are presented here primarily

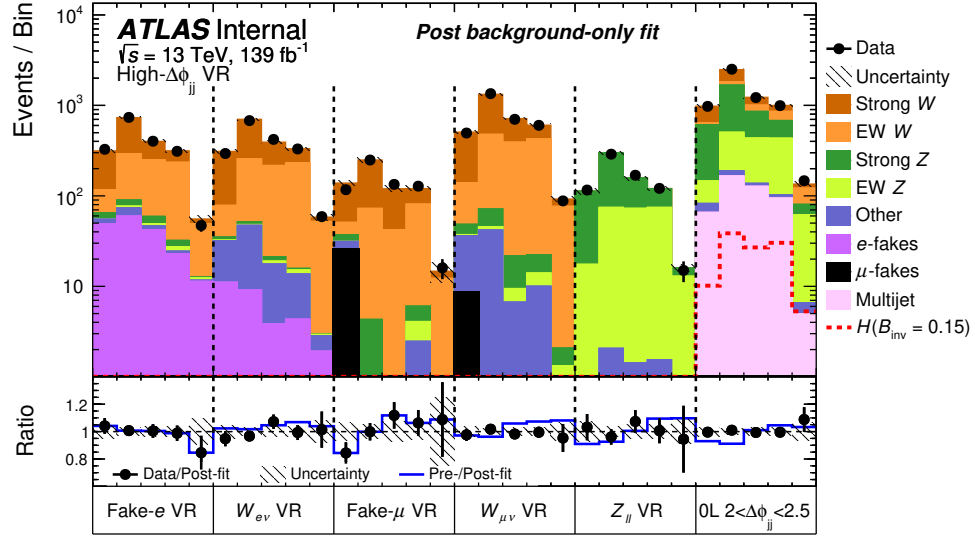


Figure 9.4: Post-fit plot, showing agreement between data and Monte Carlo in all five bins of the high- $\Delta\phi_{jj}$ validation region, with both statistical and systematic errors shown. The low- S_{MET} fake electron and low- m_T fake muon estimation regions are shown in addition to the one- and two-lepton control regions and the “signal region” with $2 < \Delta\phi_{jj} < 2.5$; as expected, very low contamination from the invisible Higgs signal is seen in this region. The blue line in the bottom panel shows the shifts in each bin before and after the fit; in general, the fit appears quite stable.

to illustrate the impact of the fit, and allow the reader to make a before-and-after comparison with the results shown in the next section.

Samples	SR	$Z \rightarrow ll$ CR	$W \rightarrow e\nu$ CR	$W \rightarrow \mu\nu$ CR	$W \rightarrow l\nu$ CR	Fake- e CR	Fake- μ CR	FJVT CR
VBFH125	5908.254	0.000	0.164	0.000	0.000	0.000	0.000	26.279
ggFH125	708.411	0.000	0.000	0.000	0.000	0.000	0.000	6.613
VH125	5.918	0.012	0.894	1.038	1.9	0.057	0.178	0.077
Z EWK	2625.1 ± 18.9	618.1 ± 3.5	12.1 ± 1.0	28.3 ± 1.1	40.4 ± 1.5	23.5 ± 0.8	15.3 ± 1.0	7.2 ± 1.5
Z strong	6032.0 ± 49.8	1216.5 ± 14.3	41.5 ± 11.3	142.5 ± 6.3	184.0 ± 12.9	146.4 ± 40.1	39.5 ± 4.3	39.6 ± 10.0
W EWK	1613.2 ± 22.6	0.0 ± 0.0	2356.3 ± 25.4	3411.4 ± 30.3	5767.7 ± 39.6	1402.2 ± 20.0	821.8 ± 14.8	22.9 ± 3.9
W strong	3709.9 ± 82.0	0.5 ± 0.1	3260.6 ± 43.6	5170.0 ± 55.0	8430.6 ± 70.1	1814.0 ± 29.2	1014.6 ± 23.9	48.7 ± 19.0
Other	179.5 ± 3.6	46.2 ± 1.3	345.6 ± 5.0	350.8 ± 5.2	696.3 ± 7.2	67.4 ± 2.4	89.1 ± 2.4	19.5 ± 1.1
Multijet	832.6 ± 23.4	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1895.1 ± 32.6
Data	16490	2051	6361	9294	15655	4563	2110	2033
Total Bkg	14992.3 ± 103.1	1881.3 ± 14.8	6214.4 ± 62.7	9146.2 ± 63.4	15360.5 ± 89.2	4563.0 ± 98.0	2110.0 ± 30.7	2033.0 ± 39.3
Data/Bkg	1.10	1.09	1.024	1.016	1.019	1.0	1.0	1.0

Table 9.4: Pre-fit yields of signal, background, and data events recorded in the signal region, the one- and two-lepton control regions, the fake electron and fake muon control regions, and the multijet (FJVT) control region. The signal predictions are normalized to $\mathcal{B}_{H \rightarrow \text{inv.}} = 0.15$. The “Other” background includes contributions from both single- t , $t\bar{t}$, and multiboson (VV/VVV) backgrounds.

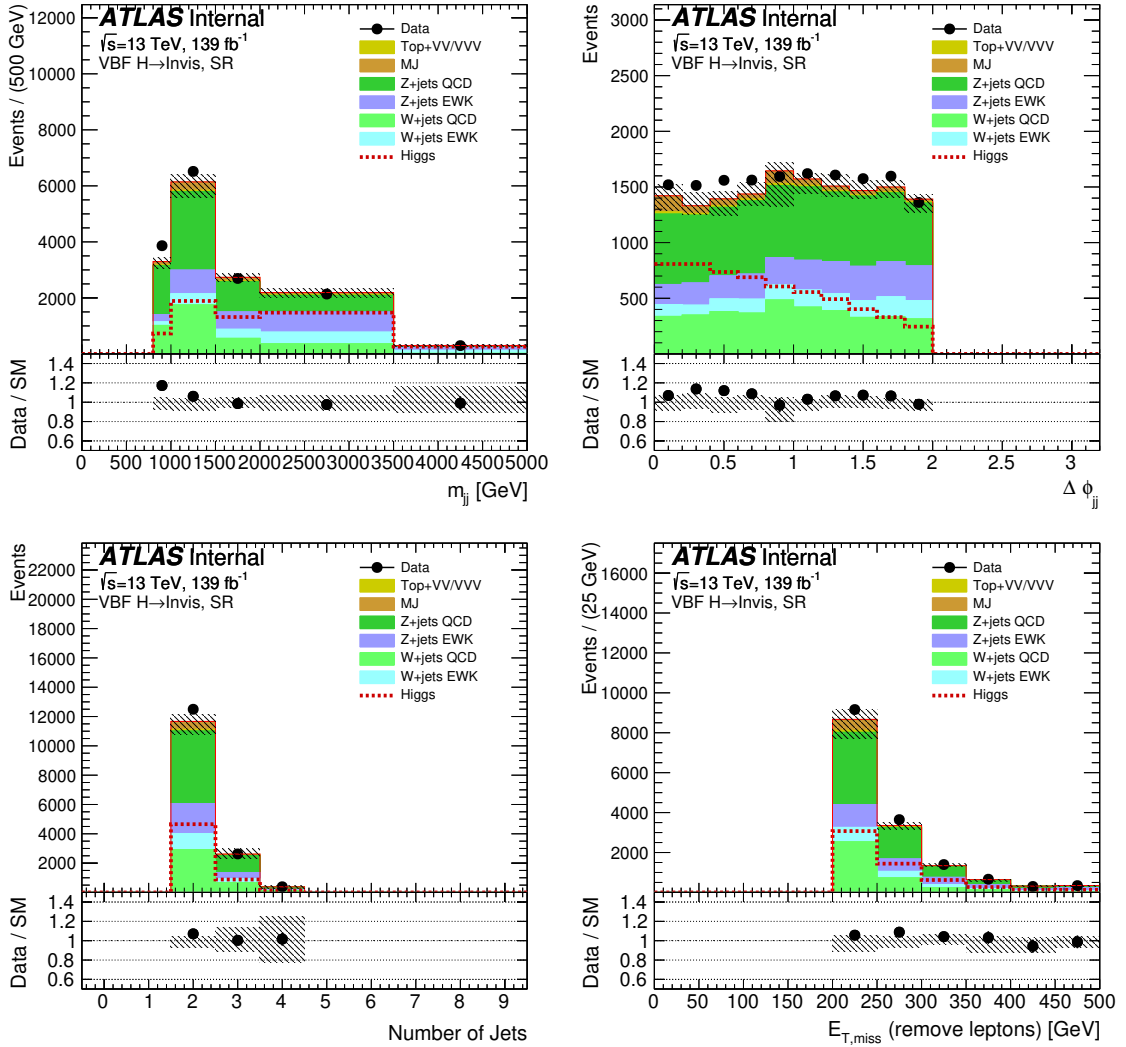


Figure 9.5: Plots of m_{jj} , $\Delta\phi_{jj}$, N_{jets} , and E_T^{miss} in the unblinded VBF+MET signal region. Unlike Figure 6.7, data is shown in addition to simulation. The error band includes both statistical and systematic uncertainties. While these plots were generated without applying the full fit, the W and Z MC was normalized in each bin using their respective control regions; note this was done for the preliminary 139 fb^{-1} analysis, and without using the W to constrain the Z as described in Section 8.2.

9.4 Unblinded Fit Results

The likelihood fit is then applied simultaneously to all the bins of all the regions, using the procedure described in Section 9.1. The fit applies the procedures discussed in Chapter 8 and rescales the background prediction in every bin to try and match the unblinded data. In the signal region, the

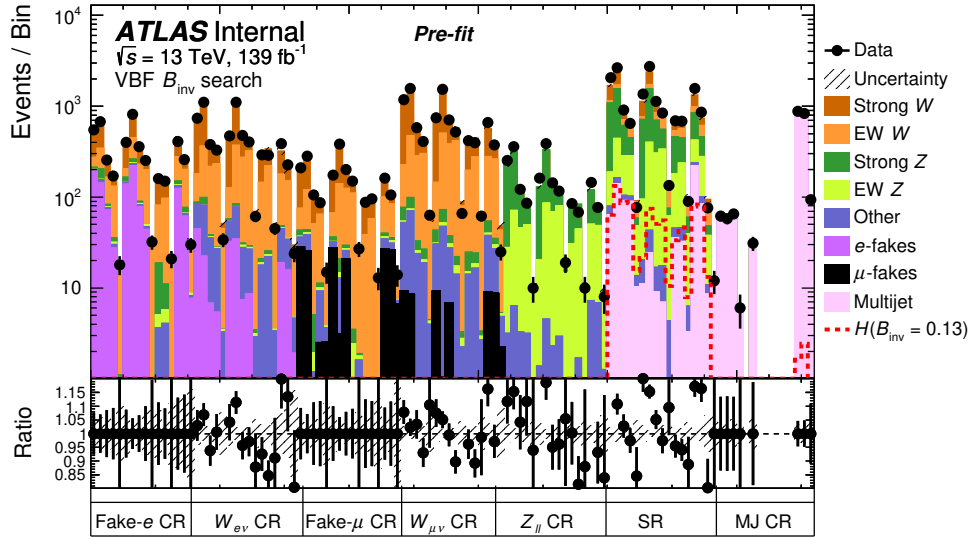


Figure 9.6: Plot showing the pre-fit agreement between data and background in all sixteen bins of all the main analysis regions: the signal region, the multijet FJVT control region, the one- and two-lepton control regions, and the corresponding fake lepton control regions. The error band includes both statistical and systematic uncertainties.

signal strength in each bin is also allowed to float, meaning that it will be varied simultaneously with the background to better fit the data. The best-fit parameters for the various transfer factors for V+Jets, fake leptons, and multijet backgrounds are shown in Table 9.5: some of these transfer factors are significantly shifted away from unity, showing the effects of the fit. The post-fit agreement between data and background is shown in Figure 9.7. Comparing this plot to the pre-fit version shown in Figure 9.6, we can see that the agreement between data and background is considerably improved in both the control and signal regions.

In most bins, no significant excess above the Standard Model background prediction is seen in most of the bins after the fit. The main exception appears to be the tenth bin, where we require $E_T^{\text{miss}} > 160 \text{ GeV}$, $m_{jj} > 3.5 \text{ TeV}$, $N_{\text{jets}} = 2$, and $1 < \Delta\phi_{jj} < 2$, where the data appears to be roughly one σ greater than the background. This excess is not significant, and so we proceed with setting a limit on the branching ratio $\mathcal{B}_{H \rightarrow \text{inv.}}$.

9.4.1 Limit Setting

Using the full run 2 ATLAS dataset, we set a 95% confidence level observed (expected) limit on $\mathcal{B}_{H \rightarrow \text{inv.}} = 0.145(0.103)$ (or rounded to 0.15 (0.11), as quoted in the abstract). The uncertainties

Bin	k_V	β_e	β_μ	β_{MJ}
1	1.21 ± 0.02	0.66 ± 0.12	0.09 ± 0.52	1.05 ± 0.30
2	1.09 ± 0.01	0.76 ± 0.18	0.39 ± 0.66	0.93 ± 0.12
3	1.03 ± 0.02	0.94 ± 0.22	-0.72 ± -1.88	0.92 ± 0.12
4	0.97 ± 0.03	1.36 ± 0.47	3.73 ± 3.72	0.88 ± 0.11
5	0.85 ± 0.07	3.70 ± 10.39	2.35 ± 1.61	0.90 ± 0.37
6	1.23 ± 0.02	0.69 ± 0.14	-0.12 ± 0.48	0.90 ± 0.20
7	1.16 ± 0.01	0.74 ± 0.13	-13.32 ± 6.61	
8	0.99 ± 0.02	1.02 ± 0.24	1.22 ± 0.69	
9	0.95 ± 0.02	1.08 ± 0.36	-1.56 ± -3.38	
10	0.94 ± 0.05	0.16 ± -3.02	1.03 ± 8.00	—
11	1.01 ± 0.03	11.18 ± 28.87	1.00 ± -0.78	
12	0.86 ± 0.02	18.74 ± 13.38	-0.37 ± -0.65	
13	0.93 ± 0.06	0.78 ± -1.82	-0.30 ± -2.10	—
14	1.24 ± 0.03	0.60 ± 0.16	0.59 ± 0.47	0.95 ± 0.03
15	1.06 ± 0.03	0.92 ± 0.25	0.82 ± 0.39	1.05 ± 0.04
16	0.62 ± 0.07	2.07 ± 0.68	0.99 ± 0.53	1.03 ± 0.12

Table 9.5: Best-fit values of k_V , the V+jets normalization factors used to correct the W and Z backgrounds, the fake electron and fake muon normalization factors β_μ and β_e , and the multijet (FJVT) normalization factors β_{MJ} in all sixteen bins. The multijet factor β_{MJ} is only shown in the bins for which the FJVT method is used, as described in Section 8.4, with a single factor in bins 6-10. The bin numbering used here was introduced in Section 6.3.4.

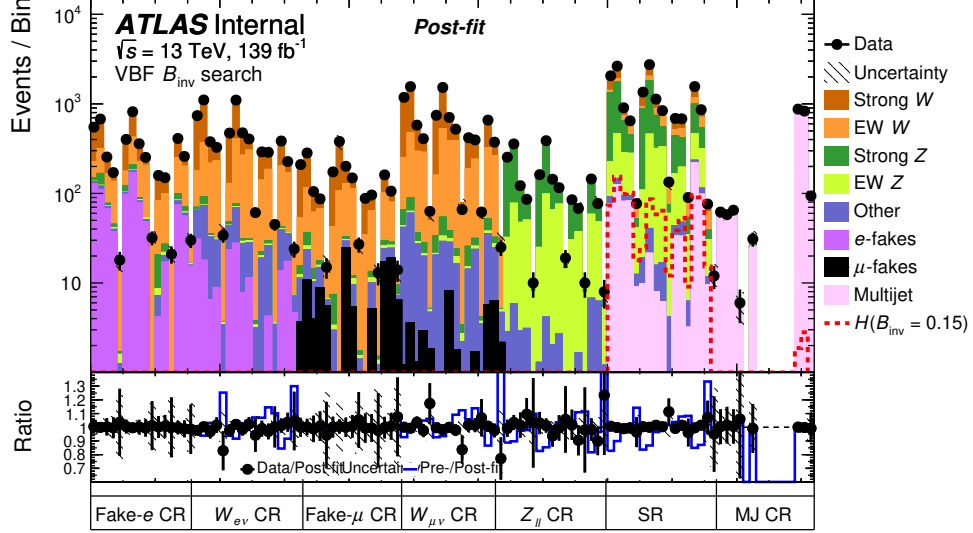


Figure 9.7: Plot showing the post-fit agreement between data and background in all sixteen bins of all the main analysis regions, as well as the Higgs to invisible signal normalized to a branching ratio of $\mathcal{B}_{H \rightarrow \text{inv.}} = 0.15$. The post-fit agreement can be compared directly to the pre-fit values seen in Figure 9.6 to see the impact of the fit on the background prediction in each bin. No significant data excess over the background is observed in any bin in the signal region.

on this limit are shown below in Table 9.6. The observed limit of 0.145 is roughly one σ higher than the expected limit of 0.103, due to the small excess observed in the tenth bin that’s visible in Figure 9.7. This represents a substantial improvement from the previous best limits presented in Section 6.1.2: recall that the previous 36.1 fb^{-1} VBF+MET analysis set a limit of 0.37 (0.28) [135].

Expected	Observed	+1 σ	-1 σ	+2 σ	-2 σ
0.103	0.145	0.144	0.075	0.195	0.056

Table 9.6: Observed and expected limits on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$, set at a 95% confidence level using the likelihood fit shown in Figure 9.7, with both statistical and systematic uncertainties included.

This limit can be compared to the results of the *preliminary* 139 fb^{-1} analysis, which were presented in Appendix C. In that version of the analysis, the observed (expected) limit was 0.132(0.132). This version of the analysis was performed without using the W to constrain the Z : instead, two separate V+jets transfer factors were used and the $Z \rightarrow \nu\nu$ background was solely estimated from the two-lepton control region. This change, as well as other improvements to the background modelling described in Appendix C, led to an improvement in the expected limit sensitivity but also led to the observed limit getting worse, as previously no excess was observed in any of the signal region bins¹¹⁹

9.4.2 Background-Only Fit

The limit set during the fit is consistent with the Standard Model hypothesis. Even though the invisible Higgs signal was allowed to float during the fit, the final result is consistent with no signal being present in any bin. To help verify that this is the case, we perform a second “background-only” fit, where the Higgs signal is *not* allowed to float. In this version of the fit, only the background prediction in each bin can be changed. Figure 9.8 shows the results of the background-only fit. The fit agreement remains quite good, and the data/background agreement very similar to Figure 9.8. This helps confirm that the signal is really not necessary to explain the observed data in the signal region.

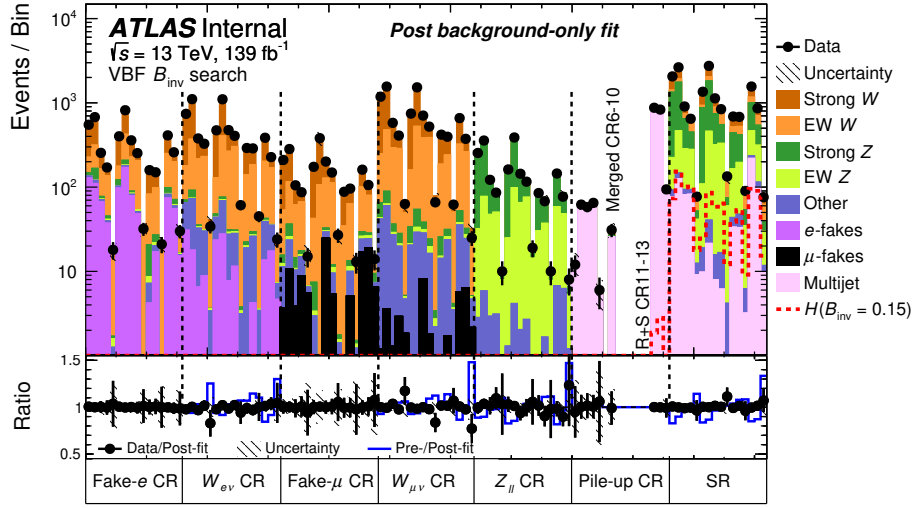


Figure 9.8: Plot showing the post-fit agreement between data and background in all sixteen bins of all the main analysis regions in a “background-only” fit. In the background-only fit, the signal strength μ is set to zero, meaning that the fit will attempt to fit the signal region data *only* by changing the background. No significant difference is seen between this fit result and the result with the signal strength allowed to float, shown above in Figure 9.7.

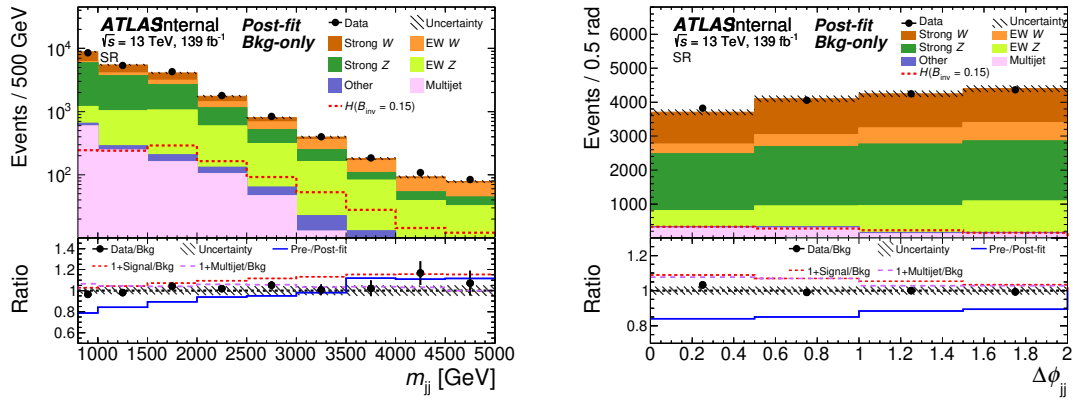


Figure 9.9: Post-fit plots showing the m_{jj} and $\Delta\phi_{jj}$ distributions in the signal region. The data and background agree much better after the fit than in comparison with the pre-fit plots shown in Figure 9.5.

9.4.3 Postfit Yields and Distributions

Figure 9.9 shows post-fit distributions of m_{jj} and $\Delta\phi_{jj}$ in the signal region. In these plots, the Higgs to invisible signal was normalized to a branching ratio of 0.15, to be consistent with the observed limit. The agreement between data and background is much improved after running the fit compared with the pre-fit distributions shown in Figure 9.5 above. The number of data, signal, and background events observed in the signal region and the various control regions after the fit are shown in Table 9.7, again with the signal normalized to a branching ratio of 0.15. As shown in this table, the ratio of data to background events is very close to 1 in all the regions, again indicating very good agreement with the Standard Model hypothesis.

Samples	SR	Z $\rightarrow$ ll CR	W $\rightarrow$ $e\nu$ CR	W $\rightarrow$ $\mu\nu$ CR	W $\rightarrow$ $l\nu$ CR	Fake- e CR	Fake- μ CR	FJVT CR
Multijet	814.8 ± 105.4	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1887.4 ± 76.2
μ -fakes	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	16.0 ± 37.9	16.0 ± 37.9	0.0 ± 0.0	48.0 ± 114.0	0.0 ± 0.0
e -fakes	0.0 ± 0.0	0.0 ± 0.0	174.6 ± 44.2	0.0 ± 0.0	174.6 ± 44.2	901.8 ± 139.6	0.0 ± 0.0	0.0 ± 0.0
Other	177.3 ± 9.3	42.8 ± 5.1	330.4 ± 16.8	340.2 ± 15.7	670.6 ± 23.0	63.5 ± 3.8	84.9 ± 6.5	17.9 ± 4.2
Z EWK	2778.5 ± 150.8	637.8 ± 35.3	12.7 ± 1.2	27.6 ± 1.3	40.3 ± 1.8	24.4 ± 1.1	15.7 ± 1.6	7.2 ± 2.3
Z strong	7011.6 ± 175.8	1365.1 ± 43.5	44.5 ± 11.6	145.1 ± 8.2	189.6 ± 14.2	155.6 ± 42.2	48.9 ± 10.3	69.3 ± 21.7
W EWK	1658.2 ± 84.1	0.0 ± 0.0	2382.6 ± 114.9	3445.8 ± 162.5	5828.3 ± 199.0	1451.0 ± 79.1	830.1 ± 48.1	20.1 ± 6.5
W strong	4004.8 ± 146.0	0.5 ± 0.2	3431.1 ± 119.8	5359.0 ± 171.2	8790.1 ± 208.9	1967.6 ± 86.7	1078.4 ± 67.0	31.4 ± 42.6
Data	16490	2051	6361	9294	15655	4563	2110	2033
Total Bkg	16445.1 ± 118.8	2046.1 ± 29.7	6375.9 ± 62.6	9333.7 ± 84.3	15709.6 ± 105.0	4563.8 ± 67.5	2106.0 ± 80.3	2033.3 ± 46.6
Data/Bkg	1.003	1.002	0.998	0.996	0.997	1.0	1.002	1.0
VBFI125	1015 ± 82	—	—	—	—	—	—	—
ggFI125	132 ± 63	—	—	—	—	—	—	—
VH125	1.07 ± 0.08	—	—	—	—	—	—	—

Table 9.7: Post-fit yields of signal, background, and data events recorded in the signal region, the one- and two- lepton control regions, the fake electron and fake muon control regions, and the multijet (FJVT) control region. These yields can be compared with the pre-fit numbers shown in Table 9.4 to see the impact of the likelihood fit. As with the numbers shown there, the Higgs to invisible signal is normalized to a branching ratio of $\mathcal{B}_{H \rightarrow \text{inv.}} = 0.15$.

9.4.4 Uncertainties

The uncertainties on the expected limit are shown in Table 9.6 above, and consist of both statistical and systematic uncertainties. Figure 9.10 shows the individual systematics which appear to have the largest impact on the fit: note that the first few are the V+Jets transfer factors k_V , followed by an uncertainty on the multijet estimate. The various uncertainties can be grouped together into different categories: statistical uncertainties, V+Jets systematics, multijet estimate systematics, reconstruction systematics, uncertainties on the signal modelling, and so on, in order to determine

¹¹⁹While the two results are statistically consistent, one could argue that, because the data did not change between the two iterations of the analysis, and because the one σ excess seems to arise due to the various background modelling improvements that were introduced, that the significance might be slightly *greater* than one σ . The analysis will need to be repeated with more data from run 3 and from the high-luminosity LHC upgrade in order to really determine whether this is just a statistical fluctuation and to improve the sensitivity further.

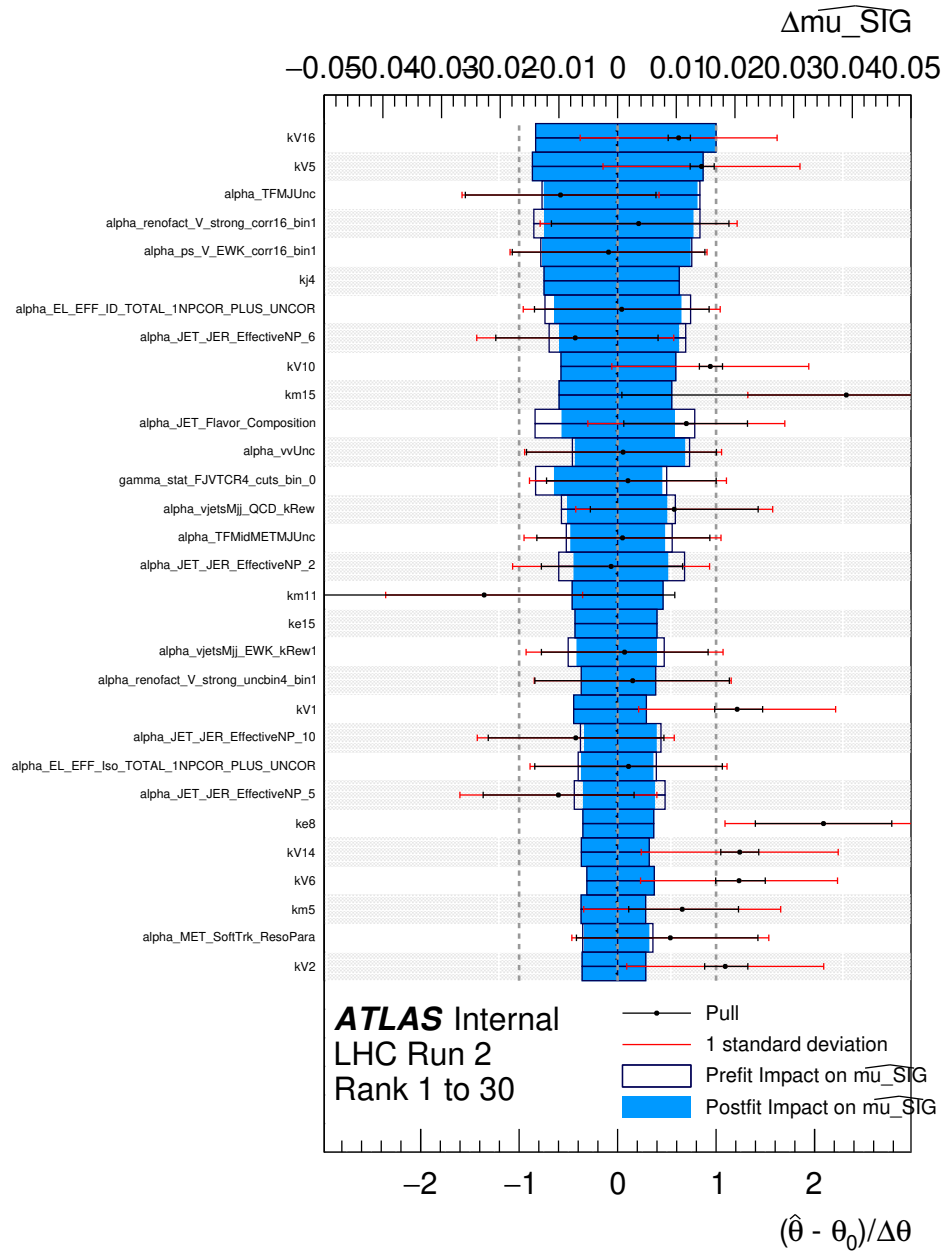


Figure 9.10: Ranking plot showing the thirty nuisance parameters that have the largest impact on the signal region fit, formatted the same way as Figure 9.3. The parameters with the largest impact are two of the V+jets normalization factors, followed by a multijet uncertainty, followed by additional V+jets systematics: this is roughly expected given the importance of these backgrounds.

which groups have the largest impact on the result. Table 9.8 shows the uncertainties grouped and sorted in this manner, listing both the contribution each group makes to the uncertainty on the limit, as well as how much the limit would improve if the group in question could be removed.

As we can see from these results, data statistics overall— both in the signal region and on the V+jets control regions— have the largest impact both on the limit and on the uncertainty on the limit. This is good, because it means that the analysis is not systematically limited: it will improve if more data from future runs is added. The next largest groups are the multijet and fake lepton systematics. It is somewhat surprising to see that the fake lepton systematics appear to be so significant, but as we now use the W to constrain the Z , the fake lepton backgrounds now impact the estimation of both the $Z \rightarrow \nu\nu$ and $W \rightarrow l\nu$ lost lepton backgrounds. Additionally, the techniques described in Section 8.3 that are used for these backgrounds are not that sophisticated, and relatively conservative uncertainties are assigned on the fake lepton transfer factors. Future iterations of the analysis would likely be able to employ more advanced methods to estimate the misidentified lepton contamination in order to bring these uncertainties down.

The large uncertainties on the multijet estimate, on the other hand, are expected, as described in Section 8.5. It was known from the preliminary analysis that these uncertainties are quite large, especially in the region with $160 < E_{\text{T}}^{\text{miss}} < 200 \text{ GeV}$. Even with the improvements for the final 139 fb^{-1} analysis and the introduction of the FJVT-based multijet transfer factors, these systematics remain quite large. Additional work to find additional ways of improving the multijet background estimate will likely be needed in order to improve the sensitivity in future iterations of the analysis.

Finally, it is good to see that after the introduction of m_{jj} -sliced V+jets samples, as discussed in Chapter 7, and the generation of sufficient statistics, that the MC statistical uncertainties are no longer dominant. This is a major improvement in this result compared to the 36.1 fb^{-1} result described in Section 6.1.2 [135], and even compared to the preliminary 139 fb^{-1} result described in Appendix C. Further work to improve the V+jets modelling at the MC generator level and fix some of the problems described in Chapter 7 would likely still be useful, but this no longer appears to be the main limitation of the analysis.

9.5 Interpretations

Having set a limit on the invisible Higgs branching ratio, we now consider ways in which this result can be interpreted. A number of different new physics scenarios might involve the Higgs decaying invisibly at rates higher than predicted by the Standard Model. The limit on the branching ratio

Source	Contribution to $\pm 1\sigma$	Limit Change ($\Delta\%$)
Data stats.	0.022	8.7
V+jets data stats.	0.015	9.4
MC stats.	0.010	3.8
Multijet	0.014	5.0
μ/e -fakes	0.014	6.5
Leptons	0.011	5.3
JER	0.011	4.2
JES	0.008	2.1
Remaining	0.010	2.8
V+jets theory	0.012	4.2
Signal theory	0.009	0.6

Table 9.8: Table showing the various statistical and systematic uncertainties, grouped together into several key categories. The second column shows the amount each group contributes to the one σ uncertainty on the expected limit; if the fit was re-ran with the group in question was removed, the total uncertainty would be reduced by the quoted amount. The third column shows the percent impact on the expected limit rather than the uncertainty on the limit; these values are provided for direct comparison with the 36.1 fb^{-1} analysis, which quoted uncertainties in this way.

can therefore be interpreted as a limit on these various models, the most notable of which is the “Higgs portal dark matter” scenario discussed in Chapters 2 and 6.1 [8]. The $\mathcal{B}_H \rightarrow \text{inv.}$ limit can be interpreted as a limit on the existence of various dark matter candidates, and compared directly to results from dark matter direct detection experiments. This interpretation is presented in this section.

Of course, these results can be used to set limits on other models beyond Higgs portal dark matter. The results from all ATLAS searches are made publicly available using the RECAST framework [201] in order to make it as easy as possible for theorists to interpret these results in the context of as many models as possible.

9.5.1 Higgs Portal Dark Matter

Direct detection dark matter experiments, like LUX [39] or PandaX [40], are looking for evidence of weakly interactive massive particles across a varying mass range. Because they look for evidence of WIMPs interacting directly with the nuclei of various elements, they present their results as a limit on the spin-independent WIMP-nucleon interaction cross section as a function of the mass of the WIMP. In order to compare our invisible Higgs limit with their results, we need to express it in this form as well. To do so, we need to write down a Lagrangian describing the interaction of a particular WIMP candidate with the Higgs portal. From the Lagrangian, we can write down an expression for the Higgs branching ratio to the WIMP, which is what we have set a limit on in this

analysis. The branching ratio expression can be rearranged to set a limit on the WIMP mass, which can then be plugged into an expression for the WIMP-nucleon cross section [44].

There are several different Lagrangians one could write, depending on the nature of the WIMP. A scalar, vector, or fermionic WIMP will behave differently, and couple differently to the Higgs. For this interpretation, we use the four models introduced in Chapter 2: a scalar model [44], a Majorana fermion model [44], and two vector models, one that is UV-complete [202] and one that is an effective field theory¹²⁰ [46]. The UV-complete vector model includes an additional scalar boson, and therefore the limit will depend on the mass m_2 of that particle. Several different limits were computed by varying m_2 to illustrate how the choice of this parameter would affect the limit. Additionally, to perform this interpretation, a nucleon “target” must be chosen to calculate a WIMP-nucleon cross section. As the limits are computed over a wide mass range, the choice does not have a large impact, and so germanium was selected. And finally, the dark matter experiments report their results as limits at a 90% confidence level limits rather than a 95% confidence level as done in Table 9.6, so the VBF+MET limit must be recomputed.

The 90% observed (expected) confidence level limit was found to be 0.127(0.087). This interpretation was then performed, and the results are shown in figure 9.11. Here, the limit on $\mathcal{B}_H \rightarrow \text{inv.}$ has been reinterpreted as a limit on the four Higgs portal Lagrangian models [47], and compared to results from three direct detection experiments, PandaX-4T [203], DarkSide-50 [204], and CRESST-III [205]. The limits from this analysis are very complementary with the direct detection results in general, and especially help to cover the region $m_{\text{WIMP}} < 10 \text{ GeV}$, where the direct detection experiments appear less sensitive. The sensitivity of the Higgs portal limit falls off considerably at around 60 GeV, or approximately half the Higgs mass of 125 GeV, as we would expect. It is also interesting to see the different behavior between the different Higgs portal limits: these models become increasingly sensitive as the WIMP mass continues to get smaller, as a consequence of the mass dependence of the Lagrangians shown in Chapter 2. Finally, note that the so-called “neutrino floor”, the expected cross section from neutrino-nucleus scattering, is also shown for comparison purposes [206]. Much like there is no way to distinguish E_T^{miss} due to neutrinos from E_T^{miss} from other invisible particles, neutrino-nucleon scattering is indistinguishable from WIMP-nucleon scattering. This therefore represents the lowest limit that can be achieved from direct detection, but

¹²⁰The preliminary version of this analysis only used the scalar and fermion models, as controversy over whether the non-UV-complete vector model(s) were reasonable [45] had resulted in their removal from previous versions of this plot. As discussed in Chapter 2, however, recent theoretical calculations have shown that the vector model is actually an effective field theory limit of a model that *is* UV-complete, and therefore reasonable to include as part of this interpretation [46] [47].

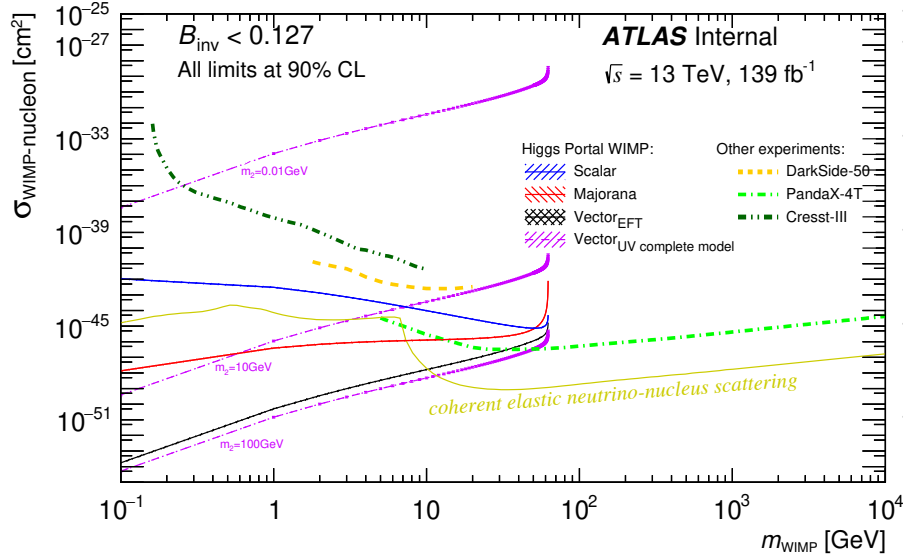


Figure 9.11: Comparison of 90% confidence level upper limits on the spin-independent WIMP-nucleon cross section between dark matter direct detection experiments [203] [204] [205] and the Higgs portal dark matter interpretation of the 139 fb⁻¹ VBF+MET analysis, as a function of WIMP mass. The blue, red, black, and purple lines show Higgs portal interpretations for different dark matter models, which are described in Chapter 2 [47]. The selected results from direct detection experiments are the most sensitive limits currently available across different m_{WIMP} ranges. As shown in the plot, the two approaches are very complementary.

as shown in the plot, some of the Higgs portal models are already more sensitive than the neutrino floor over some mass ranges.

9.5.2 Heavier Scalar Mediators

The VBF+MET analysis looks for invisible decays of a Higgs boson produced via vector boson fusion. However, there is no reason the particle in question that decays invisibly *has* to be a Higgs. Consider the analysis signature discussed in Chapter 6: we require evidence of vector boson fusion production, and we require a large enough missing transverse momentum that *could* have come from the decay of the Higgs, but we have no real way of distinguishing a boosted Higgs from, say, a new, undiscovered 500 GeV scalar particle. Higgs to Invisible signal MC is used during the limit-setting, but in principle, there is no reason that limits could not be set using for different scalar mediators instead. Additional signals samples were generated as described in Section 6.5 across a variety of “Higgs” mass points. These samples were then used to compute limits, as shown in Figure 9.12, across a mediator mass range from 50 to 3000 GeV. As seen in the plot, a consistent one σ excess

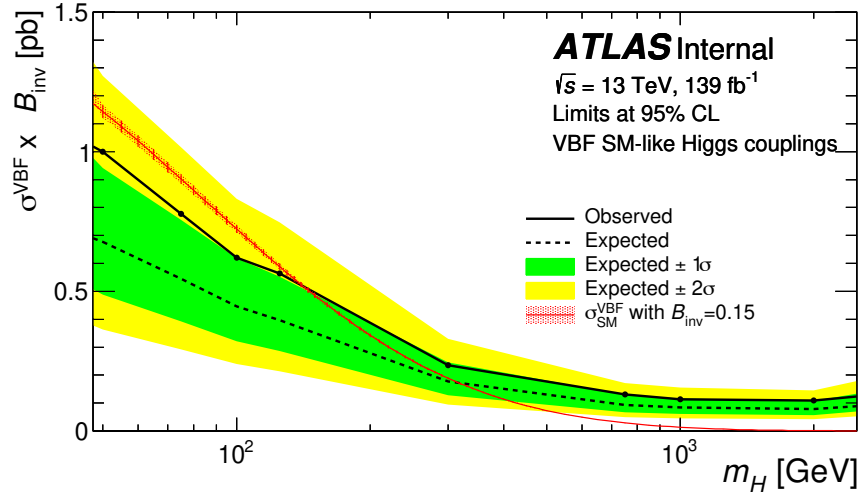


Figure 9.12: Plot showing limits on the production and invisible decay of a scalar mediator particle for a variety of mediator masses. The dashed line shows the expected limit and the solid black line the observed; black dots on the observed line show the points which were generated and interpolated to produce this plot. The red line shows the VBF cross section scaled by a branching ratio of 0.15, as was observed for the Standard Model Higgs boson, for comparison purposes. NLO electroweak corrections are not included in these signal models.

in the observed limit is present across the entire mass range.

9.6 Comparison to CMS Results

The preliminary results shown in Appendix C were released by ATLAS in March 2020. In October 2021, while a paper with the final results (and this thesis) were being prepared, the CMS collaboration released their first full run 2 limits on $\mathcal{B}_{H \rightarrow \text{inv.}}$ in the vector boson fusion channel. They set an observed (expected) 95% confidence level limit of 0.17 (0.11), with a 1.5σ excess [187]. Figure 9.13 shows how the expected CMS limit varies across the three data-taking periods, as well as the combination. The final ATLAS limit of 0.14 (0.10) is quite close in sensitivity, although CMS sees a slightly larger excess. Of course, like the Appendix C limit, this is a preliminary result, and so the final CMS limit may also slightly change change prior to publication.

For a brief comparison between the ATLAS and CMS results, Table 9.9 shows the groups of systematic and statistical uncertainties, ordered by their impact on the limit. It is interesting to see that Monte Carlo statistics are still a dominant uncertainty on the CMS result. CMS uses Madgraph to generate NLO V+Jets samples instead of Sherpa, the use of which was documented in Chapter 7.

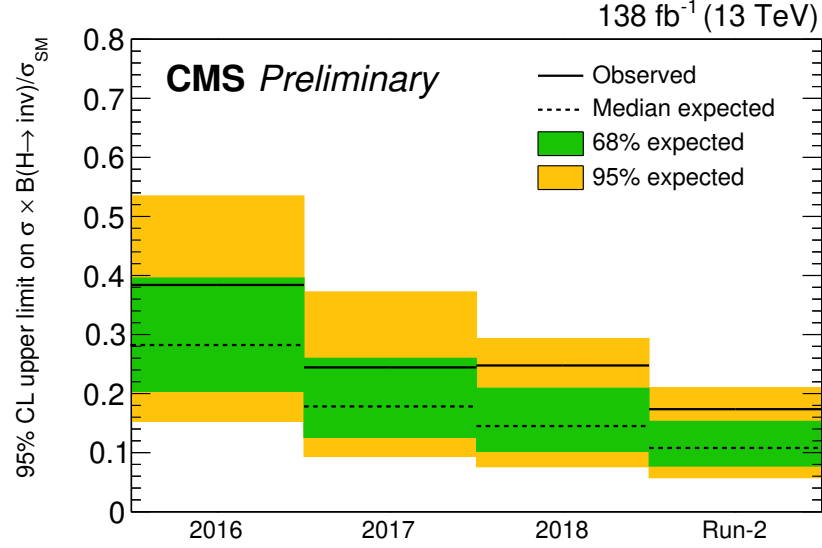


Figure 9.13: Observed and expected limits on the invisible Higgs branching ratio, $\mathcal{B}_{H \rightarrow \text{inv.}}$ in the vector boson fusion channel with the full run 2 CMS dataset. The combined full run 2 95% confidence level limit from CMS is 0.17 (0.11) [187].

Group of systematic uncertainties	Observed impact on $\mathcal{B}(H \rightarrow \text{inv})$	Expected impact on $\mathcal{B}(H \rightarrow \text{inv})$
Theory	+0.026 -0.025	± 0.024
MC stat.	+0.024 -0.023	+0.023 -0.024
Triggers	+0.021 -0.022	± 0.021
Leptons/photons/b	+0.012 -0.011	+0.010 -0.011
QCD multijet mismodeling	± 0.013	± 0.014
Jet calibration	+0.010 -0.007	± 0.007
Lumi/PU	± 0.005	+0.004 -0.005
Other systematic uncertainties	+0.013 -0.010	± 0.010
Stat.	± 0.029	± 0.030

Table 9.9: Groups of uncertainties on the observed and expected limit on $\mathcal{B}_{H \rightarrow \text{inv.}}$ from the CMS full run 2 VBF Higgs to invisible analysis [187].

On the other hand, multijet systematics do not appear to be as important as they are in the ATLAS analysis. More details on this result and how the CMS analysis differs from the one documented in this thesis can be found in the referenced CMS publication [187].

CHAPTER 10

Conclusion

This thesis presented two projects: work on the HCCStar ASIC for the ATLAS Inner Tracker upgrade project in Chapters 4 and 5, and a search for the invisible decay of the Higgs boson in Chapters 6 through 9. The HCCStar’s digital logic was tested and simulated for correctness and radiation sensitivity using the Python cocotb framework. A prototype HCCStar design, version 0, was fabricated in 2018 and tested, but found not to work in a high radiation environment. A second revision, version 1, was then developed and simulated before being submitted for fabrication in 2021 with improved protection against radiation-induced single event errors.

The search for Higgs to Invisible was performed with the full ATLAS run 2 dataset, with an integrated luminosity of 139fb^{-1} of proton-proton collisions with a center-of-mass energy of $\sqrt{s} = 13\text{TeV}$. No significant excess over the Standard Model prediction was seen. Observed (expected) upper limits were set on the invisible Higgs branching ratio at $\mathcal{B}_{\text{H} \rightarrow \text{inv.}} = 0.145$ (0.103) at a 95% confidence level. This result was then interpreted for several different dark matter models involving [WIMPs](#), and a limit set on the spin-independent WIMP nucleon cross section for comparison with dark matter direct detection experiments. Significant work was done to reduce the impact of uncertainties due to Monte Carlo background statistics seen in previous searches in this channel, and this is no longer one of the dominant sources of uncertainty on the limit.

As of this writing, the next run of the LHC, run 3, is just around the corner, with data-taking due to start in early 2022. Run 3 will last for several years, before Long Shutdown 3 begins, where the various high luminosity upgrades will be installed in preparation for the HL-LHC era. By the end of run 3, we will have collected an integrated luminosity of approximately 300fb^{-1} . Analyses like this one can therefore be repeated with approximately double the data statistics even while upgrades like the Inner Tracker are being finished up and installed. If nothing else is changed, doubling the

available data will only improve the expected limit from around 0.10 to 0.07, so it would be good to find ways to further reduce the various systematic uncertainties, especially the multijet and fake lepton estimates. And while Monte Carlo statistics appear to be under control at the moment, there may be future developments in this area that both improve the event generation performance and perhaps also the accuracy; some potential improvements and outstanding problems still to be solved were discussed in Chapter 7.

As for the HCCStar, chips are expected to return from the foundry by the end of the year, where they will then need to be carefully tested. Part of that testing will involve another irradiation campaign, where— if the simulation work described in Chapter 5 was done correctly!— the chips will hopefully work properly without discovering any new unexpected features. If another revision of the chip is required, it will likely involve moving from 130 nm to 65 nm transistors in order to fit more radiation protection in the available space, which would require another verification and simulation campaign. If the chips do work as intended, then eventually a production run will be ordered the assembly of ITk modules will begin, in preparation for installation once run 3 finishes. Then, once the Inner Tracker is installed and HL-LHC running underway, we expect to collect a total integrated luminosity of 3000 fb^{-1} over the course of the following decade. That should allow far more stringent limits on the invisible Higgs decay and other Beyond the Standard Model processes to be set, and with any luck might even lead to the discovery of new physics.

APPENDIX A

Cocotb: A Python-Based Digital Logic Framework

When designing a [field programmable gate array \(FPGA\)](#) or an [application specific integrated circuit \(ASIC\)](#), it is important to ensure that the digital logic— the source code that will be compiled into logic gates and circuit elements— actually does what it is meant to do. For [ASICs](#), which are not reprogrammable and can cost hundreds of thousands of dollars to fabricate, it is especially critical to ensure the logic is sufficiently tested and proven correct before submission. But even for [FPGAs](#), where the logic will be loaded onto the board as firmware and therefore *are* reprogrammable, having a robust set of tests to probe edge cases and catch regressions can help development, making it easier and less risky to make changes.

Chapters 4 and 5 describes the development of an [ASIC](#), the HCCStar, for the [ATLAS](#) tracker upgrade. Appendix B also describes the development of “spy buffer” firmware for use in a track processor [FPGA](#). The [cocotb](#) framework was used for digital logic verification in both projects [107]. Cocotb, short for “coroutine cosimulation testbench”, is an open source library that enables the verification to be done entirely using the [Python](#) programming language. It was originally created by two developers, Chris Higgs and Stuart Hodgson, but has since grown a large community of users and contributors. As of this writing, it is maintained by a group of about half a dozen volunteers, who merge contributions from the community and manage the release process, and is publicly available under the BSD license [107].

We found this Python-based approach to verification to be extremely powerful when using it on both the HCCStar and the track processor; see Chapter 5 and Appendix B for more details on those efforts. This appendix is intended to supplement that material and serve as a general overview of

cocotb for use as a verification tool¹²¹. It might be useful as a guide for a future student interested in or evaluating the use of cocotb for a new project.

Section A.1 describes the general philosophy taken by cocotb and contrasts it with some alternative approaches and tools. Section A.2 then demonstrates how cocotb works and steps through an example, with source code. Finally, Sections A.3 and A.4 explore the meaning of the name, outlining the cosimulation approach and use of coroutines, respectively.

A.1 Digital Logic and Verification

Digital logic is often developed using a [hardware description language \(HDL\)](#), a type of programming language specifically designed to describe hardware. The two most common are [Verilog](#) and [VHDL](#): the projects described in this thesis all use [Verilog](#), but the two are interoperable [208] [209]. These languages can describe logic at what's known as the [register-transfer level \(RTL\)](#), where the actual logic gates (AND, OR, NOT, etc.) are abstracted away. Instead, a circuit is represented as a series of binary-valued *registers*, operations on those registers, and the wires that connect those registers together.

The source code can then be compiled down into logic gates and other primitive circuit elements in a process known as synthesis. When code is synthesized, registers will often be mapped to [flip-flops \(FFs\)](#), a standard circuit element made of logic gates that can store a binary value. This process produces what's known as a *gate-level netlist*: a description of the circuit as a series of logic gates. From here, the gate-level implementation can be mapped and routed onto a physical layout, and either turned into the description of an [ASIC](#) or firmware for a [FPGA](#).

Software tools exist that can simulate both the register-transfer and gate level implementations of a circuit without needing to physically implement the design. These simulators contain [HDL](#) interpreters, as well as a number of other useful debugging tools. The built-in waveform viewer, for example, serves as a digital oscilloscope and allows the user to examine the state of any signal inside the circuit. The use of these tools to simulate a design and test it for correctness is often referred to as “verification”, and at its most basic level, this just involves proving that all possible inputs to a circuit produce the correct output. The software written to verify a design is often referred to as a digital “testbench”.

For simple circuits, this can be relatively straightforward. Suppose we wanted to prove the

¹²¹The material in this appendix is primarily derived from a similar seminar I gave at CERN [207], as well as a presentation at DPF 2019 [108], both on the use of cocotb for verification of the HCCStar

correctness of a fundamental circuit element like the XOR gate. An XOR gate takes two inputs, A and B, and performs the “exclusive OR”– the output is one if A or B are one, *unless* both are 1, in which case it is zero. We know that there are only four possible inputs, with four possible outputs, as summarized in the truth table below. Verifying the correctness of a XOR gate, then, would simply involve instantiating a XOR gate, and testing that if we step through all four input values, we get the correct output. This could be done entirely by writing a short program in Verilog, and looking at the output signal Z in the waveform viewer.

A	B	Z
0	0	0
1	0	1
0	1	1
1	1	0

Many designs, of course, are far more complex than this and require a more robust approach to verification. The HCCStar, for example, is responsible for merging a number of parallel output streams together and gracefully handling synchronization errors; it also supports a number of operating modes, controlled by a series of onboard configuration registers, and communicates in multiple serial protocols. Simply enumerating all possible states and stepping through them, relying on access to the waveforms to detect errors, is unlikely to be sufficient *or* efficient, given the vast number of states to cover and the complexity involved in understanding whether something is or isn’t an error.

A more sophisticated approach would involve automatically detecting errors in the behavior of a circuit. To do so would require developing a high-level model, which could evaluate the expected output in response to a given input stimulus. Then a simulation could compare the results of the model against the results of the actual circuit in order to determine the correctness of the design. In principle, this could be done using the same [HDLs](#) used for the design itself. However, as their name implies, these languages are intended to describe hardware and not necessarily to verify hardware. They are generally quite low-level, and are therefore not well suited for high-level behavioral descriptions of a circuit or algorithm. Two potential alternate approaches to advanced verification therefore suggest themselves. One would be to extend the hardware description languages with dedicated simulation-only features for verification. Another would be to a separate set of tools– and possibly separate programming languages– for describing and verifying a circuit design.

The first approach has been widely adopted in the digital design industry. The SystemVerilog language consists of a set of extensions to Verilog adding many simulation-only features [\[210\]](#). Many

of these features are drawn from higher level software engineering in order to enable the creation of more sophisticated testbenches and models for verification. An analogy, for readers familiar with software engineering, might be that SystemVerilog is the C++ to Verilog's C: SystemVerilog implements object-oriented programming, and its class hierarchy enables easier reuse of verification code, either across multiple projects or within a single project. Assertion statements make it easier to automatically detect errors and identify when logic enters a specific state without manual inspection. And higher-order data types, like strings and structs, make it easier to construct higher-level models [210].

Powerful verification frameworks have been built on top of these features. Perhaps the most commonly used in industry is [Universal Verification Methodology \(UVM\)](#); a set of over 300 libraries providing a number of classes and methods to build a testbench [211]. UVM provides standard implementations for objects like “drivers” and “monitors”—classes which can control an input or monitor an output— as well as scoreboards, which compare the recorded output from a monitor with the expected output from a model to determine correctness. UVM also contains classes to generate random sequences according to a given set of constraints in order to produce input stimulus for a design [211].

As the previous paragraphs may have suggested, this approach, while quite powerful, is also quite complex. UVM provides a number of useful tools, but with over 300 classes, there appears to be a fair amount of redundancy and multiple ways to architect a testbench. This complexity means that UVM has a very high learning curve; while there is documentation, it is somewhat sparse, and the apparently large amount of boilerplate code necessary to set up a UVM project makes it quite difficult to get started. In industry, it seems that long, multi-week training courses are necessary to come up to speed on UVM, and even then, it may take years to fully master this technology. And while UVM is only a single framework built on top of SystemVerilog, that language itself is complex and somewhat sparsely documented. Like Verilog itself, it's a somewhat obscure language outside the world of hardware design, which means available resources and documentation are limited relative to what might be expected from a more general tool.

Finally, it is worth remembering that a [HDL](#) is designed to describe hardware: most syntax in the original Verilog is synthesizable, meaning that it can be translated into logic gates and implemented on a real [FPGA](#) or [ASIC](#). Verification only code is not synthesizable; it is not possible to translate a string or a class into logic gates. And while digital verification frameworks are sometimes called “testbenches”, they are not actually a physical test setup: they are fundamentally software, not hardware. Using a hardware description language to build them might not be optimal.

All of this suggests an argument in favor for trying the second approach mentioned above: instead of adding features to the HDLs to make them better suited for building software, it might be better to use a general purpose programming language instead. If a testbench is software, then it makes sense to use a software language to develop that testbench, in much the same way that we generally use hardware languages to develop hardware. Writing a high-level model of a design in a high-level software language might be much simpler than trying to do so in a low-level hardware language, and many high-level software languages already have features like object oriented programming built into them. All that is needed to make this approach work would be to build an interface between the high-level verification software and the actual Verilog or VHDL implementation of the circuit being tested.

A.2 Introduction to Cocotb

The second approach discussed in the previous section is the philosophy taken by cocotb [107]. Cocotb's original developers, Chris Higgs and Stuart Hodgson, chose the Python programming language [212], which is widely used in many fields, including scientific computing. They wrote a library providing an interface from Python code to the HDL simulator, which allows the verification testbench to be written in Python.

Python is a popular and well documented language with a large standard library and an even larger ecosystem of other libraries, such as the scientific computing packages numpy and scipy [109] [213]. For our purposes, Python has another advantage, in that it is widely used inside the ATLAS experiment, and has bindings to the ROOT computing framework [75]. Physicists, or physics students, who may be asked to contribute to the verification of a chip or firmware project, are likely to already know Python or be asked to learn it at some point; while they may know something of Verilog, they're much less likely to have substantial experience with SystemVerilog (and even less likely to have experience with UVM). It is likely to be much easier for physicists to come up to speed on a project and contribute if the verification is being done with Python. That was certainly my personal experience working on projects with cocotb.

An initial version of cocotb was first released under the open-source BSD license in 2013. Since then, the community of users and developers has grown, and the project has matured. In 2018, due to concerns about the availability of the original authors, cocotb became an independent project under the Free and Open Source Silicon Foundation, a non-profit open hardware organization [214]. To ensure that the project remained active, several contributors from the community took on new

leadership roles in the project, including Philip Wagner, Kaleb Barrett, Eric Wieser, and Tomasz Hemperek¹²². Since 2018, there have been five major releases of the project, and the upstream community remains highly active.

This section walks through a simple example to demonstrate how cocotb works and how Python code can interact with the RTL design. Running any testbench with cocotb, including this example, requires a HDL simulator, just like a more traditional approach to verification would. Simulators are produced by a number of major digital design companies, such as Cadence, Mentor Graphics, and Xilinx. In addition, there are a handful of open source simulators, such as Icarus Verilog and GHDL. Most, but not all (Xilinx is a noticeable exception) of these simulators support the Verilog Procedural Interface (VPI) and VHDL Procedural Interface (VHPI) standards. These protocols, part of the Verilog IEEE standard, are designed to allow external programs to control and interact with a simulator (and vice versa, to allow Verilog or VHDL code to communicate with external programs) [208]. Cocotb implements these protocols and can interact with any third-party simulator that supports them; it does not contain its own custom simulator.

As of this writing, cocotb can be used with the simulators listed below [107]:

- Icarus Verilog (open-source)
- GHDL (open-source)
- Verilator (open-source)
- Tachyon DA CVC (open source)
- Synopsys VCD (commercial)
- Aldec Riviera-PRO and Active-HDL (commercial)
- Mentor/Siemens EDA Questa and ModelSim (commercial)
- Cadence Incisive and Xcelium (commercial)

The projects described in this thesis were simulated using the Cadence Incisive/Xcelium and Mentor Graphics Questa products, and we found cocotb to work quite well with both. While cocotb does not support the simulator used by Xilinx, cocotb can still be used to test firmware for a Xilinx FPGA, as it is possible to load the Xilinx libraries into another simulator, such as Questa.

¹²²A full accounting of contributors can be found online on the project's repository: <https://github.com/cocotb/cocotb/graphs/contributors>. Note that Tomasz Hemperek is a fellow member of the ATLAS collaboration, an engineer at the University of Bonn.

A.2.1 Example Design

To demonstrate how cocotb works using this [VPI](#) (or [VHPI](#)) interface, source code for a simple example is included as Source Code [A.1](#). While this example is a simple block only slightly more complex than the XOR gate example shown in the previous section, it is a real piece of code from the HCCStar [ASIC](#) discussed in Chapter [4](#). Verification of this block was used as an exercise in order to become more familiar with cocotb when we first considered using it for the HCCStar project.

```
// example_mux.v
module example_mux(
    output wire we_lp_muxed_o,
    input wire readout_mode_i,
    input wire L0_i,
    input wire we_lp_i
);
    // Switch between inputs depending on value of readout mode.
    assign we_lp_muxed_o = readout_mode_i ? L0_i : we_lp_i;
endmodule
```

Source Code A.1: Verilog implementation of a simple multiplexer.

Source Code [A.1](#) is a multiplexer: a type of circuit which takes multiple inputs, and can switch (or “multiplex”) between them. In this case, our example MUX switches between `L0_i` and `we_lp_i`, connecting one or the other to the output `we_lp_muxed_o` depending on the value of `readout_mode_i`¹²³.

A.2.2 Example Test Code

A simple testbench for the multiplexer, then, would involve confirming that the output is properly switched between the two inputs. Test code could toggle the two inputs with `readout_mode_i` set to different values. Source Code [A.2](#) shows an example Python testbench that does just that.

There are a few key pieces of the example code that should be highlighted. After importing the cocotb libraries, the first thing we do is to create a Python function which is annotated by `@cocotb.test()`. This is a Python *decorator*, and its presence here tells the cocotb scheduler that this function `mux_test` should be ran when the simulation starts. A testbench can have multiple tests, in which case they will be ran sequentially.

¹²³In the HCCStar ASIC, this block helps switch between the single-level and multi-level triggering modes described in Chapter [4](#). Depending on which mode the chip is in, either L0 or LP commands will be used to dispatch readout commands, and so a multiplexer is needed to switch between those two input streams. If you are interested in more details on this functionality, consult Chapter [4](#).


```

# mux_tester.py
import cocotb
from cocotb import triggers, result

@cocotb.test()
def mux_test(dut):

    # Set initial values at start of simulation.
    dut.L0_i <= 0
    dut.we_lp_i <= 0
    dut.readout_mode_i <= 1

    # Change L0_i to 1, wait 1 ns, and make sure output updates.
    dut.L0_i <= 1
    yield triggers.Timer(1, "ns")
    if dut.we_lp_muxed_o != 1:
        raise result.TestFailure("Failure!")

    # Change readout mode, and make sure output no longer points at L0_i.
    dut.readout_mode_i <= 0
    yield triggers.Timer(1, "ns")
    if dut.we_lp_muxed_o != 0:
        raise result.TestFailure("Failure!")

```

Source Code A.2: Simple Python testbench using cocotb for Source Code A.1.

The test function takes a single argument, `dut`. This variable, represents the [design under test \(DUT\)](#), which in this case is the multiplexer shown in Source Code A.1, as a Python object. Wires and registers inside the DUT can be accessed here as Python objects, so the input wire `L0_i` is available from the Python as `dut.L0_i`, and so on. If the DUT instantiated any submodules, those would be accessible here as well. Python code can read or change any value anywhere in the simulated design hierarchy [107].

The test proceeds to write several values into the input variables. When it does so, it uses the operator `<=`. In Verilog, this would be a “non-blocking assignment”¹²⁴, but in normal Python, this would not be valid syntax. However, cocotb uses this as shorthand for `dut.L0_i.value = 0`; we could instead write the test using this more verbose syntax instead, and it would be fully equivalent.

Finally, after setting some values, we yield control from the Python testbench to the Verilog simulator with the line `yield triggers.Timer(1, "ns")`. In cocotb, the Python testbench and

¹²⁴Verilog is not an imperative, sequential language, and the order at which statements are executed can be non-deterministic. A series of statements using the non-blocking assignment operator will happen *at the same time*, while a series of statements using the blocking assignment operator (`=`) will occur sequentially [208]. Confusing or mixing the two can lead to issues.

the design are simulated independently, and the Python must explicitly pass control back to the simulator to allow simulation time to advance and for changes to propagate. This is done by yielding a trigger, and here we use the Timer trigger to wait a short amount of time (1 nanosecond). A more detailed discussion of the use of triggers can be found in Section A.3.

Then, after waiting one nanosecond for simulation time to advance, we check to see if the output value is as expected. If it is not, we fail the simulation, causing the test to exit with an error. Otherwise, we change the readout mode input, wait another nanosecond for this to propagate, and then check the output again to ensure that it has changed as well.

A.2.3 Makefile Configuration

With the Python test code written, we need one additional piece in order to be able to run our testbench: a Makefile. cocotb uses the common automation tool `make` in order to configure, build, and run a simulation¹²⁵. A Makefile is written in a declarative programming language and consists of a series of instructions, often involving the compilation of code, that will be executed when running `make` in a directory.

A cocotb Makefile can be quite simple; all that we need to do is set a few variables, and include the default cocotb Makefile, which knows how to actually run the given simulator.

```
SIM ?= ius # Cadence Incisive simulator.
MODULE = mux_tester
TOPLEVEL = example_mux
TOPLEVEL_LANG = verilog
VERILOG_SOURCES = ../rtl/example_mux.v
include $(shell cocotb-config --makefiles)/Makefile.sim
```

Source Code A.3: Makefile configuration to run testbench shown in Source Code A.2.

First, we tell cocotb that the default simulator should be Cadence Incisive by setting `SIM`. (If not set, cocotb will default to using the open-source Icarus Verilog simulator). Next, we tell cocotb that `mux_tester.py` (Source Code A.2) is the main Python module to load by setting the appropriately-named variable `MODULE`. Then, we need to tell cocotb that `example_mux.v` (Source Code A.1) is the top-level design file to load, and so set `TOPLEVEL` and `TOPLEVEL_LANG` accordingly. (We could also set the latter to VHDL, if we were simulating a VHDL design). And finally, we need to include a list of the HDL source files to load, by passing `VERILOG_SOURCES` and pointing it at our example MUX

¹²⁵There is interest in the cocotb community in developing non-Makefile ways of configuring and running testbenches and perhaps eventually deprecating support for Makefiles. As of this writing, however, Makefiles are still in use.

source code. (Again, the appropriately named `VHDL_SOURCES` can be set to include source code in VHDL).

A cocotb Makefile can of course be much more complex than this if necessary; we could add conditionals, support for running different types of simulations, different targets, and so on. The Makefile could potentially be generated by an external tool, or load a configuration file, to enable more complex workflows. There are also a number of options we can set that the cocotb Makefile will respect. Two worth mentioning here are `EXTRA_ARGS`, which enables the passing of additional command-line options to the simulation, and `GUI`, which if set to 1 will attempt to load the simulator’s graphical interface (and enable the user to use the waveform viewer to interactively monitor the simulation). Details about other options can be found in the official cocotb documentation [107].

As this example hopefully demonstrates, cocotb is relatively straightforward to get started with. However, it is also possible to use cocotb to create more complex testbenches for more complex blocks. The next two sections explore two key cocotb concepts in more detail by attempting to unpack the project’s name: cosimulation and coroutines.

A.3 Cosimulation

The workflow outlined in the previous section, in which the design and testbench are simulated independently, is known as *cosimulation*. The Python test code runs on the Python interpreter, and the design under test runs on a HDL simulator, and the Python code must explicitly yield control back to the simulator in order for simulation time to advance. The testbench then can set triggers, which fire when a specific event in the simulation occurs. These triggers interrupt the simulation and cause the Python code to resume control.

The example presented in Section A.2 shows the use of the Timer trigger, where the testbench waits a fixed amount of simulation time. A number of other triggers are available in the cocotb library, allowing the creation of more complex logic. The RisingEdge and FallingEdge triggers, for example, enables code to block until a given digital signal transitions either from 0 to 1, or from 1 to 0. When testing clocked logic, these triggers allow Python code to interrupt on every cycle of the clock in order to perform some task, such as writing to a serial port or reading from one [107].

Because of the cosimulation approach used here, the performance of the simulation depends almost entirely on how often the Python code interrupts the simulator by yielding a trigger. The more often the simulation gets interrupted, and the more time spent executing Python code, the slower it will take for simulation time to advance. Cocotb offers a number of tools in order to

diagnose performance issues, most notably a simple report of the ratio of real time to simulation time at the end of every run. As an example, during a full run of the standalone HCCStar testbench described in Section 5.1, 3566.86 ns of simulation time passes on average for every second of real time.

One avenue for potential optimization is to move (or keep) part of the testbench in RTL, rather than attempt to have everything in Python. An easy way to do this would be to create a Verilog “wrapper” that sits around the actual design under test, and tell cocotb that this wrapper is the top-level block to instantiate. Then, the wrapper would simply load any RTL testbench components alongside the design. While it is not currently possible to call a Verilog task or procedure from the Python code directly, the Python testbench can simply interact with the Verilog testbench components through the VPI interface, in exactly the same way one would interact with the design.

For example, consider a design which implements a serial communications protocol. A driver for that protocol could be written in Python, which triggers on the rising edge of every clock and drives the serial input. But, if it is otherwise unnecessary to perform some task on every clock cycle, the driver could instead be written in Verilog. If the Python code wants to transmit a word over the serial interface, it simply loads it into a memory buffer through VPI access, and then the Verilog driver would read from that buffer, serializing and transmitting the word.

Having some part of the verification implemented in RTL instead of Python is potentially beneficial for other reasons, as well. If testbench code has already been written for a project, or for a component of a project, it does not need to be rewritten in Python in order for cocotb to be used. Or, as another example, the wrapper could be used to load a SystemVerilog assertion file, combining the SystemVerilog-based and Python-based verification approaches together.

A.4 Coroutines

The other key component of cocotb, and the other part of the project’s name, are *coroutines*, which are a type of *subroutine* [215]. In both software and firmware engineering, it is common to organize code into a series of reusable subroutines (also referred to as procedures, methods, tasks, or functions), which can be invoked from the “main” program or routine. Dividing code into subroutines allows specialized blocks of code to be reused, both in a single project and as libraries reused between projects. The Python standard library’s `abs(x)` function, for example, is a subroutine that returns the absolute value of a number x , so there is no need for a programmer to implement this themselves in their own code.

Coroutines are a generalized version of this concept for a cooperative multitasking paradigm [215]. When a Python program invokes the absolute value function, the “main” routine halts, execution of `abs(x)` begins, and the “main” routine does not resume execution until the subroutine finishes and returns a value (in this case, $|x|$). There is no communication between the subroutine and the “main” program until the subroutine finishes. If this function is invoked a second time to find the absolute value of y , it will again run from the beginning, and again not communicate with the “main” program until it finishes, and returns $|y|$.

Dividing a program into a “main” routine and a series of subroutines is inherently an asymmetrical way to organize code. In contrast to this, the coroutine approach is symmetrical, and in principle there *is* no “main” program. Instead, a pair of coroutines can communicate with each other at any point during their execution. One can yield control to the other at a certain point, which can then run for a brief period, and then yield control back to the first. If the first coroutine then yields control to the second again, the second coroutine will not restart from the beginning, like a subroutine, but instead resume from where it had previously stopped.

In computing, this concept dates back to a paper from Melvin Conway, in 1963, on the design of a compiler for the COBOL language [216], and is later discussed in detail in the first volume of Donald Knuth’s famous book, the Art of Computer Programming [215]. Knuth presents the following analogy for this concept: consider two programs that play chess. If these programs are implemented as coroutines, they could be made to play each other: the first program makes a move, and then yields control to the second program to make its move, which then does so and yields control back to the first [215].

In this context, it should be apparent that this cooperative multitasking approach is fundamentally how cocotb’s cosimulation approach works. As discussed in the previous sections, the simulator is paused while Python code is executing, and only resumes once the Python explicitly yields control back to it. Both the simulator and the Python code, then, can be thought of as coroutines. But there is nothing stopping the Python code from being broken into more than one coroutine, and doing so enables the creation of increasingly complex testbenches.

Coroutines can be created using cocotb from ordinary Python functions and yielded in place of a trigger. A function can be marked as a coroutine if it is decorated with the `@cocotb.coroutine` decorator¹²⁶ as long as it yields a trigger or another coroutine. Source Code A.4 shows the creation

¹²⁶The syntax described here was originally used in cocotb for the creation of coroutines, but has since been deprecated. The `asyncio` library was added to the Python standard library in Python 3.3, and contains an implementation of coroutines [217]. Cocotb 1.4, released last year, dropped support for older Python releases, supports (and defaults to) the use of `asyncio` coroutines, and has deprecated the older syntax. The HCCStar verification described

```
import cocotb
from cocotb import triggers

@cocotb.coroutine
def test_helper(dut):
    # Example coroutine that performs a task and yields a trigger.
    dut.member <= 1
    yield triggers.RisingEdge(dut.clk)

@cocotb.test()
def test(dut):
    yield test_helper(dut)
```

Source Code A.4: Example creation and yielding of a coroutine in cocotb.

and use of an example coroutine; note that a cocotb test is itself just a special type of coroutine that is automatically launched by the scheduler at the start of simulation.

Dividing a testbench into multiple coroutines enables the creation of more complex testbenches and the reuse of code. Of course, this can also be achieved by the use of ordinary Python functions, as a testbench can call ordinary functions in addition to yielding other coroutines. The key distinction is that only coroutines can yield another coroutine or trigger: ordinary Python functions cannot. From a hardware designer's perspective, an analogy with the Verilog language's own subroutine syntax suggests itself. In Verilog, there are two types of subroutines: a task, which can consume simulation time, and a function, which cannot. Since only coroutines can (in fact, they must) yield another coroutine or trigger, coroutines can consume simulation time like a Verilog task. While ordinary Python functions can be called from a coroutine, they cannot yield a trigger themselves, and so like a Verilog function cannot consume simulation time.

The example shown in Source Code A.4 is quite simple, but far more sophisticated coroutines can be created. One useful feature in cocotb is the ability for the scheduler to fork a coroutine, causing it to run in parallel. Source Code A.5 shows an example of this syntax. In this example, we create a coroutine which repeatedly waits for the rising edge of a clock and then performs some task, such as driving a signal or monitoring an output. Since this coroutine will run continually, it is forked, instead of yielded, causing it to run in the background. From a Verilog perspective, this can be thought of as the creation of an always block—a block of code which will run repeatedly when something happens.

in this thesis was primarily done using older cocotb releases, and so the older syntax is presented here. Unless backwards compatibility is needed, new projects should likely use the newer syntax, as recommended in the official documentation.

```
import cocotb
from cocotb import Triggers

@cocotb.coroutine
def always_block(dut):
    while True:
        yield triggers.RisingEdge(dut.clk)
        # Do something on the rising edge of every clock.

@cocotb.test()
def test(dut):
    thread = cocotb.fork(always_block(dut))
```

Source Code A.5: Example creation of a forked coroutine.

It is possible to communicate between forked coroutines, just as it is possible to communicate between parallel threads or processes when developing software. Cocotb provides Python-only triggers, which can be shared between coroutines to allow synchronization between them. Yielding the Python “event” trigger, for instance, will pause execution of one coroutine until the event is explicitly set from another coroutine. Information can also be passed between forked coroutines through the use of Python data structures like a queue that supports thread-safe access¹²⁷. One coroutine could insert entries into a queue, and another forked coroutine could retrieve entries from the queue as they became available.

In this way, the combination of forked coroutines can be used to build up sophisticated testbench objects. For instance, a driver for an input serial protocol and a monitor for an output serial protocol could both be implemented as coroutines. These coroutines could be forked from a test coroutine, and each given access to a queue. The test coroutine would put entries into the transmission queue, which the driver would retrieve and transmit to the design. The monitor would then receive output messages from the design and add them to another queue. The test coroutine could then retrieve messages from this monitor queue and attempt to evaluate the correctness of the design’s response. This general approach was used to implement the serial drivers and monitors for the HCCStar, as discussed in Chapter 5.

¹²⁷A queue is a data structure where the first entry inserted into the queue will be the first entry read out of the queue; it is also known as a “FIFO”, especially in hardware design. For instance, see Chapter 4 for uses of FIFOs in the design of the ITk ASICs.

APPENDIX B

Hardware Tracking for the Trigger and the Spy Buffer

This appendix covers two topics: the proposed ATLAS [Hardware Tracking for the Trigger \(HTT\)](#) upgrade project, and the development of the spy buffer, a firmware block which was originally intended for use in the that project. HTT was intended to be part of the ATLAS [Trigger and Data Acquisition \(TDAQ\)](#) upgrade program for the High Luminosity LHC, which is described briefly in Chapter [4.1.3](#) [100]. The project would have involved implementing charged particle track-finding algorithms (described in Chapter [3](#)) in [field programmable gate array \(FPGA\)](#) firmware, so that tracking information could be used as part of the initial ATLAS trigger decision when deciding whether or not to read out an event.

Ultimately, the ATLAS collaboration decided not to proceed with a hardware track trigger, and so the HTT project as described here will not be implemented for run 4. This appendix serves to document how HTT would have functioned and how it would interact with the front-end readout electronics for the Inner Tracker described in Chapter [4](#). If HTT were to have been built, it would have consisted of a large cluster of FPGA boards responsible for receiving and unpacking data from the front-end, turning recorded hits into clusters, and running track-finding algorithms over those clusters to produce tracks. The main board for each HTT unit, the [Track Processor \(TP\)](#), would contain a number of different firmware blocks responsible for each stage of the process. The track-finding step itself would have been performed by a pair of on-board “mezzanine” FPGA cards, mounted on each TP.

Many of these firmware blocks were being developed by authors from different institutions, all collaborating together on the project. Agreeing on a common interface between these blocks was a

key early step in ensuring the different blocks would be able to communicate with each other. Part of this common interface involves ensuring proper flow control: that block A cannot send block B data it is not ready to receive. In addition, when debugging firmware, it is very useful to be able to “spy” on the data as it passes through the FPGA. Appendix A talks about the use of various monitoring tools to do this when simulating the firmware on a computer, but it is much harder to debug once the firmware has been flashed onto a physical board. Since a common interface between block boundaries was already something we planned to implement, it made sense to make this common interface responsible for “spying” on data passing through blocks in addition to enabling flow control. To help get started on the development of the TP firmware, I implemented a prototype of this spy buffer firmware block in the Verilog language [208].

This appendix covers that work and is broken down into three main sections. First, Section B.1 describes some of the history of hardware-based tracking in ATLAS, while Section B.2 describes the original architecture of the HTT proposal (which will not be implemented). Finally, Section B.3 describes the spy buffer itself, the data-flow protocol it is intended to support, and the monitoring functionality in detail.

B.1 Hardware-Based Tracking in ATLAS

The charged particle tracker is an important component of the ATLAS detector, allowing us to measure the trajectories of charged particles produced from collisions. The current tracking system, known as the Inner Detector, was discussed in Section 3.2.1. This tracker will be replaced as part of ATLAS’s high luminosity upgrade program, and the replacement Inner Tracker was discussed in Chapters 4 and 5 in considerable detail. Part of this upgrade program also involves a new TDAQ system to decide whether a given proton-proton bunch crossing should be read out of the detector: as Section 3.3 describes, because bunch crossings occur every 25 ns, it is impossible to fully record and analyze every event, and so a triggering system is needed.

The current ATLAS trigger system was described in Section 3.3, and the proposed upgrade for the HL-LHC in Section 4.1.3. In both cases, the system will involve a two-stage trigger decision: a fast, hardware-based initial decision¹²⁸ which decides whether or not an event should be read out, and then a slower partially-software-based trigger¹²⁹ which decides whether or not the event should be kept. In both the present tracker and the proposed “baseline” HL-LHC upgrade, information

¹²⁸The Level 1 trigger in the current system, and the Level 0 trigger in the baseline upgrade scenario.

¹²⁹The High Level Trigger (HLT) in the current system, and the Event Filter in the baseline upgrade scenario.

from the Inner Detector (or ITk in the HL-LHC) is not available when making the initial decision and only used as part of the second stage decision. Enabling the use of tracking information in the first stage requires either implementing tracking algorithms in hardware or firmware, or finding a way to do software based tracking at a significantly higher rate.

There have been several proposals to do hardware based tracking in ATLAS. A project known as the [Fast Tracker \(FTK\)](#) was originally planned to provide hardware-based tracking for run 2 (2015-2018) [218] [72]. (FTk was being tested during part of run 2; it can be seen in some diagrams describing the current trigger system, such as Figure 3.12). Unfortunately, the project ran into significant technical difficulties, and was originally postponed to run 3 (2021-2024) before being cancelled altogether. In addition to the aforementioned technical issues, improvements to the software-based tracking used in the [HLT](#) weakened the physics case for proceeding with the project [72].

A second project, [HTT](#), was originally based off of the FTK design and was proposed as part of the HL-LHC upgrade for run 4 and beyond (2026+). In the baseline upgrade scenario, with a 1 MHz Level 0 trigger rate and a 100 KHz Event Filter rate, HTT would have served as a co-processor for the Event Filter. HTT would have also supported an “evolved” upgrade scenario, however, where the Level 0 trigger would be split into a 4 MHz Level 0 decision and a 400 KHz Level 1 decision, where tracking information would be used. For every bunch crossing accepted by Level 0 decision, up to 10% of the Inner Tracker would be read out and sent to HTT, where regional tracking would be used to make a Level 1 decision. Then, the full ITK volume would be read out for accepted events. This evolved scenario has also been referred to as “L1Track” [100].

However, as mentioned in Section 4.1.3, the decision was ultimately taken not to proceed with the evolved trigger scenario either. This decision was driven by a combination of technical problems with the front-end ITk electronics, including difficulty adding regional readout capability to the pixel detector, as well as issues adding enough radiation protection to the strip detector’s [application specific integrated circuits \(ASICs\)](#) (described in Section 5.3) as well as concern that HTT would run into similar issues as FTK before it. In addition, additional improvements to software-based tracking further weakened the case for building a fully custom hardware-based system. That has led to the decision not to build the HTT project described in the next section. As of this writing, the ATLAS collaboration is still studying several alternative schemes for the HL-LHC upgrade, including alternate solutions that still make some use of FPGAs to accelerate track-finding.

B.2 Hardware Tracking for the Trigger

The [HTT](#) project was designed to support both the baseline and evolved track trigger scenarios. If it were to be built, the HTT would consist of a cluster of FPGA boards, and will receive hit data from the ATLAS Inner Tracker and run hardware implementations of track-finding algorithms over that data [100]. In the baseline configuration, the HTT serves as an efficient co-processor for the trigger system and finds tracks across the entirety of the ITk volume. In the evolved scenario, the HTT will still be used for this purpose. However, it will also be used to perform fast track-finding at a 1 MHz rate across certain regions of interest that represent up to 10% of the detector volume at any given time. These two types of requests are respectively referred to as [Global HTT \(gHTT\)](#) and [Regional HTT \(rHTT\)](#) configurations, since they involve processing data from all or part of the detector. The same hardware and firmware could be used for both types of requests.

The HTT system itself is based on the design used in the FTK project [218] [100]. A HTT unit would consist of five main hardware elements: the [HTT Interface \(HTT-IF\)](#), two types of [TP](#) boards (the [Associative Memory TP \(AMTP\)](#) and [Second-Stage TP \(SSTP\)](#) boards), and the two types of mezzanine cards (the [Pattern Recognition Mezzanine \(PRM\)](#) and [Track Fitter Mezzanine \(TFM\)](#)) mounted on those boards. Figure B.1 shows the connections between these components and how data will flow through the system in both the [rHTT](#) and [gHTT](#) configurations [100].

The [HTT-IF](#) provides a network connection, over which hit data from the tracker is transmitted to the track processors. The two types of track processor board use the same hardware, and are both capable of receiving hits, unpacking them and decoding the data formats used by the ITk front end, and clustering hits if necessary¹³⁰. The hit data is then either transmitted to another board as appropriate for processing, or to a mezzanine card, where the actual track-finding algorithm is performed [100].

The main difference between the two [TP](#) boards is which of the two mezzanine cards is installed. The first stage of track-finding is performed on the [AMTP](#) using the [PRM](#) card, and is done by pattern recognition. A large library of patterns corresponding to tracks can be loaded onto the PRM, and these patterns are matched to a sequence of hits to determine whether or not those hits represent a track. This pattern matching algorithm is based on a similar approach attempted for the [FTK](#) project [218]. A set of twelve ASICs, known as Associative Memory chips, are installed on each PRM card and perform the pattern matching in parallel [100].

¹³⁰As described in Section 4.3.2, the front-end electronics in the ITk Strip detector already apply a clustering algorithm to hit data. The TP therefore does not need to do much beyond unpack these strip clusters. Processing data from the ITk Pixel front-end is a more involved process, and more complex firmware is needed [95].

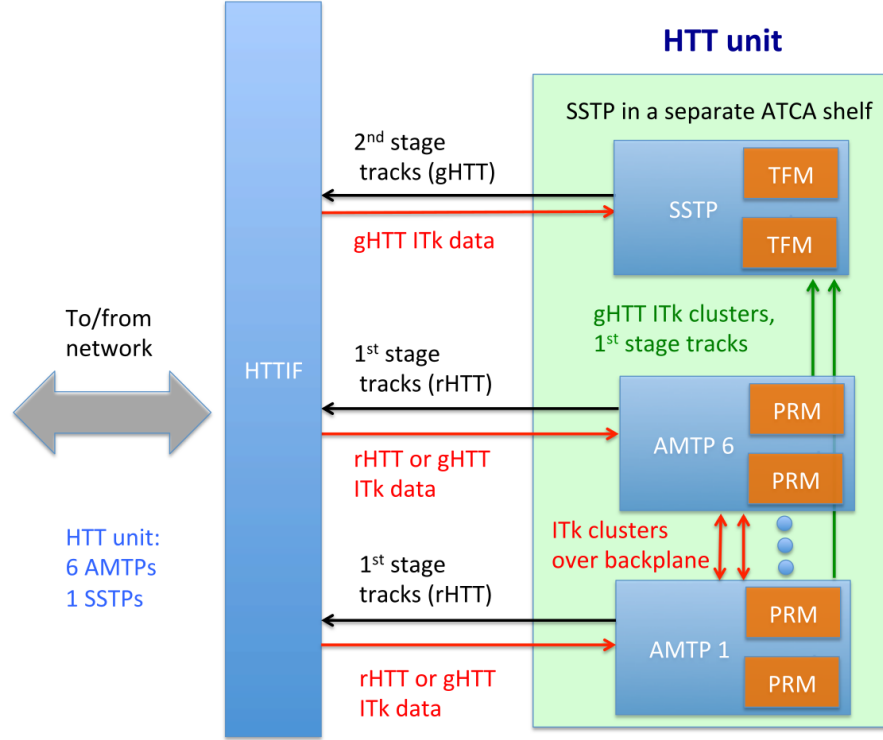


Figure B.1: Schematic diagram showing the architecture of an HTT unit, taken from the TDAQ technical design report [100]. Connections and data flow between the external interface and the **AMTP** and **SSTP** boards are shown for both a **rHTT** and **gHTT** request. Note that a single “TP” here consists of two FPGAs, so an actual unit would consist of two SSTP boards and twelve AMTP boards.

When handling a **rHTT** request, as shown in Figure B.1, the first stage patterns are all that the system produces. On the other hand, when handling a **gHTT** request, a second stage of tracking is performed by the **SSTP**. This board contains the other mezzanine card, the **TFM**, which takes the patterns generated by the **AMTPs** as input and runs a track-finding algorithm over them. The tracks produced will then be sent out to the rest of the trigger system through the network interface [100]. The need to support the sharing of hits and tracks between the different boards in these different scenarios makes the TP firmware quite complex, with many different data processing stages. The spy buffer described in the rest of this appendix was originally designed to help provide visibility into the passage of data through HTT and aid in developing the firmware.

B.3 Spy Buffer

To support the development of the TP board, a common interface between stages of the firmware was designed. It was desired that such a common interface would both support a standard flow control mechanism and debugging capability. A flow control mechanism enables a block or stage of the firmware to signal whether it is or is not ready to receive data from the previous stage. In addition, support for monitoring data passing through the flow control mechanism and transmitting it for offline analysis would help debug problems, both during development and potentially operations. This desire for a common interface led to the creation of the spy buffer block, which could perform both tasks.

When developing and simulating firmware¹³¹, it is possible for a developer to visually inspect the current state of every variable in the code. This is obviously not possible once that firmware has been flashed to a real FPGA. In principle, this is fine if the verification and simulation was sufficient. But it is always possible that new and unexpected problems will arise after flashing the firmware, especially when combining blocks of firmware created by different developers. Having a debugging mechanism that can report the inputs and output of each block is therefore useful to understand and resolve any problems after flashing.

Initially, the spy buffer was just intended for use in the TP project. Over time, other developers became interested in using the spy buffer on other projects within ATLAS. Therefore, an effort has been made to make the firmware as generic and as configurable as possible. For basic use of the spy buffer, minimal assumptions are made about the way in which data is formatted. It is assumed that data will be transmitted in chunks or words of some fixed size, which can be set using a compile-time parameter for each instance of the spy buffer. The most-significant-bit of this chunk is interpreted as a “metadata” bit, and can be used to indicate that one or more of these words is a metadata word. In the TP, it is intended that “events” will be sent from block to block of variable size. These variable-size events will then be broken down into a number of fixed-size chunks, and begin and end with a specific metadata word¹³². While the firmware is capable of parsing event boundaries (see Section B.3.3), it is not necessary to use this type of framing in order to use the block.

¹³¹Perhaps using the type of verification tool described in Appendix A.

¹³²The exact meaning of “event” in the TP may vary depending on which blocks are communicating with each other; it is possible a given message may not correspond directly to a single event.

B.3.1 Flow Control

The spy buffer provides flow control by means of a [First-In-First-Out \(FIFO\)](#). A FIFO is a type of data structure, also known as a queue. In a FIFO, entries are read out in the order that they were inserted with the oldest entry read out first. The controls to write and read from the FIFO provide a common interface to transfer data between blocks. The block writing into the FIFO can be referred to as the “write-side” block, and the block reading can be called the “read-side” block.

When the write-side block wishes to send data to the read-side block, it must first check to see if the flow control FIFO is almost full. The size of this FIFO, and the exact meaning of “almost” full are both adjustable, but by default the FIFO is almost full if there is only one entry left. The write-side block should wait until the FIFO becomes not almost full, at which point it can begin transmitting pending words by setting a write enable signal high. On the other side of the spy buffer, the read-side block should check to see if the FIFO is empty. As long as it is not empty, it can assert a read enable signal and copy a new word out of the FIFO on each clock cycle.

The initial version of the spy buffer uses an asynchronous, dual-clock FIFO based on a design by Clifford Cummings [219]. “Dual clock” in this context means that the FIFO can be written into and read from at different frequencies. This means that the firmware blocks on either side of the spy buffer can run at different speeds, or in different *clock domains*. The FIFO is then said to provide a *clock domain crossing*, where logic running at two different clock speeds can interact [220]. Building support for clock domain crossings into the standard interface between blocks means that, in principle, each block could run off of a separate clock with minimal extra overhead. Allowing different blocks to run at slower or faster clocks as necessary prevents a single slow block from slowing down the rest of the firmware.

It is not necessary to use the flow control mechanism to use the spy buffer. A compile-time parameter can be set when instantiating a spy buffer which prevents the FIFO from being created. In this mode, known as “passthrough” mode, the data input is connected directly to the data output with no flow control and no support for clock domain crossings. Passthrough mode is intended for cases where a firmware developer does not want any latency from flow control, either because they know that none is needed or because they have implemented their own elsewhere. In this mode, the spy buffer can just be used to monitor activity on a single wire.

B.3.2 Monitoring and Readout

The other main feature provided by the spy buffer is the ability to spy. Along with the flow control FIFO, the spy buffer contains a large circular buffer (the titular “spy buffer”, also referred to as a “spy memory”). During normal operations, each word transmitted through the flow control FIFO is also copied into the circular buffer¹³³. “Circular” simply means that once the buffer is full, the next word to be written in will overwrite the oldest entry. Then the next word will overwrite what was then the second-oldest entry, and so on. The size of this spy memory is a compile-time parameter that can be set separately for each spy buffer.

Reading out this memory can be done using a dedicated “spy interface”, which runs on its own “spy clock”. The spy clock domain is separate from the two FIFO clock domains, so that monitoring and data flow can occur at different speeds. This interface could be connected to controller firmware on the FPGA, or hooked up directly to an external interface to the outside world. For the TP, we have explored using the CERN IPbus protocol [221] to interact with the spy buffers and read out events¹³⁴.

To interact with the spy buffer, it must first be frozen. Freezing the spy buffer does not stop data from passing through the flow control FIFO (or interfere with passthrough in passthrough mode), but does stop that data from being written into the spy memory. Once frozen, the spy interface becomes available, and a user can request to read out specific entries from the spy memory. This is done by requesting a specific address in that memory and setting a read enable flag high, which causes the spy buffer to output the word stored at that address. For a spy buffer of size N , the entire memory could then be read by starting at address 0 and reading out each entry up to address $N - 1$. Then, the spy buffer can be unfrozen and normal operations can continue.

Because the memory is circular, at any given time the word stored in address 0 is unlikely to be the oldest entry. Therefore, the spy buffer also reports the address that contains the oldest word (and would be the next to be overwritten on the next write). Still, in a system like the TP where messages are of variable size, there is no guarantee that the oldest entry in the spy buffer will correspond to the beginning of a message. For monitoring data in such a system, the spy buffer can recognize event boundaries and store their locations as well.

¹³³In passthrough mode, the contents of the wire that is being spied on are copied into the flow control FIFO on the rising edge of each clock cycle. This only occurs if the write enable is set high; the enable is treated as a “valid” flag in this mode.

¹³⁴An initial set of software and firmware using IPbus to interact with the spy buffer was developed by Gwen Gardner (Penn).

B.3.3 Event Boundaries and the Event List

When copying a word into the spy memory, the firmware will check to see if the metadata bit— the most significant bit— is set to one. If it is, the spy buffer will then check to see if this is a word indicating the start of a variable-size event. This is done by checking the next eight bits, which are taken to indicate the type of this metadata word. Currently, in the initial version of the spy buffer, the hardcoded eight-bit pattern “10101011” indicates that this is a start-of-event word. This sequence, and the length of the metadata type field, are hardcoded for use in the TP project, but can be changed at compile time¹³⁵.

In addition to the spy memory, the spy buffer contains a smaller circular buffer, known as the event list. Like the spy memory itself, the size of the event list is configurable. When a start-of-event word is written into the spy memory, the address of that word is inserted into the event list. Then, when freezing and reading out the spy buffer, the user can first read out the event list to find the locations of all the start-of-event words in the larger spy memory. Then, a single event can be read out of the spy memory without first having to read the entire thing. As with the spy memory, the address of the oldest entry in the event list is made available to the user.

It is possible that the event list can contain entries which point to events in the main memory that have since been overwritten. Therefore, a special sentinel word is written into the event list whenever the spy memory “wraps around” and overwrites address 0. Entries in the event list consist of an extra, most-significant “sentinel” bit followed by addresses in the spy memory, so sentinel words have this bit set to one. The sentinel words can then be used to tell which entries in the spy buffer are valid and which are not using the following algorithm:

- To read out the spy buffer, first issue a freeze. Then look up the most recently written address in the main spy memory, which we’ll call x .
- Then, look up the most recently written address in the event list.
- Starting from the newest entry in the event list, read out addresses in reverse chronological order until either a sentinel word is seen or the entire list has been read.
- After the first sentinel word, continue reading until either an address $y \leq x$ or a second sentinel word is seen, or until the entire list has been read.

¹³⁵In the TP, a similar pattern (“11001101”) indicates an end-of-event word, but currently no action is taken on non-start-of-event metadata words.

- Any remaining entries in the event list are invalid and have been overwritten, and so should be ignored.

The words read out of the event list then correspond to valid start-of-event words in the main spy memory.

B.3.4 Playback Mode

The spy buffer contains another operating mode, known as “playback mode”, which inverts the normal transmission of data through the firmware. In this mode, instead of writing words into the spy memory as they pass between two blocks, words will be read *out* of the spy memory and into the flow control FIFO (or directly to the next block in passthrough mode). In playback mode, all input from the write-side block is ignored and automatic writes into the spy memory disabled. On each clock cycle, a single word will be copied out of the spy memory and “played back” to the read-side block.

Playback mode could be used to play back whatever data happens to have been captured in the spy memory. However, this is not the recommended way to use this feature. As previously noted, there are no guarantees that the oldest entries in the spy memory represent a complete event. And while it would be possible for the firmware to use the event list to find valid events, the external user would need some mechanism to specify which of these events should be transmitted. And even if this were implemented, it would be somewhat limiting if it were only possible to play back the current contents of the spy memory. A user might wish to test how the firmware will behave in response to a specific pattern that may or may not already be in the spy memory. It would therefore be highly useful to load such patterns in before enabling playback.

To support this, a “playback write” mode was implemented. In this mode, the spy buffer is completely disabled and a new interface for external writes into the spy memory becomes available. When switching to this mode, the spy buffer should first be reset to zero out the spy memory and flow control FIFO. Then, this new interface can be used to write one or more words into the spy buffer. Upon switching to playback mode, the firmware will read words starting from address zero on every clock cycle. The address will increment until the current value of the spy buffer’s write pointer is reached, meaning that it is not necessary to fill the spy memory when writing. Only words written into the spy memory will be played back.

Exactly how playback mode halts depends on how the spy buffer was configured. Two valid playback modes are supported: “playback once” and “playback loop”. In the “once” mode, each

word in the spy buffer will be played back exactly one time. Once the internal counter reaches the write pointer, playback will stop and the spy buffer will remain idle. Normal operations will not resume until the user changes the playback state. On the other hand, “playback loop” does not stop once every word has been read out. In the loop state, playback continues indefinitely, with the internal counter being set back to zero every time the maximum value is reached. The same words are transmitted over and over until the state is changed.

This playback state is governed by a two-bit state register. This state can be toggled through the four different configurations: normal mode, playback write, playback once, and playback loop. It is intended that this state be directly controllable via the same external interface used for monitoring.

B.4 Spy Buffer Status and Verification

An initial version of the spy buffer firmware was written in the Verilog language [208] as an early step in developing the TP board’s firmware. It evolved over time in response to feedback from firmware developers in the HTT community, with key contributions from Bill Ashmanskas (Penn), Priya Sundarajan (Irvine), Daniel Antrim (formerly Irvine), and Gwen Gardner (Penn). Over time, interest grew in potentially using the spy buffer block in projects besides the TP, including the HTT mezzanine cards, as well as in other ATLAS upgrade projects like the new muon trigger processor, known as LOMDT. It is intended to ultimately open-source the current code to facilitate better sharing and collaboration with the community.

The firmware was written to avoid depending on any proprietary libraries from either Xilinx or Intel, the two main FPGA vendors. By avoiding the use of such libraries, it remains possible to use the block on either an Intel or Xilinx FPGA without having to make major alterations. While the TP intended to use a Xilinx FPGA, the mezzanine cards intended to use Intel, and these decisions are subject to change depending on how the broader FPGA ecosystem evolves over the next few years.

The spy buffer firmware was simulated and verified using cocotb, the Python-based verification framework described in Appendix A [107]. Using cocotb, a Python library was written which could write words into the flow control FIFO and read them out from the other side, and compare the results. Methods were then written to support interacting with the spy memory, both to support freezes/readouts and writes/playback. A set of six tests were then developed using this library to exercise the different features of the spy buffer described above. The testbench was designed to run against both a spy buffer with and without the flow control FIFO present to test passthrough mode.

In addition, a verification framework for the TP board was developed using cocotb and the spy buffer¹³⁶. Because firmware blocks in the TP were intended to be connected to two or more spy buffers (one per each input connection and one per each output connection), the flow control interface could be used to verify each block. Python code was developed to load test vectors into the input spy buffer and then store the output of the output spy buffer. This output could then be compared to another set of test vectors containing the expected outputs of this stage of the track processing algorithm. Both sets of test vectors could be produced from an external, high level simulation and the comparison done offline. As of this writing, this verification framework has been used to help test and develop a few of the firmware blocks which have been written for the TP thus far.

¹³⁶This framework written by Daniel Antrim, who was a graduate student at Irvine at the time. It was partially based on the spy buffer testbench. The test vector format, and code to parse it, was contributed by Elliot Lipeles (Penn).

APPENDIX C

Preliminary Run 2 VBF+MET Results

This thesis covers the full run 2 (139fb^{-1}) ATLAS “VBF+MET” analysis: a search for invisible decays of a Higgs boson produced via vector boson fusion. The full run 2 analysis was performed with proton-proton collision data at a center-of-mass energy of $\sqrt{s} = 13\text{TeV}$ recorded from 2015 through 2018. As described in Chapter 6, there were actually *two* versions of the full run 2 analysis released by the ATLAS collaboration: a preliminary result in the spring of 2020, that was released as an online physics briefing¹³⁷ [140], followed by a final result planned to be published as a paper in the fall or winter of 2021. During the approval of the preliminary result, a small number of potential improvements and follow-ups were identified and planned for the paper. After releasing the preliminary result, these items were implemented over the next year in the final version of the analysis.

Most of these changes were covered over the course of Chapters 6, 7, and 8. Theoretical calculations were employed to reweight the $W \rightarrow l\nu$ control region in order to use it to constrain the $Z \rightarrow \nu\nu$ background in the signal region. Additional V+Jets background Monte Carlo was generated. A second, independent FJVT-based technique was developed to estimate the multijet background. And the missing transverse momentum requirement was lowered from $E_{\text{T}}^{\text{miss}} > 200\text{GeV}$ to $E_{\text{T}}^{\text{miss}} > 160\text{GeV}$, turning an 11-bin signal region into a 16-bin signal region. More detail about these changes can be found throughout the main body of the thesis¹³⁸. However, Chapter 9 only presents results from the “final” version of the analysis: not the preliminary. This appendix contains the summary plots, tables, limits, and uncertainties calculated in the “preliminary” version of the 139fb^{-1} analysis [140]. The statistical methods introduced in Section 9.1 were used to produce

¹³⁷This was intended to be presented at the Rencontres de Moriond conference, which was scheduled for late March, 2020, but was cancelled due to the COVID-19 pandemic.

¹³⁸In particular, see Sections 6.3.4, 8.2, and 8.5.

these results as well [193].

The appendix is organized as follows. Section C.1 contains a summary of the preliminary analysis's unblinded signal region before running the fit; this is analogous to the results shown in Section 9.3. Then, Section C.2 contains the various post-fit results calculated in the preliminary analysis, including the observed and expected limit. These results can be compared with the information presented in Section 9.4. Section C.3 then summarizes the differences between the two analyses and quantifies the impact each change has on the limit. And finally, Section C.4 shows a combination of this limit with other Higgs to Invisible analyses on ATLAS in other channels, as well as run 1 data.

C.1 Pre-Fit Signal Region

In the preliminary 139 fb^{-1} analysis, the signal region is divided into eleven bins, as shown in Figure 6.5. All events with $N_{\text{jets}} = 2$ are grouped into a single bin, and then events with exactly two jets are divided into low- and high- $\Delta\phi_{\text{jj}}$ regions (0-1 and 1-2, respectively). These two regions are then split into five m_{jj} bins: $800 < m_{\text{jj}} < 1000\text{ GeV}$, $1000 < m_{\text{jj}} < 1500\text{ GeV}$, $1500 < m_{\text{jj}} < 2000\text{ GeV}$, $2000 < m_{\text{jj}} < 3500\text{ GeV}$, and $m_{\text{jj}} > 3500\text{ GeV}$, giving a total of eleven bins. All events must have $E_{\text{T}}^{\text{miss}} > 200\text{ GeV}$, due to high uncertainties from the rebalance-and-smear based multijet estimate at lower $E_{\text{T}}^{\text{miss}}$. The multijet estimate in this region was performed using a data-driven rebalance and smear technique, as described in Section 8.4.

This version of the analysis was unblinded in early 2020. Figure C.1 shows the agreement between data and Monte Carlo background prediction before the fit, and Table C.1 shows the numbers of observed data, background, and signal events in the signal region and various control regions. Note that, unlike the final versions of these plots shown in Chapter 9, only a fake electron estimate was performed for the preliminary analysis. A rough calculation described in Section 8.3.2 was used to argue that this background is small, and could be neglected. As explained in that section, we revisited this assumption and performed a fake muon estimate for the final version of the analysis; as a result, none of the plots or tables in this section show a fake muon contribution to the background¹³⁹.

¹³⁹A fake muon estimate was also not performed in the 36.1 fb^{-1} partial run 2 analysis; it is a new component of the final 139 fb^{-1} paper.

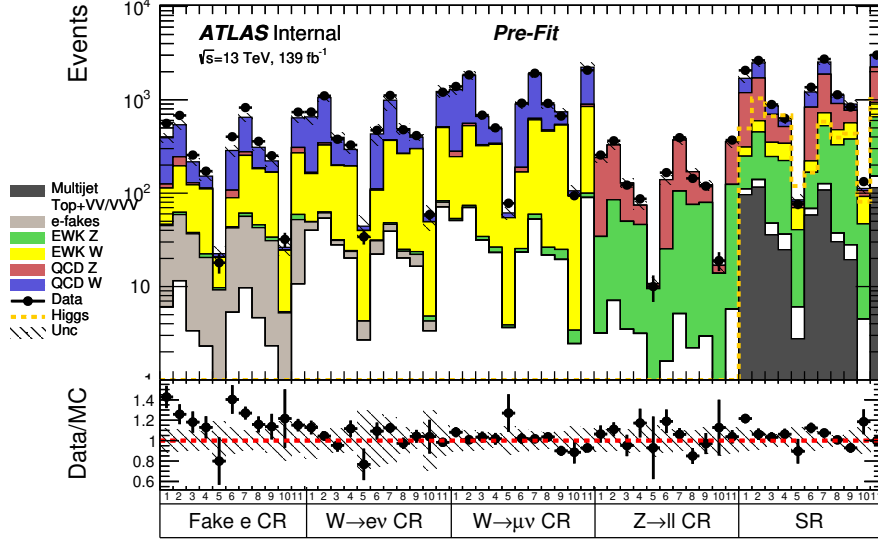


Figure C.1: Plot showing the pre-fit agreement between data and background in all eleven bins of all the main analysis regions of the preliminary 139 fb^{-1} analysis: the signal region, the one- and two-lepton control regions, and the fake electron control region. The error band includes both statistical and systematic uncertainties.

Samples	SR	$Z \rightarrow ll$ CR	$W \rightarrow e\nu$ CR	$W \rightarrow \mu\nu$ CR	Fake- e CR
VBFH125	4973.6	0.0	0.2	0.0	0.0
ggFH125	693.3	0.0	0.0	0.0	0.0
VH125	6.2	0.0	0.9	1.3	0.1
Z QCD	6247.5	1308.9	43.7	196.2	155.1
Z EWK	2555.1	612.2	11.8	42.1	27.2
W QCD	3701.0	0.4	3480.9	6698.1	1753.7
W EWK	1369.3	0.0	2124.0	3765.5	1123.0
Other	152.9	35.4	317.8	389.1	56.3
e -Fakes	0.0	0.0	59.2	0.0	343.6
Multijet	605.5	0.0	0.0	0.0	0.0
Data	15511	2050	6323	11095	4293
Total Bkg	14631.4	1956.9	6037.3	11091.0	3458.9
Data/Bkg	1.060	1.048	1.047	1.000	1.241

Table C.1: Pre-fit yields of signal, background, and data events recorded in the signal region, the one- and two-lepton control regions, and the fake electron control region from the preliminary analysis. The “Other” background includes contributions from both single- t , $t\bar{t}$, and multiboson (VV/VVV) backgrounds [140].

C.2 Post-Fit Results

After unblinding the signal region, a simultaneous likelihood fit was performed in all regions and bins. The preliminary version of the statistical model was mostly identical to the fit described in

Section 9.1, but with a few key differences:

- Most obviously, the fit was performed over an 11-bin region rather than a 16-bin region.
- As noted above, the multijet estimate came entirely from rebalance and smear, with no FJVT transfer factors in any of the bins.
- Also as noted above, a fake muon estimate was not included, just a fake electron estimate.
- The reweighting procedure used to estimate the $Z \rightarrow \nu\nu$ background was *not* used. Instead, two separate transfer factors were calculated for the W and Z backgrounds: the k_W factors were derived from the one-lepton $W \rightarrow l\nu$ control regions, and the k_Z factors were derived from the two-lepton $Z \rightarrow ll$ control regions.

The post-fit agreement between data and background in all bins of all regions is shown in Figure C.2. No significant excess over the Standard Model background prediction is observed in any bin, and data and MC appear to agree quite well within the statistical and systematic errors. The best-fit values for the W and Z transfer factors, as well as the fake electron transfer factors, are shown below in Table C.2.

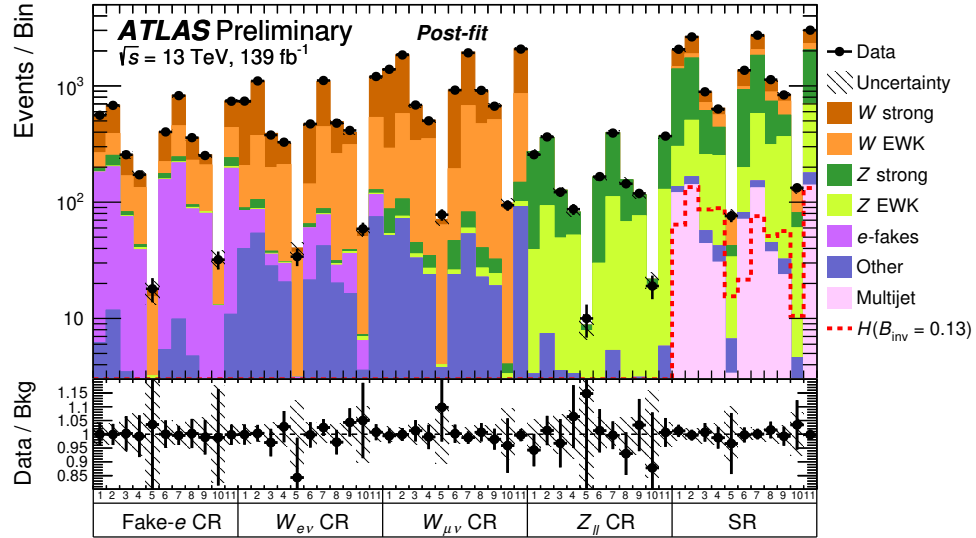


Figure C.2: Plot showing the post-fit agreement between data and background in all sixteen bins of all the main analysis regions, as well as the Higgs to invisible signal normalized to a branching ratio of $\mathcal{B}_{H \rightarrow \text{inv.}} = 0.13$. The post-fit agreement can be compared directly to the pre-fit values seen in Figure C.1 to see the impact of the fit on the background prediction in each bin. No significant data excess over the background is observed in any bin in the signal region.

m_{jj}	$k_W \Delta\phi_{jj} < 1$	k_W	$k_Z \Delta\phi_{jj} < 1$	k_Z	$\beta \Delta\phi_{jj} < 1$	$\beta 1 < \Delta\phi_{jj} < 2$
0.8-1.0 TeV	1.07 ± 0.26	1.02 ± 0.24	1.20 ± 0.30	1.19 ± 0.28	4.49 ± 1.12	4.07 ± 0.96
1.0-1.5 TeV	1.03 ± 0.20	1.08 ± 0.20	1.11 ± 0.24	1.08 ± 0.24	3.97 ± 1.09	4.43 ± 1.08
1.5-2.0 TeV	1.02 ± 0.19	1.01 ± 0.19	1.01 ± 0.22	0.92 ± 0.20	2.14 ± 0.70	2.16 ± 0.72
2.0-3.5 TeV	1.05 ± 0.18	0.96 ± 0.16	1.11 ± 0.20	0.96 ± 0.17	2.01 ± 1.01	2.71 ± 0.78
>3.5 TeV	1.16 ± 0.29	0.98 ± 0.24	0.80 ± 0.25	1.36 ± 0.33	0.28 ± 0.59	2.45 ± 1.35
$2 < N_{\text{jets}} < 5$	0.94 ± 0.22	0.94 ± 0.22	1.04 ± 0.28	1.04 ± 0.28	4.44 ± 1.11	4.44 ± 1.11

Table C.2: Best-fit values of k_W and k_Z , the V+jets transfer factors used to normalize the W and Z backgrounds, and the fake electron transfer factor $\beta^{e\text{-fake}}$ in all eleven bins.

C.2.1 Limits Setting

Since no excess is seen, we proceed with setting a limit on the invisible Higgs branching ratio. The observed (expected) limit from the preliminary 139 fb^{-1} analysis was $\mathcal{B}_{\text{H} \rightarrow \text{inv.}} = 0.132(0.132)$, and the uncertainties on this limit are shown below in Table C.3. The good agreement between observed and expected limits is consistent with no excess being observed in Figure C.2 above. This result is a significant improvement compared to the 36.1 fb^{-1} observed (expected) limit of $\mathcal{B}_{\text{H} \rightarrow \text{inv.}} = 0.37(0.28)$ from the last round of the analysis.

Expected	Observed	$+1\sigma$	-1σ	$+2\sigma$	-2σ
0.132	0.132	0.183	0.095	0.248	0.071

Table C.3: Observed and expected limits on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$, set at a 95% confidence level [140]. This is done from the preliminary 139 fb^{-1} analysis using the likelihood fit shown in Figure C.2, with both statistical and systematic uncertainties included.

C.2.2 Yields and Distributions

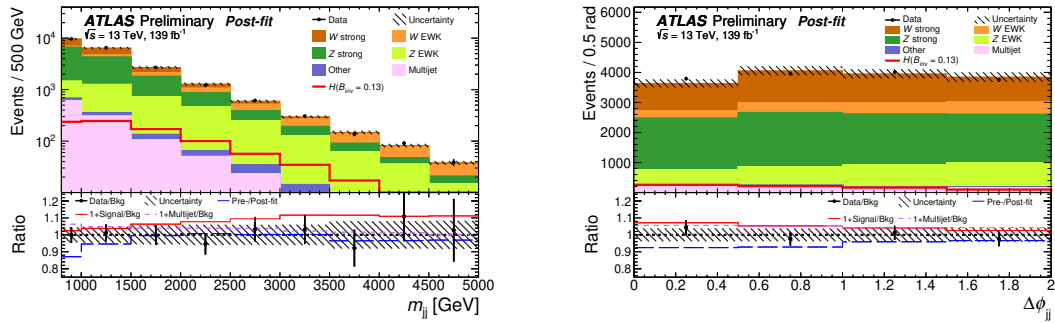


Figure C.3: Post-fit plots showing the m_{jj} and $\Delta\phi_{jj}$ distributions in the preliminary analysis's signal region [140]. Data and background agree very well after the fit.

Figure C.3 shows distributions of the dijet mass m_{jj} , and the azimuthal separation $\Delta\phi_{jj}$ in the

signal region after applying the fit. Good agreement is seen between data and MC across the full range of both variables. The numbers of observed data, signal, and MC events after running the fit are shown below in Table C.4 in the signal region as well as the various control regions. The data to MC ratio in the signal region is extremely close to 1, as expected given the very good agreement between observed and expected limits.

Samples	SR	$Z \rightarrow ll$ CR	$W \rightarrow e\nu$ CR	$W \rightarrow \mu\nu$ CR	$W \rightarrow l\nu$	Fake- e CR
Z EWK	2659 ± 237	634 ± 56	12 ± 1	41 ± 2	53 ± 2	26 ± 2
W EWK	1376 ± 101	0 ± 0	2141 ± 159	3768 ± 273	5909 ± 316	1122 ± 89
Z strong	6805 ± 304	1394 ± 66	48 ± 14	193 ± 13	241 ± 20	153 ± 41
W strong	3753 ± 239	0 ± 0	3526 ± 184	6734 ± 308	10259 ± 359	1755 ± 105
Multijet	740 ± 115	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
Other	155 ± 12	37 ± 5	322 ± 21	395 ± 24	717 ± 32	57 ± 4
e-fakes	0 ± 0	0 ± 0	239 ± 31	0 ± 0	239 ± 31	1187 ± 115
Data	15511 ± 125	2050 ± 45	6323 ± 80	11095 ± 105	17418 ± 132	4293 ± 66
Total Bkg	15489 ± 478	2065 ± 87	6288 ± 246	11131 ± 412	17419 ± 480	4300 ± 184
Data/MC	1.001	0.993	1.006	0.997	1.0	0.998

Table C.4: Post-fit yields of signal, background, and data events recorded in the signal region, the one- and two- lepton control regions, and the fake electron control region from the preliminary analysis [140]. These yields can be compared with the pre-fit numbers shown in Table C.1 to see the impact of the likelihood fit.

C.2.3 Uncertainties

The uncertainties on the expected limit are shown in Table C.5 below. The various statistical and systematic uncertainties are divided into several major groups, and the percentage impact represents the amount the expected limit would improve by were each group to be removed¹⁴⁰. In the preliminary analysis, we can see that data statistics are the dominant uncertainty, followed closely by Monte Carlo background statistics and multijet estimation systematics. The V+Jet MC statistics are still quite large despite the introduction of m_{jj} -sliced samples as described in Chapter 7: in order to remedy this for the final version of the analysis, additional MC events at high m_{jj} were generated in order to bring down these uncertainties further.

It is also disheartening to see that the multijet estimate systematics are so large, given the relative size of this background. The missing transverse momentum cut was increased to $E_T^{\text{miss}} > 200 \text{ GeV}$ from 160 GeV in the preliminary version of the analysis, as otherwise the multijet systematic

¹⁴⁰This is consistent with how the uncertainties were presented in the 36.1 fb^{-1} analysis. However, these limit impacts do not add in quadrature, which makes them a little tricky to interpret. For the final version of the 139 fb^{-1} analysis, we have quantified the amount each group contributes to the *uncertainty* on the limit as well as the impact on the limit itself; these impacts on the uncertainty *do* add in quadrature. See Table 9.8.

Source	Limit Change (Δ %)
Data Stats	17.3
MC Stats	7.9
Multijet	7.0
JER	5.5
JES	1.8
Lepton	4.6
Other	1.9
V+Jets Theory	1.6
Signal Theory	1.0

Table C.5: Table showing the various statistical and systematic uncertainties on the preliminary expected limit, grouped together into several key categories. The uncertainty is presented as the “impact” on the limit, the amount by which the expected limit would improve were the uncertainty group in question to be removed. These numbers can be compared with the 36.1 fb^{-1} analysis or the third column in Table 9.8 [140].

uncertainties would have been larger than even MC statistical uncertainties. Even with the 200 GeV cut, however, we can see that these uncertainties remain quite large. This motivated the introduction of the FJVT-based rebalance and smear approach described in Section 8.5.

C.3 Comparison to Final Analysis

In the results from the final analysis presented in Chapter 9, the observed (expected) limit was $\mathcal{B}_{\text{H} \rightarrow \text{inv.}} = 0.145(0.103)$. Comparing to the preliminary results presented above, we can see that the expected limit improved by about 22%, but that the observed limit got worse. It is useful to try and quantify the impact each of the major changes between the two iterations of the analysis had on the limit, in order to understand both what resulted in the largest sensitivity improvements and, possibly, what resulted in the small excess emerging in the final version of the analysis. Table C.6 shows the impact each major change had on the expected and observed limits.

In the table above, the third entry– “11 bin $E_{\text{T}}^{\text{miss}} > 200 \text{ GeV}$ ”– is the result from the preliminary analysis presented in this appendix. The first two entries are an attempt to extrapolate what limit have been seen had the 36.1 fb^{-1} analysis not been reoptimized. In that analysis, forward jet vertex tagger information was not used, the missing transverse momentum cut was set at 180 GeV, and the signal region was only divided into three m_{jj} bins: $1000 < m_{\text{jj}} < 1500 \text{ GeV}$, $1500 < m_{\text{jj}} < 2000 \text{ GeV}$, and $m_{\text{jj}} > 2000 \text{ GeV}$. Assuming no other changes, with the dataset size increased by a factor of four, the expected sensitivity might have been 18%; or 15% with FJVT information added. Then, the remaining entries in the table quantify the impact of the major changes between the two versions of

Description	Expected Limit	% Change Rel. Prev.	Observed Limit	Significance
3-bin $E_T^{\text{miss}} > 180$ GeV no fjvt	18%	22%		
3-bin $E_T^{\text{miss}} > 180$ GeV	15%	17%		
11-bin $E_T^{\text{miss}} > 200$ GeV	13.2%	13%	13.2%	0σ
Increased MC stats	12.5%	5%	12.5%	0σ
Additional binning	12.1%	3%	13.0%	0.32σ
MJ updates	11.0%	9%	12.5%	0.34σ
W to constrain Z	10.3%	6%	14.5%	1.02σ

Table C.6: Expected and observed limits for different versions of the 139 fb^{-1} VBF+MET analysis. The first two entries show a projection of the expected limit we might have seen had the 36.1 fb^{-1} analysis not been re-optimized, with or without imposing a FJVT requirement on the lead jets. The third entry shows the result from the preliminary analysis, and the subsequent entries show improvements from each of the four major changes made between the preliminary and final analyses.

the 139 fb^{-1} analysis, which were:

- Generating more high- m_{jj} QCD V+jets events to improve background MC statistics.
- Changes to the analysis binning, adding the $160 < E_T^{\text{miss}} < 200$ GeV bins and dividing the $N_{\text{jets}} = 2$ bin into three separate bins, as shown in Figure 6.6.
- Improvements to the multijet estimate; primarily, moving to Monte Carlo as the input to the rebalance and smear procedure described in Section 8.4 and introducing the FJVT-based transfer factor method (Section 8.5) where possible in some of the bins instead.
- Reweighting $W \rightarrow l\nu$ using theory calculations and using the W control region to constrain the $Z \rightarrow \nu\nu$ background, as described in Section 8.2.

The combination of these four changes resulted in the 22% improvement in the expected limit seen in the results presented in Chapter 9.

C.4 Combination with Other Limits

ATLAS performs other Higgs to Invisible analyses in other channels, including gluon-gluon fusion, Higgs-strahlung, and top fusion, as shown in Section 6.1. As of this writing, there is currently only one full run 2 limit on $\mathcal{B}_{\text{H} \rightarrow \text{inv.}}$ available, a preliminary result in the top fusion channel. That analysis set an observed (expected) limit of 0.40 ($0.36^{+0.15}_{-0.10}$) on the invisible branching ratio [222]. A combination was then done in which the preliminary VBF result limit presented in this appendix, the ttH result, and run 1 measurements were combined to set a common ATLAS limit on Higgs to invisible.

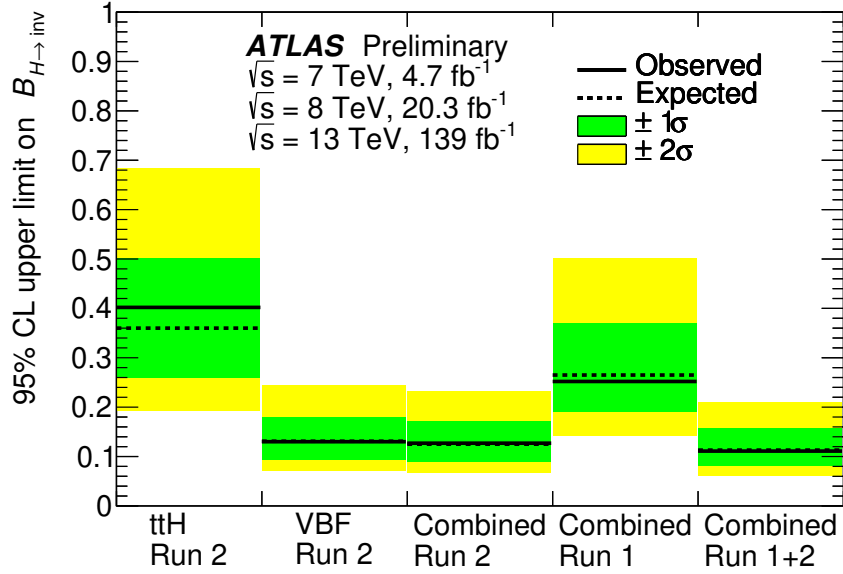


Figure C.4: Observed and expected limits on the branching ratio $B_{H \rightarrow \text{inv}}$ from the preliminary 139 fb^{-1} VBF+MET analysis, combined with a full run 2 ttH analysis and with ATLAS run 1 results [222]. The VBF analysis, as the most significant invisible channel, drives the sensitivity of this combination.

A previous version of this combination for partial run 2 analyses was shown in Section 6.1.2. The basic idea is that the product of the likelihoods from the individual searches comprising the combination is maximized during a combined fit, which is used to set a single limit. The results of this combination for these two full run 2 analyses is shown in Figure C.4, and the combined 95% confidence level limit was set at 0.11 ($0.11^{+0.04}_{-0.03}$) [222]. Note that the improvements listed above in Section C.3 already lead to a more sensitive limit in the VBF channel alone, with the expected limit improved to 0.10. A combination has yet to be run with the final VBF limit presented in Chapter 9, but it will be interesting to see the result, especially once more full run 2 analyses in other channels are included as well.

APPENDIX D

Glossary

ABCStar ATLAS Binary Chip. [54](#)

ACEs Analysis Consultants and Experts. [205–207](#)

AMAC Autonomous Monitor and Control Chip. [54](#), [84](#)

AMTP Associative Memory TP. [275](#), [276](#)

ASIC application specific integrated circuit. [xviii](#), [54](#), [258](#), [259](#), [261](#), [264](#), [274](#)

ATLAS particle detector at the Large Hadron Collider formerly known as A Toroidal LHC ApparatuS. [27](#), [28](#), [258](#)

BCR bunch counter reset. [63](#)

CKM Cabibbo-Kobayashi-Maskawa. [17](#)

CMB Cosmic Microwave Background. [19](#)

cocotb "coroutine cosimulation testbench", open-source Python-based digital logic verification framework. [258](#)

CR control region. [131](#)

DUT design under test. [265](#)

FF flip-flop. [100](#), [259](#)

- FIFO** First-In-First-Out. [67](#), [278](#)
- FJVT** Forward Jet Vertex Tagging. [47](#)
- FPGA** field programmable gate array. [41](#), [80](#), [258](#), [259](#), [261](#), [272](#)
- FTK** Fast Tracker. [274](#), [275](#)
- ggF** Gluon-Gluon Fusion. [124](#)
- gHTT** Global HTT. [275](#), [276](#)
- HCCStar** Hybrid Controller Chip. [xviii](#), [54](#), [81](#)
- HDL** hardware description language. [xviii](#), [78](#), [259–263](#), [266](#), [267](#)
- HL-LHC** High Luminosity Large Hadron Collider. [48](#)
- HLT** High Level Trigger. [273](#), [274](#)
- HPR** high priority register. [68](#)
- HS** Hard Scatter. [217–220](#)
- HS+PU** Hard Scatter and Pile-Up. [217–220](#), [223](#)
- HTT** Hardware Tracking for the Trigger. [52](#), [272](#), [274](#), [275](#)
- HTT-IF** HTT Interface. [275](#)
- ITK** Inner Tracker. [48](#)
- JVT** Jet Vertex Tagging. [47](#)
- L0A** level 0 accept. [63](#)
- L1** level 1 accept. [66](#)
- LCB** L0A/CMD/BCR. [63](#)
- LHC** Large Hadron Collider. [6](#), [23](#)
- LIPS** Lorentz invariant phase space. [8](#)

LO Leading Order. [145](#), [150](#), [161](#), [192](#)

NLO Next to Leading Order. [145](#), [150](#), [152](#), [161](#), [192](#)

NNLO Next to Next to Leading Order. [145](#), [152](#), [192](#), [205](#)

PDF parton distribution function. [154](#)

PLL phase-locked loop. [76](#), [116](#)

PMNS Pontecorvo–Maki–Nakagawa–Sakata. [17](#)

PRM Pattern Recognition Mezzanine. [275](#)

Python open source, general purpose programming language, widely used in scientific computing (and many other fields). [258](#)

QCD Quantum chromodynamics. [6](#), [18](#)

QED Quantum electrodynamics. [6](#)

R3 regional readout request. [66](#)

rHTT Regional HTT. [275](#), [276](#)

RTL register-transfer level. [81](#), [259](#), [263](#), [268](#)

SEE single event effects. [99](#)

SET single event transient. [100](#)

SEU single event upset. [100](#)

SR signal region. [129](#)

SSTP Second-Stage TP. [275](#), [276](#)

TDAQ Trigger and Data Acquisition. [40](#), [51](#), [272](#), [273](#)

TFM Track Fitter Mezzanine. [275](#), [276](#)

TMR triple modular redundancy. [104](#)

TP Track Processor. [272](#), [275](#)

ttH Top Fusion. [124](#)

UVM Universal Verification Methodology. [261](#)

VBF Vector Boson Fusion. [124](#)

Verilog a hardware description language. [78](#), [259](#), [270](#), [273](#)

VH Higgs-strahlung. [124](#)

VHDL a hardware description language; the "V" is nominally short for "Very High Speed Integrated Circuit". [259](#)

VHPI VHDL Procedural Interface. [263](#), [264](#)

VPI Verilog Procedural Interface. [263](#), [264](#)

WIMP Weakly Interacting Massive Particle. [20](#), [124](#), [256](#)

Bibliography

- [1] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003 (cit. on pp. xviii, 1, 23, 26–39).
- [2] V. C. Rubin and W. K. Ford Jr., *Rotation of the Andromeda Nebula from a Spectroscopic Survey of Emission Regions*, Astrophys. J. **159** (1970) 379 (cit. on pp. 1, 19).
- [3] D. Clowe et al., *A direct empirical proof of the existence of dark matter*, Astrophys. J. Lett. **648** (2006) L109, arXiv: [astro-ph/0608407](#) (cit. on pp. 1, 18, 19).
- [4] CMS Collaboration, *The CMS experiment at the CERN LHC*, JINST **3** (2008) S08004 (cit. on pp. 1, 26).
- [5] L. R. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008) S08001, This report is an abridged version of the LHC Design Report (CERN-2004-003), URL: <https://cds.cern.ch/record/1129806> (cit. on pp. 1, 23, 25, 26).
- [6] ATLAS Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Phys. Lett. B **716** (2012) 1, arXiv: [1207.7214 \[hep-ex\]](#) (cit. on pp. 1, 17).
- [7] CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Phys. Lett. B **716** (2012) 30, arXiv: [1207.7235 \[hep-ex\]](#) (cit. on pp. 1, 17).
- [8] G. Arcadi, A. Djouadi, and M. Raidal, *Dark Matter through the Higgs portal*, Phys. Rept. **842** (2020) 1, arXiv: [1903.03616 \[hep-ph\]](#) (cit. on pp. 2, 123, 251).
- [9] M. Schwartz, *Quantum Field Theory and the Standard Model*, Cambridge University Press, 2014, ISBN: 9781107034730 (cit. on pp. 3, 7–9, 12–16).
- [10] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory*, Addison-Wesley, 1995, ISBN: 0201503972 (cit. on pp. 3, 6, 9–14, 17).
- [11] M. Thomson, *Modern Particle Physics*, New York: Cambridge University Press, 2013, ISBN: 978-1-107-03426-6 (cit. on pp. 3, 6–9, 13, 15, 26, 29, 30, 34).
- [12] P. A. Zyla et al. (Particle Data Group), *Review of Particle Physics*, PTEP **2020** (2020) 083C01 (cit. on pp. 3, 20, 35, 37, 123, 125, 154, 202, 203, 230).

- [13] Wikimedia Commons Users MissMJ, Cush, *Standard Model of Particle Physics*, CC-BY 3.0 (<https://creativecommons.org/licenses/by/3.0/deed.en>), 2021, URL: https://commons.wikimedia.org/w/index.php?title=File:Standard_Model_of_Elementary_Particles.svg (cit. on p. 4).
- [14] LHCb Collaboration, *The LHCb Detector at the LHC*, JINST **3** (2008) S08005 (cit. on pp. 4, 26).
- [15] LHCb Collaboration, *Observation of the resonant character of the $Z(4430)^-$ state*, Phys. Rev. Lett. **112** (2014) 222002, arXiv: 1404.1903 [hep-ex] (cit. on p. 4).
- [16] E. Noether, *Invariante variationsprobleme*, Nachr. d. König. Gesellsch. d. Wiss. zu Göttingen, Math-phys. Klasse (1918) 235, M. A. Tavel's English translation originally appeared in Transport Theory and Statistical Physics, 1 (3), 183–207 (1971)., eprint: [arXiv:physics/0503066](https://arxiv.org/abs/physics/0503066) (cit. on p. 5).
- [17] D. Griffiths, *Introduction to Elementary Particles*, Physics textbook, Wiley, 2008, ISBN: 9783527618477 (cit. on p. 7).
- [18] J. Ellis, *TikZ-Feynman: Feynman diagrams with TikZ*, Comput. Phys. Commun. **210** (2017) 103, arXiv: 1601.05437 [hep-ph] (cit. on p. 10).
- [19] K. G. Wilson and J. Kogut, *The renormalization group and the expansion*, Physics Reports **12** (1974) 75, ISSN: 0370-1573 (cit. on p. 12).
- [20] C. G. Callan Jr., *Broken scale invariance in scalar field theory*, Phys. Rev. D **2** (1970) 1541 (cit. on p. 12).
- [21] K. Symanzik, *Small distance behavior in field theory and power counting*, Commun. Math. Phys. **18** (1970) 227 (cit. on p. 12).
- [22] S. Glashow, *Partial symmetries of weak interactions*, Nucl.Phys. **22** (1961) 579 (cit. on p. 12).
- [23] J. Goldstone, A. Salam, and S. Weinberg, *Broken symmetries*, Physical Review **127** (1962) 965 (cit. on p. 12).
- [24] J. Goldstone, *Field theories with superconductor solutions*, Il Nuovo Cimento (1955-1965) **19** (1 1961) 154, 10.1007/BF02812722, ISSN: 1827-6121 (cit. on p. 12).
- [25] F. Englert and R. Brout, *Broken symmetry and the mass of gauge vector mesons*, Phys. Rev. Lett. **13** (9 1964) 321 (cit. on p. 13).
- [26] P. W. Higgs, *Broken symmetries, massless particles and gauge fields*, Physics Letters **12** (1964) 132, ISSN: 0031-9163 (cit. on p. 13).
- [27] P. W. Higgs, *Broken symmetries and the masses of gauge bosons*, Phys. Rev. Lett. **13** (16 1964) 508 (cit. on p. 13).
- [28] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble, *Global conservation laws and massless particles*, Phys. Rev. Lett. **13** (20 1964) 585 (cit. on p. 13).
- [29] M. Kobayashi and T. Maskawa, *CP-Violation in the renormalizable theory of weak interaction*, Progress of Theoretical Physics **49** (1973) 652 (cit. on p. 17).
- [30] Z. Maki, M. Nakagawa, and S. Sakata, *Remarks on the unified model of elementary particles*, Prog. Theor. Phys. **28** (1962) 870 (cit. on p. 17).

- [31] S. Dittmaier et al., *Handbook of LHC Higgs Cross Sections: 1. Inclusive Observables*, (2011), arXiv: [1101.0593 \[hep-ph\]](#) (cit. on pp. 18, 124).
- [32] A. Mahdavi, H. y. Hoekstra, A. y. Babul, D. y. Balam, and P. Capak, *A Dark Core in Abell 520*, *Astrophys. J.* **668** (2007) 806, arXiv: [0706.3048 \[astro-ph\]](#) (cit. on p. 19).
- [33] N. Aghanim et al., *Planck 2018 results. VI. Cosmological parameters*, *Astron. Astrophys.* **641** (2020) A6, [Erratum: *Astron. Astrophys.* 652, C4 (2021)], arXiv: [1807.06209 \[astro-ph.CO\]](#) (cit. on p. 19).
- [34] P. van Dokkum et al., *A galaxy lacking dark matter*, *Nature* **555** (2018) 629, arXiv: [1803.10237 \[astro-ph.GA\]](#) (cit. on p. 20).
- [35] R. Peccei and H. R. Quinn, *CP Conservation in the Presence of Pseudoparticles*, *Phys. Rev. Lett.* **38** (25 1977) 1440 (cit. on p. 20).
- [36] R. Peccei and H. R. Quinn, *Constraints imposed by CP conservation in the presence of pseudoparticles*, *Phys. Rev. D* **16** (6 1977) 1791 (cit. on p. 20).
- [37] W. Hu, R. Barkana, and A. Gruzinov, *Cold and fuzzy dark matter*, *Phys. Rev. Lett.* **85** (2000) 1158, arXiv: [astro-ph/0003365](#) (cit. on p. 20).
- [38] S. Kanemura, S. Matsumoto, T. Nabeshima, and N. Okada, *Can WIMP Dark Matter overcome the Nightmare Scenario?*, *Phys. Rev. D* **82** (2010) 055026, arXiv: [1005.5651 \[hep-ph\]](#) (cit. on pp. 20, 21, 124).
- [39] D. S. Akerib et al., *The Large Underground Xenon (LUX) Experiment*, *Nucl. Instrum. Meth. A* **704** (2013) 111, arXiv: [1211.3788 \[physics.ins-det\]](#) (cit. on pp. 21, 251).
- [40] X. Cao et al., *PandaX: A Liquid Xenon Dark Matter Experiment at CJPL*, *Sci. China Phys. Mech. Astron.* **57** (2014) 1476, arXiv: [1405.2882 \[physics.ins-det\]](#) (cit. on pp. 21, 251).
- [41] J. Conrad, “Indirect Detection of WIMP Dark Matter: a compact review”, *Interplay between Particle and Astroparticle physics (IPA2014) London, United Kingdom, August 18-22, 2014*, 2014, arXiv: [1411.1925 \[hep-ph\]](#) (cit. on p. 21).
- [42] *The Large Area Telescope on the Fermi Gamma-ray Space Telescope Mission*, *Astrophys. J.* **697** (2009) 1071, arXiv: [0902.1089 \[astro-ph.IM\]](#) (cit. on p. 21).
- [43] B. Patt and F. Wilczek, *Higgs-field portal into hidden sectors*, (2006), arXiv: [hep-ph/0605188](#) (cit. on p. 21).
- [44] A. Djouadi, O. Lebedev, Y. Mambrini, and J. Quevillon, *Implications of LHC searches for Higgs-portal dark matter*, *Phys. Lett. B* **709** (2012) 65, arXiv: [1112.3299 \[hep-ph\]](#) (cit. on pp. 21, 22, 252).
- [45] S. Baek, P. Ko, and W.-I. Park, *Invisible Higgs Decay Width vs. Dark Matter Direct Detection Cross Section in Higgs Portal Dark Matter Models*, *Phys. Rev. D* **90** (2014) 055014, arXiv: [1405.3530 \[hep-ph\]](#) (cit. on pp. 22, 252).
- [46] G. Arcadi, A. Djouadi, and M. Kado, *The Higgs-portal for vector dark matter and the effective field theory approach: A reappraisal*, *Phys. Lett. B* **805** (2020) 135427, arXiv: [2001.10750 \[hep-ph\]](#) (cit. on pp. 22, 252).

- [47] M. Zaazoua, L. Truong, K. A. Assamagan, and F. Fassi, *Higgs portal vector dark matter interpretation: review of Effective Field Theory approach and ultraviolet complete models*, (2021), arXiv: [2107.01252 \[hep-ph\]](#) (cit. on pp. 22, 252, 253).
- [48] *LEP design report*, Geneva: CERN, 1984, URL: <https://cds.cern.ch/record/102083> (cit. on p. 23).
- [49] R. Barate et al., *Search for the standard model Higgs boson at LEP*, *Phys. Lett. B* **565** (2003) 61, arXiv: [hep-ex/0306033](#) (cit. on p. 23).
- [50] J. Haffner, *The CERN accelerator complex. Complexe des accélérateurs du CERN*, (2013), General Photo, URL: <https://cds.cern.ch/record/1621894> (cit. on p. 24).
- [51] M. Vretenar et al., *Linac4 design report*, vol. 6, CERN Yellow Reports: Monographs, Geneva: CERN, 2020, URL: <https://cds.cern.ch/record/2736208> (cit. on p. 25).
- [52] E. Henley and A. Garcia, *Subatomic Physics*, World Scientific, 2007, ISBN: 9789812700568, URL: <https://books.google.com/books?id=gcIPAQAAMAAJ> (cit. on pp. 25, 30).
- [53] ALICE Collaboration, *The ALICE experiment at the CERN LHC*, *JINST* **3** (2008) S08002 (cit. on p. 26).
- [54] ATLAS Collaboration, *Expected performance of the ATLAS experiment: detector, trigger and physics*, Geneva: CERN, 2008, eprint: [arXiv:0901.0512](#) (cit. on pp. 26, 29, 41–44, 46).
- [55] ATLAS Collaboration, *Luminosity Public Results Run 2*, URL: <https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResultsRun2> (visited on 10/25/2021) (cit. on p. 27).
- [56] ATLAS Collaboration, *Luminosity determination in pp collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector at the LHC*, (2019), URL: <https://cdsweb.cern.ch/record/2677054/> (cit. on pp. 27, 48, 49, 128, 132, 143).
- [57] J. Pequeno and P. Schaffner, “How ATLAS detects particles: diagram of particle paths in the detector”, 2013, URL: <https://cds.cern.ch/record/1505342> (cit. on p. 29).
- [58] G. Avoni et al., *The new LUCID-2 detector for luminosity measurement and monitoring in ATLAS*, *JINST* **13** (2018) P07017 (cit. on p. 30).
- [59] O. Adriani et al., *The LHCf detector at the CERN Large Hadron Collider*, *JINST* **3** (2008) S08006 (cit. on p. 30).
- [60] *ATLAS inner detector: Technical Design Report, Vol. 1*, Geneva: CERN, 1997, URL: <https://cds.cern.ch/record/331063> (cit. on p. 30).
- [61] G. Aad et al., *ATLAS pixel detector electronics and sensors*, *JINST* **3** (2008) P07007 (cit. on p. 31).
- [62] ATLAS Collaboration, *ATLAS Insertable B-Layer Technical Design Report*, CERN, 2010, URL: <https://cds.cern.ch/record/1291633> (cit. on p. 32).
- [63] A. Ahmad et al., *The Silicon microstrip sensors of the ATLAS semiconductor tracker*, *Nucl. Instrum. Meth. A* **578** (2007) 98 (cit. on p. 33).
- [64] E. Abat et al., *The ATLAS Transition Radiation Tracker (TRT) proportional drift tube: Design and performance*, *JINST* **3** (2008) P02013 (cit. on p. 33).

- [65] *ATLAS liquid-argon calorimeter: Technical Design Report*, Geneva: CERN, 1996, URL: <https://cds.cern.ch/record/331061> (cit. on p. 35).
- [66] *ATLAS tile calorimeter: Technical Design Report*, Geneva: CERN, 1996, URL: <https://cds.cern.ch/record/331062> (cit. on p. 36).
- [67] A. Artamonov et al., *The ATLAS forward calorimeters*, *JINST* **3** (2008) P02010 (cit. on p. 37).
- [68] *ATLAS muon spectrometer: Technical Design Report*, Geneva: CERN, 1997, URL: <https://cds.cern.ch/record/331068> (cit. on p. 37).
- [69] CERN CASTOR, *Physics Data on Tape at CERN*, URL: https://castorwww.web.cern.ch/castorwww/namespace_statistics.png (visited on 08/01/2021) (cit. on p. 40).
- [70] ATLAS Collaboration, *Performance of the ATLAS Trigger System in 2015*, *Eur. Phys. J. C* **77** (2017) 317, arXiv: 1611.09661 [hep-ex] (cit. on pp. 40, 41).
- [71] ATLAS TDAQ Collaboration, *The ATLAS Data Acquisition and High Level Trigger system*, *JINST* **11** (2016) P06008 (cit. on p. 40).
- [72] ATLAS Collaboration, *The ATLAS Fast TrackKer system*, *JINST* **16** (2021) P07006, arXiv: 2101.05078 [physics.ins-det] (cit. on pp. 40, 274).
- [73] ATLAS Collaboration, *Reconstruction of hadronic decay products of tau leptons with the ATLAS experiment*, *Eur. Phys. J. C* **76** (2016) 295, arXiv: 1512.05955 [hep-ex] (cit. on p. 41).
- [74] ATLAS Collaboration, *Athena*, version 21.0.127, 2021, URL: <https://doi.org/10.5281/zenodo.4772550> (cit. on p. 42).
- [75] *ROOT*, <http://root.cern.ch>, 2013 (cit. on pp. 42, 102, 262).
- [76] *Atlas Computing: technical design report*, Geneva: CERN, 2005 (cit. on p. 42).
- [77] R. Fruhwirth, *Application of Kalman filtering to track and vertex fitting*, *Nucl. Instrum. Meth.* **A262** (1987) 444 (cit. on p. 42).
- [78] T. G. Cornelissen et al., *The global χ^2 track fitter in ATLAS*, *J. Phys. Conf. Ser.* **119** (2008) 032013, ed. by R. Sobie, R. Tafirout, and J. Thomson (cit. on p. 42).
- [79] ATLAS Collaboration, *Track Reconstruction Performance of the ATLAS Inner Detector at $\sqrt{s} = 13$ TeV*, ATL-PHYS-PUB-2015-018, 2015, URL: <https://cds.cern.ch/record/2037683> (cit. on p. 42).
- [80] ATLAS Collaboration, *Performance of the ATLAS track reconstruction algorithms in dense environments in LHC Run 2*, *Eur. Phys. J. C* **77** (2017) 673, arXiv: 1704.07983 [hep-ex] (cit. on p. 42).
- [81] ATLAS Collaboration, *Reconstruction of primary vertices at the ATLAS experiment in Run 1 proton–proton collisions at the LHC*, *Eur. Phys. J. C* **77** (2017) 332, arXiv: 1611.10235 [hep-ex] (cit. on p. 42).
- [82] ATLAS Collaboration, *Electron reconstruction and identification in the ATLAS experiment using the 2015 and 2016 LHC proton-proton collision data at $\sqrt{s} = 13$ TeV*, *Eur. Phys. J. C* **79** (2019) 639, arXiv: 1902.04655 [physics.ins-det] (cit. on pp. 43, 143).

- [83] ATLAS Collaboration, *Electron and photon performance measurements with the ATLAS detector using the 2015–2017 LHC proton-proton collision data*, *JINST* **14** (2019) P12006, arXiv: [1908.00005 \[hep-ex\]](#) (cit. on pp. 43, 135, 143).
- [84] ATLAS Collaboration, *Topological cell clustering in the ATLAS calorimeters and its performance in LHC Run 1*, *Eur. Phys. J. C* **77** (2017) 490, arXiv: [1603.02934 \[hep-ex\]](#) (cit. on p. 43).
- [85] ATLAS Collaboration, *Muon reconstruction performance of the ATLAS detector in proton-proton collision data at $\sqrt{s} = 13$ TeV*, *Eur. Phys. J. C* **76** (2016) 292, arXiv: [1603.05598 \[hep-ex\]](#) (cit. on pp. 44, 135, 143).
- [86] ATLAS Collaboration, *Jet reconstruction and performance using particle flow with the ATLAS Detector*, *Eur. Phys. J. C* **77** (2017) 466, arXiv: [1703.10485 \[hep-ex\]](#) (cit. on pp. 44, 46, 134).
- [87] M. Cacciari, G. P. Salam, and G. Soyez, *The anti- k_t jet clustering algorithm*, *JHEP* **04** (2008) 063, arXiv: [0802.1189 \[hep-ph\]](#) (cit. on pp. 44, 45, 134, 156, 170).
- [88] ATLAS Collaboration, *Measurements of b -jet tagging efficiency with the ATLAS detector using $t\bar{t}$ events at $\sqrt{s} = 13$ TeV*, *JHEP* **08** (2018) 089, arXiv: [1805.01845 \[hep-ex\]](#) (cit. on pp. 46, 138).
- [89] M. Aaboud et al., *Performance of missing transverse momentum reconstruction with the ATLAS detector using proton-proton collisions at $\sqrt{s} = 13$ TeV*, *Eur. Phys. J. C* **78** (2018) 903, arXiv: [1802.08168 \[hep-ex\]](#) (cit. on pp. 46, 137).
- [90] ATLAS Collaboration, *Performance of pile-up mitigation techniques for jets in pp collisions at $\sqrt{s} = 8$ TeV using the ATLAS detector*, *Eur. Phys. J. C* **76** (2016) 581, arXiv: [1510.03823 \[hep-ex\]](#) (cit. on pp. 47, 134, 144).
- [91] ATLAS Collaboration, *Identification and rejection of pile-up jets at high pseudorapidity with the ATLAS detector*, *Eur. Phys. J. C* **77** (2017) 580, arXiv: [1705.02211 \[hep-ex\]](#) (cit. on pp. 47, 138, 144), Erratum: *Eur. Phys. J. C* **77** (2017) 712.
- [92] ATLAS Collaboration, *Forward jet vertex tagging using the particle flow algorithm*, (2019), All figures including auxiliary figures are available at <https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2019-026>, URL: <https://cds.cern.ch/record/2683100> (cit. on p. 47).
- [93] ATLAS Collaboration, *ATLAS Phase-II Upgrade Scoping Document*, CERN, 2015, URL: <https://cds.cern.ch/record/2055248> (cit. on pp. 48, 49).
- [94] ATLAS Collaboration, *Technical Design Report for the ATLAS Inner Tracker Strip Detector*, TDR CERN-LHCC-2017-005. ATLAS-TDR-025, CERN, 2017, URL: <https://cds.cern.ch/record/2257755> (cit. on pp. 50, 52–57, 61).
- [95] ATLAS Collaboration, *Technical Design Report for the ATLAS Inner Tracker Pixel Detector*, TDR ATLAS-TDR-030, CERN, 2017, URL: <https://cds.cern.ch/record/2285585> (cit. on pp. 50, 275).
- [96] ATLAS Collaboration, *ATLAS Liquid Argon Calorimeter Phase-II Upgrade: Technical Design Report*, TDR, CERN, 2017, URL: <https://cds.cern.ch/record/2285582> (cit. on p. 51).

- [97] ATLAS Collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS Tile Calorimeter*, TDR ATLAS-TDR-028, CERN, 2017, URL: <https://cds.cern.ch/record/2285583> (cit. on p. 51).
- [98] ATLAS Collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS Muon Spectrometer*, TDR, CERN, 2017, URL: <https://cds.cern.ch/record/2285580> (cit. on p. 51).
- [99] ATLAS Collaboration, *Technical Design Report: A High-Granularity Timing Detector for the ATLAS Phase-II Upgrade*, TDR, CERN, 2020, URL: <https://cds.cern.ch/record/2719855> (cit. on p. 51).
- [100] ATLAS Collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*, TDR, CERN, 2017, URL: <https://cds.cern.ch/record/2285584> (cit. on pp. 51, 52, 272, 274–276).
- [101] F. Faccio et al., “FEAST2: A Radiation and Magnetic Field Tolerant Point-of-Load Buck DC/DC Converter”, *2014 IEEE Radiation Effects Data Workshop (REDW)*, 2014 1 (cit. on p. 54).
- [102] W. Lu et al., *Development of the ABCStar front-end chip for the ATLAS silicon strip upgrade*, *Journal of Instrumentation* **12** (2017) C04017, URL: <https://doi.org/10.1088/1748-0221/12/04/c04017> (cit. on pp. 57, 58).
- [103] A. X. Widmer, “DC-balanced 6B/8B transmission code with local parity”, 6876315, 2005, URL: <https://www.freepatentsonline.com/6876315.html> (cit. on p. 63).
- [104] R. W. Hamming, *Error detecting and error correcting codes*, *The Bell System Technical Journal* **29** (1950) 147 (cit. on pp. 70, 102, 105).
- [105] P. A. Franaszek and A. X. Widmer, “Byte oriented DC balanced (0,4) 8B/10B partitioned block transmission code”, 4486739, 1984, URL: <https://www.freepatentsonline.com/4486739.html> (cit. on p. 75).
- [106] A. X. Widmer and P. A. Franaszek, *A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code*, *IBM Journal of Research and Development* **27** (1983) 440 (cit. on p. 75).
- [107] C. Higgs, S. Hodgeson, et al., *cocotb*, URL: <https://docs.cocotb.org/en/stable/> (visited on 10/13/2020) (cit. on pp. 81, 258, 262, 263, 265, 267, 282).
- [108] B. J. Rosser, “Verification of Readout Electronics in the ATLAS ITk Strips Detector”, *Proceedings of the 2019 Meeting of the Division of Particles and Fields of the American Physical Society*, ed. by L. Skinnari, 2019, arXiv: [1910.06694](https://arxiv.org/abs/1910.06694) [[physics.ins-det](https://arxiv.org/abs/1910.06694)] (cit. on pp. 81, 259).
- [109] C. R. Harris et al., *Array programming with NumPy*, *Nature* **585** (2020) 357 (cit. on pp. 96, 262).
- [110] J. D. Hunter, *Matplotlib: A 2D graphics environment*, *Computing in Science & Engineering* **9** (2007) 90 (cit. on p. 96).
- [111] F. Faccio and G. Cervelli, *Radiation-induced edge effects in deep submicron CMOS transistors*, *IEEE Transactions on Nuclear Science* **52** (2005) 2413 (cit. on p. 98).

- [112] M. Gadlage et al., *Single event transient pulse widths in digital microcircuits*, *IEEE Transactions on Nuclear Science* **51** (2004) 3285 (cit. on pp. 99, 100).
- [113] M. Huhtinen and F. Faccio, *Computational method to estimate Single Event Upset rates in an accelerator environment*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **450** (2000) 155, ISSN: 0168-9002, URL: <https://www.sciencedirect.com/science/article/pii/S0168900200001558> (cit. on pp. 99, 100).
- [114] L. Eklund et al., *SEU rate estimates for the ATLAS/SCT front-end ASIC*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **515** (2003) 415, ISSN: 0168-9002, URL: <https://www.sciencedirect.com/science/article/pii/S0168900203023003> (cit. on pp. 99, 100).
- [115] J. L. Titus, *An Updated Perspective of Single Event Gate Rupture and Single Event Burnout in Power MOSFETs*, *IEEE Transactions on Nuclear Science* **60** (2013) 1912 (cit. on p. 100).
- [116] E. Blackmore, “Operation of the TRIUMF (20-500 MeV) proton irradiation facility”, *2000 IEEE Radiation Effects Data Workshop. Workshop Record. Held in conjunction with IEEE Nuclear and Space Radiation Effects Conference (Cat. No.00TH8527)*, 2000 1 (cit. on p. 101).
- [117] E. Blackmore, P. Dodd, and M. Shaneyfelt, “Improved capabilities for proton and neutron irradiations at TRIUMF”, *2003 IEEE Radiation Effects Data Workshop*, 2003 149 (cit. on p. 101).
- [118] G. Berger, G. Ryckewaert, R. Harboe-Sorensen, and L. Adams, “The heavy ion irradiation facility at CYCLONE - a dedicated SEE beam line”, *1996 IEEE Radiation Effects Data Workshop. Workshop Record. Held in conjunction with The IEEE Nuclear and Space Radiation Effects Conference*, 1996 78 (cit. on p. 101).
- [119] J. Dandoy, “Development and Testing of the ATLAS ITk HCCStar ASIC”, 2019 Meeting of the Division of Particles and Fields of the American Physical Society, 2019, URL: <https://indico.cern.ch/event/782953/contributions/3468379/> (cit. on pp. 103, 104).
- [120] J. M. Johnson and M. J. Wirthlin, “Voter Insertion Algorithms for FPGA Designs Using Triple Modular Redundancy”, *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA ’10, Monterey, California, USA: Association for Computing Machinery, 2010 249, ISBN: 9781605589114, URL: <https://dl.acm.org/doi/10.1145/1723112.1723154> (cit. on p. 104).
- [121] S. Kulis, *Single Event Effects mitigation with TMRG tool*, *Journal of Instrumentation* **12** (2017) C01082, URL: <http://stacks.iop.org/1748-0221/12/i=01/a=C01082> (cit. on pp. 104, 106).
- [122] Wikimedia Commons User Mewtow, *Triple Modular Redundancy et sa variante améliorée*, CC-BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/deed.en>), 2016, URL: https://commons.wikimedia.org/wiki/File:Triple_Modular_Redundancy_et_sa_variante_am%C3%A9lior%C3%A9e.png (cit. on p. 106).

- [123] S. Takamaeda-Yamazaki,
“Pyverilog: A Python-Based Hardware Design Processing Toolkit for Verilog HDL”,
Applied Reconfigurable Computing, vol. 9040, Lecture Notes in Computer Science,
Springer International Publishing, 2015 451,
URL: http://dx.doi.org/10.1007/978-3-319-16214-0_42 (cit. on p. 108).
- [124] D. de Florian et al.,
Handbook of LHC Higgs Cross Sections: 4. Deciphering the Nature of the Higgs Sector,
2/2017 (2016), arXiv: [1610.07922 \[hep-ph\]](#) (cit. on pp. 123, 124, 145).
- [125] S. Dittmaier et al., *Handbook of LHC Higgs Cross Sections: 2. Differential Distributions*,
(2012), arXiv: [1201.3084 \[hep-ph\]](#) (cit. on p. 124).
- [126] ATLAS Collaboration, *Standard Model Summary Plots June 2021*,
(2021), All figures including auxiliary figures are available at
<https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2021-032>, URL: <https://cds.cern.ch/record/2777014> (cit. on pp. 125, 203).
- [127] ATLAS Collaboration, *Measurements of top-quark pair to Z-boson cross-section ratios at $\sqrt{s} = 13, 8, 7$ TeV with the ATLAS detector*, **JHEP 02** (2017) 117,
arXiv: [1612.03636 \[hep-ex\]](#) (cit. on p. 125).
- [128] ATLAS Collaboration, *Search for new phenomena in events with an energetic jet and missing transverse momentum in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*,
Phys. Rev. D 103 (2021) 112006, arXiv: [2102.10874 \[hep-ex\]](#) (cit. on p. 125).
- [129] ATLAS Collaboration,
Measurements of the production cross section of a Z boson in association with high transverse momentum jets in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector,
(2021), All figures including auxiliary figures are available at
<https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/CONFNOTES/ATLAS-CONF-2021-033>, URL: <https://cds.cern.ch/record/2777239> (cit. on p. 125).
- [130] ATLAS Collaboration, *Differential cross-section measurements for the electroweak production of dijets in association with a Z boson in proton-proton collisions at ATLAS*,
Eur. Phys. J. C 81 (2021) 163, arXiv: [2006.15458 \[hep-ex\]](#) (cit. on p. 125).
- [131] ATLAS Collaboration,
Combination of searches for invisible Higgs boson decays with the ATLAS experiment,
Phys. Rev. Lett. 122 (2019) 231801, arXiv: [1904.05105 \[hep-ex\]](#) (cit. on pp. 126, 127).
- [132] CMS Collaboration, *Search for invisible decays of a Higgs boson produced through vector boson fusion in proton-proton collisions at $\sqrt{s} = 13$ TeV*, **Phys. Lett. B 793** (2019) 520,
arXiv: [1809.05937 \[hep-ex\]](#) (cit. on pp. 126–128).
- [133] ATLAS Collaboration, *Search for invisible decays of a Higgs boson using vector-boson fusion in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector*, **JHEP 01** (2016) 172,
arXiv: [1508.07869 \[hep-ex\]](#) (cit. on p. 127).
- [134] W. K. Balunas, “A Search for Nothing: Dark Matter and Invisible Decays of the Higgs Boson at the ATLAS Detector”, Presented 19 Apr 2018, 2018,
URL: <https://cds.cern.ch/record/2316129> (cit. on pp. 127, 128).
- [135] ATLAS Collaboration, *Search for invisible Higgs boson decays in vector boson fusion at $\sqrt{s} = 13$ TeV with the ATLAS detector*, **Phys. Lett. B 793** (2019) 499,
arXiv: [1809.06682 \[hep-ex\]](#) (cit. on pp. 127, 130, 139, 217, 246, 250).

- [136] ATLAS Collaboration, *Search for an invisibly decaying Higgs boson or dark matter candidates produced in association with a Z boson in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *Phys. Lett. B* **776** (2018) 318, arXiv: [1708.09624 \[hep-ex\]](#) (cit. on p. 127).
- [137] ATLAS Collaboration, *Search for dark matter in events with a hadronically decaying vector boson and missing transverse momentum in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *JHEP* **10** (2018) 180, arXiv: [1807.11471 \[hep-ex\]](#) (cit. on p. 127).
- [138] N. Metropolis and S. Ulam, *The Monte Carlo Method*, *Journal of the American Statistical Association* **44** (1949) 335, ISSN: 01621459, URL: <http://www.jstor.org/stable/2280232> (cit. on p. 129).
- [139] N. Metropolis, *The Beginning of the Monte Carlo Method*, Los Alamos Science (1987) 125, URL: <https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-88-9067> (cit. on pp. 129, 130).
- [140] ATLAS Collaboration, *Search for invisible Higgs boson decays with vector boson fusion signatures with the ATLAS detector using an integrated luminosity of 139 fb^{-1}* , (2020), URL: <https://cds.cern.ch/record/2715447> (cit. on pp. 132, 284, 286, 288–290).
- [141] ATLAS Collaboration, *The ATLAS transverse-momentum trigger performance at the LHC in 2011*, ATLAS-CONF-2014-002, 2014, URL: <https://cds.cern.ch/record/1647616> (cit. on pp. 133, 142).
- [142] M. Cacciari, G. P. Salam, and G. Soyez, *FastJet User Manual*, *Eur. Phys. J. C* **72** (2012) 1896, arXiv: [1111.6097 \[hep-ph\]](#) (cit. on p. 134).
- [143] ATLAS Collaboration, *Selection of jets produced in 13 TeV proton–proton collisions with the ATLAS detector*, ATLAS-CONF-2015-029, 2015, URL: <https://cds.cern.ch/record/2037702> (cit. on p. 134).
- [144] ATLAS Collaboration, *Observation and measurement of Higgs boson decays to WW^* with the ATLAS detector*, *Phys. Rev. D* **92** (2015) 012006, arXiv: [1412.2641 \[hep-ex\]](#) (cit. on p. 135).
- [145] ATLAS Collaboration, *Observation of electroweak production of two jets in association with an isolated photon and missing transverse momentum, and search for a Higgs boson decaying into invisible particles at 13 TeV with the ATLAS detector*, (2021), arXiv: [2109.00925 \[hep-ex\]](#) (cit. on p. 136).
- [146] ATLAS Collaboration, *Measurement of the photon identification efficiencies with the ATLAS detector using LHC Run-1 data*, (2016), arXiv: [1606.01813 \[hep-ex\]](#) (cit. on p. 136).
- [147] ATLAS Collaboration, *Performance of electron and photon triggers in ATLAS during LHC Run 2*, *Eur. Phys. J. C* **80** (2020) 47, arXiv: [1909.00761 \[hep-ex\]](#) (cit. on p. 143).
- [148] ATLAS Collaboration, *Performance of the ATLAS muon trigger in pp collisions at $\sqrt{s} = 8$ TeV*, *Eur. Phys. J. C* **75** (2015) 120, arXiv: [1408.3179 \[hep-ex\]](#) (cit. on p. 143).

- [149] ATLAS Collaboration,
Validation of the muon momentum corrections for the ATLAS simulation using the $\Upsilon \rightarrow \mu\mu$ channel based on 36.5 fb^{-1} of pp collision data collected in 2015 and 2016, (2019),
URL: <https://cds.cern.ch/record/2674152> (cit. on p. 143).
- [150] ATLAS Collaboration, *Jet Calibration and Systematic Uncertainties for Jets Reconstructed in the ATLAS Detector at $\sqrt{s} = 13 \text{ TeV}$* , ATL-PHYS-PUB-2015-015, 2015,
URL: <https://cds.cern.ch/record/2037613> (cit. on p. 144).
- [151] ATLAS Collaboration, *Jet energy scale measurements and their systematic uncertainties in proton–proton collisions at $\sqrt{s} = 13 \text{ TeV}$ with the ATLAS detector*,
Phys. Rev. D **96** (2017) 072002, arXiv: 1703.09665 [[hep-ex](#)] (cit. on p. 144).
- [152] ATLAS Collaboration,
 E_T^{miss} performance in the ATLAS detector using 2015–2016 LHC p–p collisions, (2018),
URL: <https://cds.cern.ch/record/2625233> (cit. on p. 144).
- [153] A. Denner, S. Dittmaier, S. Kallweit, and A. Mück, *HAWK 2.0: A Monte Carlo program for Higgs production in vector-boson fusion and Higgs strahlung at hadron colliders*,
Comput. Phys. Commun. **195** (2015) 161, arXiv: 1412.5390 [[hep-ph](#)] (cit. on pp. 146, 158).
- [154] ATLAS Collaboration, *The ATLAS Simulation Infrastructure*,
Eur. Phys. J. C **70** (2010) 823, arXiv: 1005.4568 [[physics.ins-det](#)] (cit. on p. 151).
- [155] E. Bothmann et al., *Event Generation with Sherpa 2.2*, *SciPost Phys.* **7** (2019) 034,
arXiv: 1905.09127 [[hep-ph](#)] (cit. on pp. 151, 153, 155, 160, 162, 164, 206).
- [156] T. Gleisberg et al., *Event generation with SHERPA 1.1*, *JHEP* **02** (2009) 007,
arXiv: 0811.4622 [[hep-ph](#)] (cit. on p. 151).
- [157] F. Krauss, R. Kuhn, and G. Soff, *AMEGIC++ 1.0: A Matrix element generator in C++*,
JHEP **02** (2002) 044, arXiv: [hep-ph/0109036](#) (cit. on pp. 152, 153).
- [158] R. Kleiss and R. Pittau, *Weight optimization in multichannel Monte Carlo*,
Comput. Phys. Commun. **83** (1994) 141, arXiv: [hep-ph/9405257](#) (cit. on pp. 152, 153).
- [159] T. Gleisberg and S. Höche, *Comix, a new matrix element generator*, *JHEP* **12** (2008) 039,
arXiv: 0808.3674 [[hep-ph](#)] (cit. on pp. 153, 160).
- [160] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*,
JHEP **07** (2014) 079, arXiv: 1405.0301 [[hep-ph](#)] (cit. on pp. 153, 162).
- [161] J. Alwall et al., *A Standard format for Les Houches event files*,
Comput. Phys. Commun. **176** (2007) 300, arXiv: [hep-ph/0609017](#) (cit. on p. 153).
- [162] S. Schumann and F. Krauss,
A parton shower algorithm based on Catani-Seymour dipole factorization,
JHEP **03** (2008) 038, arXiv: 0709.1027 [[hep-ph](#)] (cit. on p. 153).
- [163] S. Catani and M. H. Seymour,
A General algorithm for calculating jet cross-sections in NLO QCD,
Nucl. Phys. B **485** (1997) 291, [Erratum: *Nucl.Phys.B* 510, 503–504 (1998)],
arXiv: [hep-ph/9605323](#) (cit. on p. 153).
- [164] S. Höche and S. Prestel, *The midpoint between dipole and parton showers*,
Eur. Phys. J. C **75** (2015) 461, arXiv: 1506.05057 [[hep-ph](#)] (cit. on p. 153).

- [165] T. Sjöstrand et al., *An introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159, arXiv: [1410.3012 \[hep-ph\]](#) (cit. on pp. [153](#), [157](#), [158](#), [162](#), [163](#)).
- [166] R. D. Ball et al., *Parton distributions for the LHC Run II*, *JHEP* **04** (2015) 040, arXiv: [1410.8849 \[hep-ph\]](#) (cit. on pp. [154](#), [160](#), [162](#), [163](#)).
- [167] L. A. Harland-Lang, A. D. Martin, P. Motylinski, and R. S. Thorne, *Parton distributions in the LHC era: MMHT 2014 PDFs*, *Eur. Phys. J. C* **75** (2015) 204, arXiv: [1412.3989 \[hep-ph\]](#) (cit. on pp. [154](#), [162](#)).
- [168] J. Butterworth et al., *PDF4LHC recommendations for LHC Run II*, *J. Phys. G* **43** (2016) 023001, arXiv: [1510.03865 \[hep-ph\]](#) (cit. on pp. [154](#), [158](#)).
- [169] E. Bothmann, M. Schönherr, and S. Schumann, *Reweighting QCD matrix-element and parton-shower calculations*, *Eur. Phys. J. C* **76** (2016) 590, arXiv: [1606.08753 \[hep-ph\]](#) (cit. on p. [155](#)).
- [170] S. Frixione and B. R. Webber, *Matching NLO QCD computations and parton shower simulations*, *JHEP* **06** (2002) 029, arXiv: [hep-ph/0204244](#) (cit. on pp. [155](#), [167](#), [168](#)).
- [171] S. Hoeche, F. Krauss, M. Schonherr, and F. Siegert, *A critical appraisal of NLO+PS matching methods*, *JHEP* **09** (2012) 049, arXiv: [1111.1220 \[hep-ph\]](#) (cit. on p. [155](#)).
- [172] S. Höche, F. Krauss, S. Schumann, and F. Siegert, *QCD matrix elements and truncated showers*, *JHEP* **05** (2009) 053, arXiv: [0903.1219 \[hep-ph\]](#) (cit. on p. [156](#)).
- [173] S. Höche, F. Krauss, M. Schönherr, and F. Siegert, *QCD matrix elements + parton showers. The NLO case*, *JHEP* **04** (2013) 027, arXiv: [1207.5030 \[hep-ph\]](#) (cit. on pp. [156](#), [170](#)).
- [174] L. Lonnblad and S. Prestel, *Matching Tree-Level Matrix Elements with Interleaved Showers*, *JHEP* **03** (2012) 019, arXiv: [1109.4829 \[hep-ph\]](#) (cit. on p. [157](#)).
- [175] S. Agostinelli et al., *GEANT4 - a simulation toolkit*, *Nucl. Instrum. Meth.* **A506** (2003) 250 (cit. on p. [157](#)).
- [176] S. Frixione, P. Nason, and C. Oleari, *Matching NLO QCD computations with Parton Shower simulations: the POWHEG method*, *JHEP* **11** (2007) 070, arXiv: [0709.2092 \[hep-ph\]](#) (cit. on p. [158](#)).
- [177] P. Nason and C. Oleari, *NLO Higgs boson production via vector-boson fusion matched with shower in POWHEG*, *JHEP* **02** (2010) 037, arXiv: [0911.5299 \[hep-ph\]](#) (cit. on p. [158](#)).
- [178] K. Hamilton, P. Nason, E. Re, and G. Zanderighi, *NNLOPS simulation of Higgs boson production*, *JHEP* **10** (2013) 222, arXiv: [1309.0017 \[hep-ph\]](#) (cit. on p. [158](#)).
- [179] S. Alioli, P. Nason, C. Oleari, and E. Re, *A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX*, *JHEP* **06** (2010) 043, arXiv: [1002.2581 \[hep-ph\]](#) (cit. on p. [158](#)).
- [180] I. W. Stewart and F. J. Tackmann, *Theory Uncertainties for Higgs and Other Searches Using Jet Bins*, *Phys. Rev. D* **85** (2012) 034011, arXiv: [1107.2117 \[hep-ph\]](#) (cit. on p. [159](#)).

- [181] F. Cascioli, P. Maierhofer, and S. Pozzorini, *Scattering Amplitudes with Open Loops*, [*Phys. Rev. Lett.* **108** \(2012\) 111601](#), arXiv: [1111.5206 \[hep-ph\]](#) (cit. on p. 160).
- [182] M. Bahr et al., *Herwig++ Physics and Manual*, [*Eur. Phys. J. C* **58** \(2008\) 639](#), arXiv: [0803.0883 \[hep-ph\]](#) (cit. on pp. 162, 207).
- [183] J. Bellm et al., *Herwig 7.2 release note*, [*Eur. Phys. J. C* **80** \(2020\) 452](#), arXiv: [1912.06509 \[hep-ph\]](#) (cit. on p. 162).
- [184] K. Arnold et al., *VBFNLO: A Parton level Monte Carlo for processes with electroweak bosons*, [*Comput. Phys. Commun.* **180** \(2009\) 1661](#), arXiv: [0811.4559 \[hep-ph\]](#) (cit. on p. 162).
- [185] M. Dobbs and J. B. Hansen, *The HepMC C++ Monte Carlo event record for High Energy Physics*, [*Comput. Phys. Commun.* **134** \(2001\) 41](#) (cit. on p. 167).
- [186] A. Buckley et al., *The HepMC3 event record library for Monte Carlo event generators*, [*Comput. Phys. Commun.* **260** \(2021\) 107310](#), arXiv: [1912.08005 \[hep-ph\]](#) (cit. on p. 167).
- [187] CMS Collaboration, *Search for invisible decays of a Higgs boson produced via vector boson fusion with 138 fb⁻¹ of proton-proton collisions at $\sqrt{s} = 13$ TeV*, (2021), URL: <https://cds.cern.ch/record/2784571> (cit. on pp. 176, 254, 255).
- [188] ATLAS Collaboration, *Object-based missing transverse momentum significance in the ATLAS detector*, (2018), URL: <https://cds.cern.ch/record/2630948> (cit. on pp. 195, 209, 213).
- [189] ATLAS Collaboration, *Measurement of $W^\pm$ and Z-boson production cross sections in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, [*Phys. Lett. B* **759** \(2016\) 601](#), arXiv: [1603.09222 \[hep-ex\]](#) (cit. on p. 203).
- [190] J. M. Lindert et al., *Precise predictions for V+jets dark matter backgrounds*, [*Eur. Phys. J. C* **77** \(2017\) 829](#), arXiv: [1705.04664 \[hep-ph\]](#) (cit. on p. 204).
- [191] ATLAS Collaboration, *Search for dark matter and other new phenomena in events with an energetic jet and large missing transverse momentum using the ATLAS detector*, [*JHEP* **01** \(2018\) 126](#), arXiv: [1711.03301 \[hep-ex\]](#) (cit. on p. 204).
- [192] J. D'Hondt et al., *Fitting of Event Topologies with External Kinematic Constraints in CMS*, (2006), URL: <https://cds.cern.ch/record/926540> (cit. on p. 218).
- [193] G. Cowan, K. Cranmer, E. Gross, and O. Vitells, *Asymptotic formulae for likelihood-based tests of new physics*, [*Eur. Phys. J. C* **71** \(2011\) 1554](#), [Erratum: *Eur. Phys. J. C* **73** (2013) 2501], arXiv: [1007.1727 \[physics.data-an\]](#) (cit. on pp. 229, 234, 235, 285).
- [194] T. Junk, *Confidence level computation for combining searches with small statistics*, [*Nucl. Instrum. Meth. A* **434** \(1999\) 435](#), arXiv: [hep-ex/9902006](#) (cit. on p. 229).
- [195] A. L. Read, *Presentation of search results: the CL_s technique*, [*J. Phys. G* **28** \(2002\) 2693](#) (cit. on pp. 229, 235).
- [196] K. Cranmer, G. Lewis, L. Moneta, A. Shibata, and W. Verkerke, *HistFactory: A tool for creating statistical models for use with RooFit and RooStats*, (2012), URL: <https://cds.cern.ch/record/1456844> (cit. on p. 229).
- [197] M. Baak et al., *HistFitter software framework for statistical data analysis*, [*Eur. Phys. J. C* **75** \(2015\) 153](#), arXiv: [1410.1280 \[hep-ex\]](#) (cit. on p. 229).

- [198] W. Verkerke and D. P. Kirkby, *The RooFit toolkit for data modeling*, eConf **C0303241** (2003) MOLT007, ed. by L. Lyons and M. Karagoz, arXiv: [physics/0306116](#) (cit. on p. 229).
- [199] L. Moneta et al., *The RooStats Project*, PoS **ACAT2010** (2010) 057, ed. by T. Speer et al., arXiv: [1009.1003 \[physics.data-an\]](#) (cit. on p. 229).
- [200] D. Bertsekas and J. Tsitsiklis, *Introduction to Probability, 2nd Edition*, Athena Scientific, 2008, ISBN: 9781886529236, URL: <http://athenasc.com/probbook.html> (cit. on pp. 229–231).
- [201] K. Cranmer and I. Yavin, *RECAST: Extending the Impact of Existing Analyses*, JHEP **04** (2011) 038, arXiv: [1010.2506 \[hep-ex\]](#) (cit. on p. 251).
- [202] A. DiFranzo, P. J. Fox, and T. M. P. Tait, *Vector Dark Matter through a Radiative Higgs Portal*, JHEP **04** (2016) 135, arXiv: [1512.06853 \[hep-ph\]](#) (cit. on p. 252).
- [203] Y. Meng et al., *Dark Matter Search Results from the PandaX-4T Commissioning Run*, (2021), arXiv: [2107.13438 \[hep-ex\]](#) (cit. on pp. 252, 253).
- [204] P. Agnes et al., *Low-Mass Dark Matter Search with the DarkSide-50 Experiment*, Phys. Rev. Lett. **121** (2018) 081307, arXiv: [1802.06994 \[astro-ph.HE\]](#) (cit. on pp. 252, 253).
- [205] A. H. Abdelhameed et al., *First results from the CRESST-III low-mass dark matter program*, Phys. Rev. D **100** (2019) 102002, arXiv: [1904.00498 \[astro-ph.CO\]](#) (cit. on pp. 252, 253).
- [206] J. Billard, L. Strigari, and E. Figueroa-Feliciano, *Implication of neutrino backgrounds on the reach of next generation dark matter direct detection experiments*, Phys. Rev. D **89** (2014) 023524, arXiv: [1307.5458 \[hep-ph\]](#) (cit. on p. 252).
- [207] B. J. Rosser, “Cocotb: a Python-based digital logic verification framework”, CERN, 2018, URL: <https://indico.cern.ch/event/776422/> (cit. on p. 259).
- [208] *IEEE Standard for Verilog Hardware Description Language*, IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001) (2006) 1 (cit. on pp. 259, 263, 265, 273, 282).
- [209] *IEEE Standard for VHDL Language Reference Manual*, IEEE Std 1076-2019 (2019) 1 (cit. on p. 259).
- [210] *IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language*, IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012) (2018) 1 (cit. on pp. 260, 261).
- [211] *IEEE Standard for Universal Verification Methodology Language Reference Manual*, IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017) (2020) 1 (cit. on p. 261).
- [212] Python Software Foundation, *Python*, URL: <https://www.python.org> (visited on 06/01/2021) (cit. on p. 262).
- [213] P. Virtanen et al., *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods **17** (2020) 261 (cit. on p. 262).
- [214] *Free and Open Source Silicon Foundation*, URL: <https://www.fossi-foundation.org/> (visited on 06/01/2021) (cit. on p. 262).

- [215] D. E. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, Third, Reading, Mass.: Addison-Wesley, 1997, ISBN: 0201896834 9780201896831 (cit. on pp. 268, 269).
- [216] M. E. Conway, *Design of a Separable Transition Diagram Compiler*, Communications of the ACM **6** (1963) 396 (cit. on p. 269).
- [217] G. van Rossum, *Asynchronous IO Support Rebooted: the "asyncio" Module*, PEP 3156, 2012, URL: <https://www.python.org/dev/peps/pep-3156/> (cit. on p. 269).
- [218] M. Shochet et al., *Fast Tracker (FTK) Technical Design Report*, TDR, ATLAS Fast Tracker Technical Design Report, 2013, URL: <https://cds.cern.ch/record/1552953> (cit. on pp. 274, 275).
- [219] C. E. Cummings, "Simulation and synthesis techniques for asynchronous FIFO design", *SNUG 2002 (Synopsis Users Group Conference, San Jose, CA, 2002) User Papers*, 2002, URL: http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIF01.pdf (cit. on p. 278).
- [220] C. E. Cummings, *Clock domain crossing (CDC) design & verification techniques using SystemVerilog*, SNUG-2008, Boston (2008), URL: http://www.sunburst-design.com/papers/CummingsSNUG2008Boston_CDC.pdf (cit. on p. 278).
- [221] C. G. Larrea et al., *IPbus: a flexible Ethernet-based control system for xTCA hardware*, *Journal of Instrumentation* **10** (2015) C02019, URL: <https://doi.org/10.1088/1748-0221/10/02/c02019> (cit. on p. 279).
- [222] ATLAS Collaboration, *Combination of searches for invisible Higgs boson decays with the ATLAS experiment*, (2020), URL: <https://cds.cern.ch/record/2743055> (cit. on pp. 291, 292).