NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF APPLIED MATHEMATICAL AND PHYSICAL SCIENCES

DEPARTMENT OF PHYSICS
HIGH ENERGY PHYSICS LABORATORY

Research and Development of the Electronics and Data Acquisition System for the New Small Wheel Upgrade of the ATLAS Detector at CERN and Performance Evaluation of the Micromegas Chamber

A dissertation presented by

Christos Bakalis

to
The Department of Physics
for the degree of
Doctor of Philosophy
in the subject of Physics

Athens, July 2021

Research and Development of the Electronics and Data Acquisition System for the New Small Wheel Upgrade of the ATLAS Detector at CERN and Performance Evaluation of the Micromegas Chamber

PhD Thesis

Christos Bakalis

Advisor: Theodoros Alexopoulos
Professor N.T.U.A.

Exam committee:

.....	.....	.....	
Theodoros Alexopoulos Professor - NTUA, Greece	Stavros Maltezos Professor - NTUA, Greece	Evangelos Gazis Professor - NTUA, Greece	
.....	.....	.....	.....
Venetios Polychronakos Researcher A - BNL, USA	Theodoros Geralis Research Director - NCSR Demokritos, Greece	Dimitrios Sampsonidis Professor - AUTH, Greece	Georgios Iakovidis Researcher - BNL, USA

Athens, July 2021

.....

Χρήστος Μπακάλης
Διπλωματούχος Φυσικός Εφαρμογών, Σ.Ε.Μ.Φ.Ε., Ε.Μ.Π

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΦΥΣΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΦΥΣΙΚΗΣ ΥΨΗΛΩΝ ΕΝΕΡΓΕΙΩΝ ΚΑΙ ΣΥΝΑΦΟΥΣ ΟΡΓΑΝΟΛΟΓΙΑΣ

Έρευνα και Ανάπτυξη των Ηλεκτρονικών και του Συστήματος Απόσπασης Δεδομένων της Αναβάθμισης New Small Wheel του Ανιχνευτή ATLAS στο CERN, και Χαρακτηρισμός της Απόδοσης του Θαλάμου Micromegas

Διδακτορική διατριβή του

Χρήστου Μπακάλη

Διπλωματούχου Φυσικού Εφαρμογών, Σ.Ε.Μ.Φ.Ε., Ε.Μ.Π

ΕΠΙΒΛΕΠΩΝ: Θεόδωρος Αλεξόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΦΥΣΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΦΥΣΙΚΗΣ ΥΨΗΛΩΝ ΕΝΕΡΓΕΙΩΝ ΚΑΙ ΣΥΝΑΦΟΥΣ ΟΡΓΑΝΟΛΟΓΙΑΣ

Έρευνα και Ανάπτυξη των Ηλεκτρονικών και του Συστήματος Απόσπασης Δεδομένων της Αναβάθμισης New Small Wheel του Ανιχνευτή ATLAS στο CERN, και Χαρακτηρισμός της Απόδοσης του Θαλάμου Micromegas

Διδακτορική διατριβή του

Χρήστου Μπακάλη

Διπλωματούχου Φυσικού Εφαρμογών, Σ.Ε.Μ.Φ.Ε., Ε.Μ.Π

ΤΡΙΜΕΛΗΣ ΣΥΜΒΟΥΛΕΥΤΙΚΗ ΕΠΙΤΡΟΠΗ:

1. Θεόδωρος Αλεξόπουλος, Καθ. Ε.Μ.Π.
2. Σταύρος Μαλτέζος, Καθ. Ε.Μ.Π.
3. Ευάγγελος Γαζής, Καθ. Ε.Μ.Π.

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

1. Θεόδωρος Αλεξόπουλος - Καθηγητής, Ε.Μ.Π.
2. Σταύρος Μαλτέζος - Καθηγητής, Ε.Μ.Π.
3. Ευάγγελος Γαζής - Καθηγητής, Ε.Μ.Π.
4. Βενέτιος Πολυχρονάκος - Ερευνητής Ά, BNL (USA)
5. Θεόδωρος Γέραλης - Διευθυντής Ερευνών, Ε.Κ.Ε.Φ.Ε. Δημόκριτος
6. Δημήτριος Σαμψωνίδης - Καθηγητής, ΑΠΘ
7. Γεώργιος Ιακωβίδης - Ερευνητής, BNL (USA)

Αθήνα, Ιούλιος 2021

Περίληψη στα Ελληνικά

Ο Μεγάλος Επιταχυντής Αδρονίων (Large Hadron Collider (LHC)), του Ευρωπαϊκού Κέντρου Πυρηνικών Ερευνών (CERN) ξεκίνησε να επιταχύνει τις πρώτες του δέσμες το 2008. Από τις πρώτες μέρες λειτουργίας του, ο LHC έχει επιτρέψει στην επιστημονική κοινότητα να τελέσει πρωτοπόρα πειράματα, που έχουν σαν στόχο να απαντήσουν θεμελιώδη ερωτήματα για τη φύση της ύλης και της ενέργειας. Επειδή τα πειράματα φυσικής υψηλών ενεργειών είναι βασισμένα στη συλλογή δεδομένων μεγάλης κλίμακας, η αύξηση του ρυθμού αντιδράσεων θεωρείται θέμα μείζονας σημασίας. Όσο πιο υψηλός ο ρυθμός αλληλεπιδράσεων, τόσο περισσότερα δεδομένα καταγράφονται, και σπάνια φαινόμενα που υπό άλλες συνθήκες θα επικαλύπτονταν από άλλες διεργασίες, περισσότερο συχνές, θα μπορέσουν να μελετηθούν. Για το λόγο αυτό, ο LHC θα προβεί στις ανάλογες αναβαθμίσεις, οι οποίες θα αυξήσουν την ενέργεια κέντρου μάζας, και την φωτεινότητά του. Μετά από την αναβάθμιση της δεύτερης μεγάλης παύσης (Long Shutdown (LS2)), η οποία θα ολοκληρωθεί το 2022, οι δέσμες θα έχουν μία τελική φωτεινότητα της τάξης των $\mathscr{L} = 2.5 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$. Επιπλέον αναβαθμίσεις που θα οδηγήσουν στην Phase-II 2026-2038, θα αυξήσουν τη φωτεινότητα στα $5 - 7.5 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, με την ενέργεια κέντρου μάζας να φτάνει τα $14\,\mathrm{TeV}$. Η αύξηση της φωτεινότητας θα οδηγήσει και σε αύξηση του ρυθμού αλληλεπιδράσεων, άρα και στη ροή σωματιδίων που διαπερνούν τους ανιχνευτές του μεγάλου επιταχυντή αδρονίων. Για αυτό το λόγο ο ανιχνευτής Toroidal LHC ApparatuS (ATLAS), που είναι ο μεγαλύτερος του LHC, θα αντικαταστήσει τα εσωτερικά καπάκια του μιονικού φασματομέτρου κατά τη διάρκεια της δεύτερης μεγάλης παύσης. Η αναβάθμιση New Small Wheel (NSW) όπως καλείται, θα αποτελείται από δύο τεχνολογίες ανιχνευτών, τους Micromegas (MM) και τους small-strip Thin Gap Chambers (sTGC). Ο NSW έχει σχεδιαστεί να υπομένει το αυξημένο υπόβαθρο λόγω των αναβαθμίσεων του επιταχυντή, προσφέροντας δεδομένα ανακατασκευής τροχιών μιονίων στον ATLAS, καθώς και πληροφορίες σκανδαλισμού. Ο ακρογωνιαίος λίθος του συστήματος

ανάγνωσης δεδομένων του NSW, είναι το VMM Application-Specific Integrated Circuit (ASIC), μία ηλεκτρονική μονάδα που θα χρησιμοποιηθεί και από τις δύο τεχνολογίες ανιχνευτών του NSW. Λόγω του σχεδιασμού του, το VMM έχει προταθεί και σε μία πληθώρα άλλων πειραμάτων που κάνουν χρήση ανάλογων ανιχνευτικών συστημάτων. Το VMM αποτελείται από 64 ανεξάρτητα κανάλια, κάθε ένα εκ των οποίων προβαίνει σε μετρήσεις ακριβείας πάνω στους ηλεκτρονικούς παλμούς που δημιουργούνται από τους ανιχνευτές όταν διαπεραστούν από μιόνια, ενώ προσφέρει και γρήγορα δεδομένα για το σύστημα σκανδαλισμού του ATLAS. Η πρώτη έκδοση του VMM έκανε την εμφάνισή του το 2012, και μετά από τέσσερις εκδόσεις, αποφάνθηκε ότι είναι έτοιμο να εξυπηρετήσει τις ανάγκες του NSW. Καμία από αυτές τις αναβαθμίσεις δεν θα μπορούσε να είχε ολοκληρωθεί, αν δεν υπήρχε μία αξιόπιστη πλατφόρμα χαρακτηρισμού της μονάδας. Αυτή η πλατφόρμα ήρθε στη μορφή του VMM Readout System (VRS), που κάνει χρήση μονάδων Field-Programmable Gate Array (FPGA), προκειμένου να ληφθούν τα δεδομένα από το VMM, και να διαμορφωθεί τη λειτουργία του. Με έμφαση στην ευελιξία, το υλικολογισμικό των FPGA του VRS, είχε σχεδιαστεί με τέτοιο τρόπο προκειμένου να εξυπηρετήσει διάφορα σενάρια λήψης δεδομένων (π.χ. εργαστηριακές συνθήκες χωρίς ανιχνευτή, ή τεστ-δέσμης). Ένα μεγάλο κομμάτι της παρούσας διατριβής αφιερώνεται στην περιγραφή της αρχιτεκτονικής του εν λόγω υλικολογισμικού, που αναπτύχθηκε για να καλύψει τις ανάγκες της αναβάθμισης του ATLAS NSW. Το σύστημα χρησιμοποιήθηκε για να επιβεβαιώσει την ορθή λειτουργία του VMM, να κάνει τον μαζικό έλεγχο των τελικών μονάδων του VMM πριν αυτά εγκατασταθούν στον ATLAS, και να λάβει τα δεδομένα από το VMM, μαζί με τον ανιχνευτή Micromegas, σε συνθήκες τεστ-δέσμης. Μετά την παραγωγή των τελικών μονάδων ASIC που διαβάζουν και διαμορφώνουν τις λειτουργίες του VMM στο πείραμα ATLAS, το σύστημα που βασιζόταν σε FPGA αντικαταστάθηκε από το τελικό, που είχε σα βάση ένα σύστημα λήψης δεδομένων επόμενης γενιάς, που ονομάζεται Front-End LInk eXchange (FELIX). Ένα ποσοστό της παρούσας εργασίας περιγράφει τα εργαλεία λογισμικού που αναπτύχθηκαν προκειμένου να διευκολυνθεί η διαδικασία ενσωμάτωσης του ηλεκτρονικού συστήματος του NSW με το FELIX. Επίσης, τα εν λόγω πακέτα λογισμικού χρησιμοποιήθηκαν και για στη διαδικασία ελέγχου της ορθής λειτουργίας των τελικών ανιχνευτών του Micromegas, πριν αυτοί εγκατασταθούν στον ATLAS, καθώς τα δεδομένα τους λαμβάνονταν από το τελικό σύστημα λήψης δεδομένων. Το τελευταίο Κεφάλαιο της παρούσας διατριβής, αφιερώνεται στην περιγραφή του Slow Control Adapter eXtension (SCAX), το οποίο είναι ένα πακέτο υλικολογισμικού που ενσωματώνεται σε ένα FPGA και μιμείται μία βασική ηλεκτρονική μονάδα του NSW, ονόματι SCA ASIC. Το SCA είναι μια μονάδα που βρίσκεται στα ηλεκτρονικά του NSW, και χρησιμοποιείται για τη διαμόρφωση των λειτουργιών όλων των άλλων μονάδων ASIC του συστήματος. Το SCAX από την άλλη, έχει σχεδιαστεί για να υποστηρίξει FPGA που είναι επίσης

μέρος του συστήματος ηλεκτρονικών του ATLAS, και βρίσκονται μακριά από περιοχές υψηλής ραδιενέργειας. Δίνει τη δυνατότητα στο χρήστη του να γράψει παραμέτρους διαμόρφωσης λειτουργιών στη λογική του FPGA που βρίσκεται, και να αναγνώσει τιμές κατάστασης από καταχωρητές του υπόλοιπου υλικολογισμικού. Το SCAX μιμείται το πρωτόκολλο I^2C που το SCA χρησιμοποιεί για να επικοινωνήσει με άλλες συσκευές, ενώ επίσης μιμείται και το πρωτόκολλο μεταξύ αυτού και του συστήματος FELIX. Με αυτόν τον τρόπο, επιτρέπει τη χρήση της ήδη υπάρχουσας υποδομής λογισμικού και ηλεκτρονικών, ώστε να διαμορφώσει τις λειτουργίες του FPGA μέσα στο οποίο έχει υλοποιηθεί. Το SCAX χρησιμοποιείται από τον επεξεργαστή σκανδαλισμού του NSW, και μπορεί να χρησιμοποιηθεί από οποιοδήποτε FPGA που επικοινωνεί με το FELIX.

Abstract

The Large Hadron Collider (LHC) at CERN, which began operations in 2008, has allowed the scientific community to conduct groundbreaking experiments that aim to answer fundamental questions about energy and matter. Since the high energy physics experiments rely heavily on statistics, the increase of reaction rate is considered to be of utmost importance. The higher the reaction rate, the more data are recorded, and rare events that would otherwise be hidden by other processes, emerge. For this reason, the LHC plans to perform upgrades in the way it delivers its beams, leading to a higher center-of-mass energy and instantaneous luminosity. After the second Long Shutdown (LS2) upgrade, which will be finalized in 2021, the beams will have an ultimate luminosity of $\mathscr{L} = 2.5 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$. Additional upgrades that will lead to the so-called Phase-II (2026-2038), will see the LHC deliver a luminosity of $5 - 7.5 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ and a center-of-mass energy of $14\,\mathrm{TeV}$. The increase in luminosity will lead to an increase in reaction rate and hence, particle flux, stressing the LHC's detectors and their associated data acquisition systems to their limits. For this reason, the Toroidal LHC ApparatuS (ATLAS) detector, the largest of the LHC, is planned to replace its innermost end-cap stations of its muon spectrometer during LS2. The New Small Wheel (NSW) upgrade as it is called, will be comprised of two detector technologies, the Micromegas (MM) and the small-strip Thin Gap Chambers (sTGC). Designed to cope with the increased background rate of future run conditions, the NSW will provide muon precision tracking data to ATLAS, and also contribute to its triggering scheme. In order to support the newly-installed detector media, a next-generation data acquisition system was designed, that will be able to cope with the increasing demand in throughput in the following years of ATLAS' operation, and endure the harsh radiation environment of the innermost end-cap region of the muon spectrometer. The cornerstone of the said front-end electronics system, is the VMM Application-Specific Integrated Circuit (ASIC), which will be used as the front-end

chip for both Micromegas and sTGC detectors of the ATLAS NSW upgrade. Due to its flexibility and several configurable parameters, it has also been proposed to a variety of tracking detectors in other experiments. The VMM integrates 64 independent channels, each providing precise amplitude and timing measurements in digital format for tracking purposes, and fast signals for triggering. This mixed-signal all-in-one solution, underwent three major revisions and a minor one, since its first prototype that was produced in 2012. None of these revisions could had been conducted, if a reliable testing and characterization platform did not exist. This gap was filled by the VMM Readout System (VRS), which employs Field-Programmable Gate Array (FPGA) devices that reside on the same board that the VMM is on. This FPGA is able to configure, read-out and calibrate the ASIC, and be adaptable to different implementation scenarios (i.e. testbench and testbeam use-cases), and different boards or FPGA packages. A large part of this dissertation, is devoted into describing this FPGA firmware that was developed for the needs of the NSW upgrade. The firmware was employed to validate the ASIC's performance, perform the mass-testing of its final production version, and handle the tracking and triggering data when the VMM was used in conjunction with the early Micromegas chamber prototypes during testbeam, thus aiding the collaboration in determining the operational parameters of the detector and its front-end ASIC. After the production of the final ASICs that would read-out and configure the VMM, the FPGA readout scheme was replaced by the final one, which made use of the next-generation data acquisition scheme of ATLAS, the Front-End LInk eXchange (FELIX). Another significant part of this work, describes the integration process of the NSW electronics system with FELIX and its related back-end infrastructure, which involved the development of software tools that eased the integration process, and the Micromegas detector commissioning prior to its deployment into the ATLAS cavern. Finally, the last Chapter of this dissertation, is devoted into describing the Slow Control Adapter eXtension (SCAX), which is an FPGA module that emulates the SCA ASIC. The latter is a radiation-tolerant device housed by all front-end boards of the NSW electronics system, and is used to access the register address space of other ASICs, in order to configure and monitor them. The SCAX on the other hand, is designed to support FPGA systems that are part of the ATLAS electronics scheme and are situated outside the radiation area, by writing into the configuration parameters or reading back any of the status registers of their logic. The SCAX emulates both the I^2C interface of the SCA ASIC used by the NSW devices, as well as the communication protocol implemented between itself and the back-end electronics scheme. It thereby enables using the same back-end software suite that support the ASICs, to also access the address space of the FPGAs within the same system. It is being used by the NSW Trigger Processor, and can be employed by any FPGA that interfaces with FELIX.

Acknowledgments

I would like to devote this section to anyone who has helped me in this long journey.

First and foremost, I would like to thank my thesis advisor, Theo Alexopoulos. He has been my supervisor for the last six years, since my undergraduate years, providing me with valuable insight, research opportunities, and support for my academic endeavors.

I would also like to thank Vinnie Polychronakos, for his invaluable support, throughout my tenure at CERN.

The help that Lorne Levinson provided me in many CERN-related projects, is also something that I will not forget.

Stephanie Zimmerman, has also supported me while being at CERN in various ways, and I would like to thank her as well.

Finally, I would like to thank Gianluigi De Geronimo, for the fruitful meetings and conversations we had on the VMM.

My colleagues in my group, should also of course not be left unmentioned: Paris, Stathis, George I., Kostas, (the other) Christos, Dimitris, Aimilios, Panos, Polyneikis, Maria, Marios, Chara, Yannis M., Prof. Maltezos, Theo G., and Andreas.

Finally, I would like to thank people outside of my group, whom I collaborated with during my four-year presence at CERN: Reid, Pia, Lev, Tiesheng, Zhen, Philipp, Riccardo, Alex T., Larry, Karri, Ann, Nathan, John, Thiago, Verena, Alex R., Enrique, Enrico, Ken, Kade, Garrett, Danny, Michael, Cenk, Serguei, Sorin, Stefan, Andrea, Henk, Frans, Jörn, Nikolina, Mark, Xu, Liang, Rongkun, Jorgen, Cristovao, Israel, Noam, Patrick, Vlad, and George Ch.

I also had the luck of becoming good friends with some of the above (I will not be specific, you know who you are!), and I am truly grateful for that.

I hope I did not forget any of my colleagues in this list. If anyone out there thinks they were left out by mistake, they should feel free to send me an angry e-mail.

Finally, I would like to thank my family and friends. I would not have been here without their support.

It is difficult to express my gratitude towards everyone mentioned here actually. I hope these lines were enough.

不闻不若闻之
闻之不若见之
见之不若知之
知之不若行之
学至于行之而止矣

I hear and I forget
I see and I remember
I do and I understand

Xunzi - Chinese Confucian Philosopher - 310-235 BC

Contents

Chapter I

Σύνοψη στα Ελληνικά

Στο παρών Κεφάλαιο, θα παρατεθεί μία σύνοψη της διατριβής, στην Ελληνική γλώσσα.

I.1 Ο Μεγάλος Επιταχυντής Αδρονίων

Γενικά, μπορεί να πει κανείς ότι οι επιταχυντές είναι τα ”μικροσκόπια” για τους Φυσικούς Υψηλών Ενεργειών [1]. Ιστορικά, ως πρώτος επιταχυντής μπορεί να θεωρηθεί ο καθοδικός σωλήνας που κατασκεύασε το 1895 ο Röntgen για να παράξει ακτίνες X, όπου μία διαφορά δυναμικού επιτάχυνε ηλεκτρόνια σε ενέργειες της τάξης των μερικών keV [1]. Σήμερα, οι ενέργειες που αγγίζουν οι επιταχυντές είναι της τάξης των TeV, και δεν επιταχύνουν μόνο ηλεκτρόνια, αλλά και πρωτόνια ή και βαρέα ιόντα. Στα πειράματα αυτά, μία δέσμη σωματιδίων επιταχύνεται είτε σε ευθεία, είτε σε κυκλική τροχιά με τη βοήθεια ηλεκτρομαγνητών που παράγουν πεδία αρκετά ισχυρά ώστε να αυξάνουν την ταχύτητα των σωματιδίων ενώ ταυτόχρονα τα κατευθύνουν σε αυστηρά καθορισμένες τροχιές. Τελικά, τα σωματίδια αυτά συγκρούονται είτε σε σταθερούς στόχους, είτε με άλλες δέσμες σωματιδίων, για να παράξουν βαρέα σωματίδια υψηλών ενεργειών, που στη συνέχεια μελετώνται από τους ερευνητές.

Ο Μεγάλος Επιταχυντής Αδρονίων (Large Hadron Collider, LHC), είναι ο μεγαλύτερος επιταχυντής που έχει κατασκευαστεί ποτέ, και ξεκίνησε επίσημα τη λειτουργία του το 2008. Πρόκειται για έναν κυκλικό επιταχυντή όπου υπέρλεπτες δέσμες πρωτονίων επιταχύνονται τόσο πολύ που πλησιάζουν την ταχύτητα του φωτός, ώσπου τελικά

[1] Υπενθυμίζεται ότι η μονάδα ενέργειας του eV *(ηλεκτρονιοβόλτ - electronvolt)* ισοδυναμεί με την ενέργεια που αποκτά ένα ηλεκτρόνιο, όταν αυτό επιταχυνθεί από μία διαφορά δυναμικού της τάξης του 1 V.

αλληλοσυγκρούονται μεταξύ τους παράγοντας έτσι σωματίδια προς μελέτη. Η πειραματική αυτή διάταξη βρίσκεται εκατοντάδες μέτρα κάτω από την επιφάνεια της Γης, στα σύνορα Γαλλίας-Ελβετίας, ενώ ο κυκλικός σωλήνας τον οποίο διατρέχουν τα πρωτόνια έχει περιφέρεια 27 km.

Figure I.1: Αεροφωτογραφία του CERN και του μεγάλου επιταχυντή αδρονίων. Είναι ευδιάκριτος ο μεγάλος δακτύλιος που επιταχύνονται τα σωματίδια, ενώ στα δεξιά φαίνεται το αεροδρόμιο και η λίμνη της Γενεύης. Ο μικρότερος δακτύλιος μέσα στον κύριο, είναι ο SPS, ο οποίος διοχετεύει τον κύριο δακτύλιο του μεγάλου επιταχυντή αδρονίων με πρωτόνια ενέργειας μερικών εκατοντάδων GeV.

Μία σειρά από όλο και μεγαλύτερους δακτυλίους επιταχύνουν προοδευτικά πρωτόνια που προέρχονται από μία απλή φιάλη υδρογόνου, τα οποία πρωτόνια θα καταλήξουν τελικά στον μεγάλο δακτύλιο του μεγάλου επιταχυντή αδρονίων, ο οποίος διαχωρίζει δύο δέσμες πρωτονίων και τις ανακυκλώνει για πολλές ώρες σε αντίθετες κατευθύνσεις μέσα του[2]. Οι δέσμες αυτές μπορούν να συγκρουσθούν μεταξύ τους σε συγκεκριμένα σημεία πάνω στον δακτύλιο, και οι ισχυρές αυτές συγκρούσεις απελευθερώνουν μεγάλα ποσά ενέργειας, δηλαδή βαριά σωματίδια, ή φωτόνια υψηλής συχνότητας, που παρέχουν πολύτιμες πληροφορίες για τη δομή της ύλης στους ερευνητές του CERN. Ακριβώς σε αυτά τα διαφορετικά σημεία των συγκρούσεων υπάρχουν οι *ανιχνευτές*, δηλαδή εξειδικευμένες διατάξεις κατάλληλης τεχνολογίας, που μπορούν να ανιχνεύσουν με ακρίβεια τα σωματίδια που παράγονται από τις συγκρούσεις, και να αναλύσουν πλήρως τις τροχιές και τις ιδιότητες των σωματιδίων αυτών. Οι κυριότεροι

[2]Η διαδικασία της επιτάχυνσης και ανακατεύθυνσης των δεσμών επιτυγχάνεται με ισχυρά ηλεκτρομαγνητικά πεδία που δημιουργούνται από ηλεκτρομαγνήτες υπεραγωγών που λειτουργούν μόνο σε θερμοκρασίες κοντά στο απόλυτο μηδέν.

ανιχνευτές στον LHC είναι οι: ATLAS, CMS, ALICE και LHCb[3].

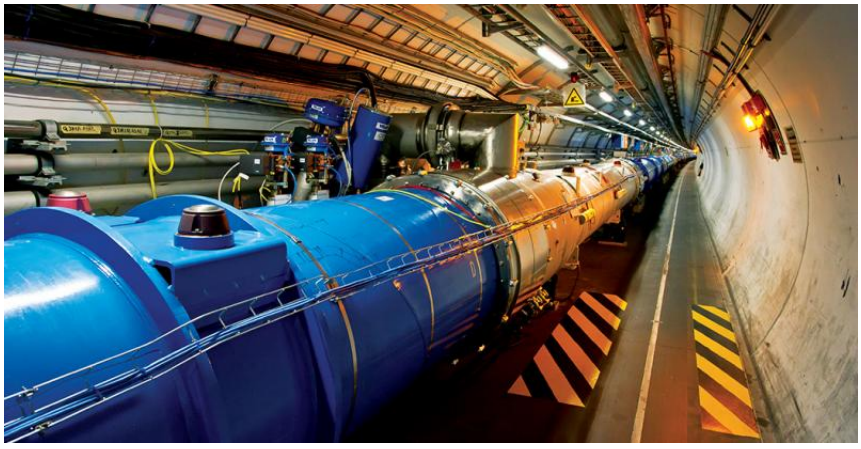


Figure I.2: Άποψη της σήραγγας του LHC όπου επιταχύνονται τα πρωτόνια.

Στην προσπάθεια βελτίωσης των πειραμάτων για εξαγωγή καλύτερων αποτελεσμάτων σε όλο και υψηλότερες ενέργειες, διαπιστώθηκε ότι όσο ψηλότερος ο ρυθμός αλληλεπιδράσεων ϕ, τόσο περισσότερα γεγονότα ανιχνεύονται, με αποτέλεσμα να εξάγονται και περισσότερες πληροφορίες από το πείραμα. Ξεκινώντας από τα απλά, δηλαδή από τα πειράματα σταθερού στόχου, ο ρυθμός αλληλεπίδρασης εξαρτάται από το ρυθμό πρόσπτωσης n των σωματιδίων της δέσμης πάνω στο στόχο, την ενεργό διατομή σ της υπό μελέτη αντίδρασης[4], και το πάχος d του στόχου [1]. Συνδυάζοντας τα μεγέθη αυτά, ορίζεται το μέγεθος της *Φωτεινότητας* $\mathscr{L}$:

$$\phi = \sigma \mathscr{L} \tag{I.1}$$

Στα σύγχρονα πειράματα Υψηλών Ενεργειών όπου δεν προτιμάται η μέθοδος σταθερών στόχων, αφού με αλληλοσυγκρουόμενες δέσμες η μέγιστη ενέργεια που επιτυγχάνεται είναι σημαντικά μεγαλύτερη, η φωτεινότητα (και άρα ο ρυθμός αλληλεπιδράσεων που είναι ανάλογος της φωτεινότητας) ορίζεται περισσότερο περίπλοκα. Πλέον η φωτεινότητα αντιπροσωπεύει ένα μέτρο του πλήθους των σωματιδίων ανά μονάδα εμβαδού επιφανείας και χρόνου. Αν N_1 και N_2 το πλήθος των σωματιδίων της κάθε δέσμης, και σ_x, σ_y η εγκάρσια και διαμήκης διατομή των δεσμών , τότε η φωτεινότητα $\mathscr{L}$ σχετίζεται με τις παραμέτρους αυτές μέσω του τύπου:

[3] Οι ανιχνευτές γενικού ενδιαφέροντος, ATLAS και CMS, ήταν εκείνοι που ανίχνευσαν το 2012 το μποζόνιο Higgs.

[4] Η ενεργός διατομή έχει μονάδες επιφάνειας και εκφράζει ουσιαστικά την πιθανότητα πραγμάτωσης μίας αντίδρασης.

$$\mathscr{L} \propto \frac{N_1 N_2}{\sigma_x \sigma_y} \tag{I.2}$$

Έτσι λοιπόν, το πλήθος των αλληλεπιδράσεων N_{exp} που αναμένεται στο πείραμα, ορίζεται από τη σχέση [2]:

$$N_{exp} = \sigma_{exp} \int \mathscr{L}(t) dt \tag{I.3}$$

Όπου η ποσότητα σ_{exp} ονομάζεται *ενεργός διατομή*, έχει μονάδες μεγέθους εμβαδού επιφάνειας και εκφράζει την πιθανότητα να συμβεί μία συγκεκριμένη αντίδραση.

Σχετικά με τους τύπους που μόλις αναφέρθηκαν και τη σχέση τους με τα πειράματα, αξίζει να αναφερθεί ότι περισσότερη έμφαση δίνεται στη μείωση της διατομής των δεσμών, ένα εγχείρημα που χαρακτηρίζεται από εγγενείς περιορισμούς, καθώς μία δέσμη που αποτελείται μόνο από πρωτόνια για παράδειγμα, αδυνατεί να μείνει συγκεντρωμένη, λόγω των ισχυρών ηλεκτρικών δυνάμεων άπωσης μεταξύ των όμοια φορτισμένων πρωτονίων, που βρίσκονται τόσο κοντά μεταξύ τους μέσα στη δέσμη. Τέτοια τεχνικά ζητήματα καλούνται να λυθούν σε κάθε πείραμα Φυσικής Υψηλών Ενεργειών, πόσο μάλλον στον μεγάλο επιταχυντή αδρονίων, όπου οι ενέργειες και η φωτεινότητα είναι τάξεις μεγέθους μεγαλύτερες σε σχέση με παλιότερα εγχειρήματα.

Πιο συγκεκριμένα [3, 4], οι ανιχνευτές ATLAS και CMS, είναι σχεδιασμένοι για να ανιχνεύουν σε φωτεινότητες της τάξης του $\mathscr{L} = 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, ενώ το πείραμα LHCb λειτουργεί σε χαμηλότερες φωτεινότητες ($\mathscr{L} = 10^{32}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$). Από τη κατασκευή του μέχρι σήμερα, ο LHC παραμένει (και θα παραμείνει για πολλά χρόνια) η μεγαλύτερη και πιο περίπλοκη πειραματική διάταξη που κατασκευάστηκε ποτέ. Ξεκίνησε το 2011-2012 στα $3.5\,\mathrm{TeV}$ ενέργεια ανά δέσμη και ανέβηκε στα $4\,\mathrm{TeV}$ το 2012, ενώ έχει προσφέρει συνολική φωτεινότητα της τάξης $\int \mathscr{L} dt = 28.26\,\mathrm{fb}^{-1}$ στους ανιχνευτές ATLAS και CMS. Στη συνέχεια, θα εξεταστεί η δομή του ανιχνευτή ATLAS, ο οποίος είναι και ο ανιχνευτής στον οποίο αναφέρεται η παρούσα εργασία, καθώς οι αναβαθμίσεις που σχεδιάζονται για το μέλλον χρήζουν νέων μελετών στους ανιχνευτές και στα ηλεκτρονικά τους συστήματα, προκειμένου να αντεπεξέλθουν στις νέες απαιτήσεις των επικείμενων πειραμάτων.

I.2 Ο Ανιχνευτής ATLAS

Ο ανιχνευτής ATLAS (**A** **T**oroidal **L**HC **A**pparatu**S**), είναι ένας ανιχνευτής γενικής χρήσης, σχεδιασμένος να καλύψει τις ανάγκες πολλών πειραμάτων του μεγάλου επιταχυντή αδρονίων, όπως την αναζήτηση του σωματιδίου Higgs, τη μελέτη της υπερσυμμετρίας, την απάντηση ερωτημάτων σχετικά με τη σκοτεινή ύλη και τυχόν ύπαρξη επιπλέον διαστάσεων στο Σύμπαν. Το σχήμα του έχει κυλινδρική μορφή, με μήκος διαμήκη άξονα περίπου 45 m, και διάμετρο περί τα 25 m. Ζυγίζει περίπου 7000 τόννους, και είναι μέχρι σήμερα ο μεγαλύτερος ανιχνευτής σε όγκο που κατασκευάστηκε ποτέ. Από το 2012, 3000 επιστήμονες από 38 χώρες συνεργάζονται για την υποστήριξη και αναβάθμιση του ανιχνευτή. Στο Σχ. I.3 απεικονίζεται σε τρισδιάστατο μοντέλο μία γενική άποψη του ανιχνευτή.

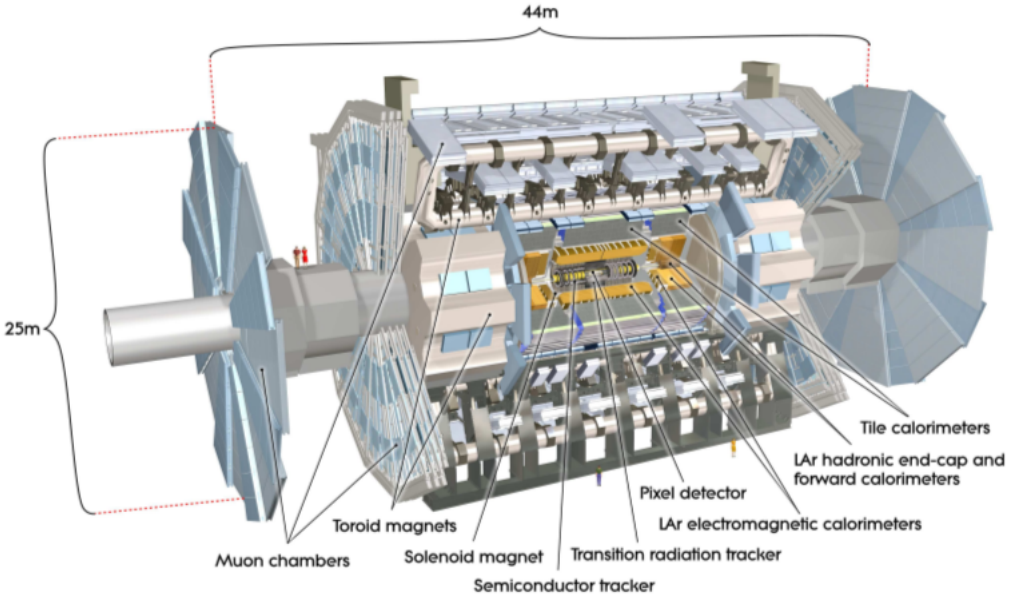


Figure I.3: Τομή του ανιχνευτή ATLAS.

Οι δέσμες των σωματιδίων περνούν από τον νοητό άξονα κυλινδρικής συμμετρίας του ανιχνευτή (από το κέντρο των κυκλικών ”καπακιών”) και συγκρούονται στο κέντρο του, παράγοντας νέα σωματίδια. Τα διαφορετικά υποσυστήματα του ATLAS, τοποθετημένα σε στρώματα, καταγράφουν την τροχιά, την ορμή και την ενέργεια των σωματιδίων, ταυτοποιώντας τα πλήρως. Ισχυροί μαγνήτες κάμπτουν τις τροχιές των φορτισμένων σωματιδίων, επιτρέποντας τη μέτρηση της ορμής τους, ενώ τα δεδομένα που συλλέγονται από τα ηλεκτρονικά συστήματα του ανιχνευτή, υποβάλλονται σε συνεχή ανάλυση ώστε να εξαχθούν οι επιθυμητές πληροφορίες από αυτά. Η όλη διάταξη θυμίζει τη μορφή ενός βαρελιού, και για το λόγο αυτό είναι σύνηθες να

αναφέρονται τα διάφορα τμήματα του ανιχνευτή ως ”περίβλημα του βαρελιού” (ή απλά ”βαρέλι”), και ως ”καπάκια”.

Προκειμένου να γίνουν σαφείς οι λεπτομέρειες για τα ανιχνευτικά συστήματα του ATLAS που θα αναφερθούν στη συνέχεια, είναι καλό σε αυτό το σημείο να οριστεί η έννοια της *ψευδοωκύτητας* (η). Αν η γωνία σε σχέση με τον άξονα της δέσμης ισούται με θ, τότε η ψευδοωκύτητα ορίζεται ως [5]:

$$\eta \equiv -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \tag{I.4}$$

Ο ορισμός της ποσότητας η είναι πολύ σημαντικός, καθώς γίνεται έτσι εύκολα αντιληπτή η γεωμετρία που καλύπτει το κάθε ανιχνευτικό υποσύστημα του ανιχνευτή. Πιο συγκεκριμένα, αναφέρεται πιο συχνά η ποσότητα $|\eta|$, που καλύπτει και τα δύο νοητά ημιεπίπεδα που σχηματίζει η ευθεία της δέσμης. Ένα προκύπτον σωματίδιο με μεγάλη τιμή ψευδοωκύτητας, θα κινείται σχεδόν συγγραμικά με τη δέσμη, ενώ μικρότερες τιμές του $|\eta|$ υπονοούν γωνίες διαφυγής σε σχέση με τη δέσμη της τάξης των $90° - 45°$. Έτσι, είναι φανερό ότι τα καπάκια του ανιχνευτή αντιστοιχούν σε μεγάλες τιμές του $|\eta|$, ενώ το κυλινδρικό περίβλημα του βαρελιού σε μικρότερες.

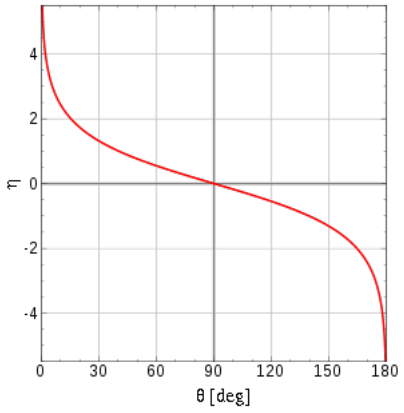


Figure I.4: Γράφημα της ψευδοωκύτητας σε σχέση με τη γωνία από τον άξονα της δέσμης.

Τα κυριότερα υποσυστήματα του ATLAS είναι τα εξής [3, 4]:

- Το σύστημα μαγνητών
- Ο εσωτερικός ανιχνευτής (tracker)
- Τα θερμιδόμετρα (ή καλορίμετρα)

- Το μιονικό φασματόμετρο (ή φασματόμετρο)

Ο μαγνήτης είναι ουσιαστικά ένα λεπτό υπεραγώγιμο σωληνοειδές, που περικλείει το κενό ανάμεσα από το σημείο αλληλεπίδρασης/σύγκρουσης σε σχέση με τον εσωτερικό ανιχνευτή. Εκτός από τον εσωτερικό μαγνήτη, τρία μεγαλύτερα σωληνοειδή βρίσκονται τοποθετημένα γύρω από τα καλορίμετρα. Ο εσωτερικός μαγνήτης παράγει ένα μαγνητικό πεδίο της τάξης των $2\,\mathrm{T}$, και επιτρέπει στον εσωτερικό ανιχνευτή (με μήκος $6\,\mathrm{m}$ και διάμετρο $2\,\mathrm{m}$) που καλύπτει το εύρος $|\eta| < 2.5$, να αναλύσει τις τροχιές των σωματιδίων που μόλις έχουν δραπετεύσει από τα σημεία σύγκρουσης. Περίπου 1000 σωματίδια προκύπτουν από το σημείο σύγκρουσης κάθε $25\,\mathrm{ns}$. Παρά τη μεγάλη πυκνότητα τροχιών που δημιουργούνται στον ανιχνευτή, επιτυγχάνεται βέλτιστος προσδιορισμός ορμών και vertex[5], από τα συστήματα των Pixel Detectors και SemiConductor Trackers (SCTs). Γύρω από αυτούς, βρίσκονται Transition Radiation Trackers (TRT), που αναλύουν την ακτινοβολία μετάπτωσης[6].

Γύρω από τον εσωτερικό ανιχνευτή, βρίσκονται τα καλορίμετρα, ή αλλιώς θερμιδόμετρα. Τα καλορίμετρα είναι συσκευές σχεδιασμένες με τέτοιο τρόπο ώστε να απορροφούν πλήρως συγκεκριμένου τύπου σωματίδια που προσπίπτουν πάνω τους. Έτσι όλη η ενέργεια των σωματιδίων εναποτίθεται στο εσωτερικού του θερμιδόμετρου. Συνήθως, τα καλορίμετρα είναι είτε αδρονικά (ανιχνεύουν δηλαδή μεσόνια ή βαρυόνια), είτε ηλεκτρομαγνητικά (ανιχνεύουν ηλεκτρόνια, ποζιτρόνια και φωτόνια). Το αδρονικό καλορίμετρο χωρίζεται σε τρία μέρη, ανάλογα το εύρος της ψευδοωκύτητας που καλύπτει. Υπάρχει το τμήμα που καλύπτει το βαρέλι ($|\eta| < 1.7$), τα καπάκια του ($1.5 < |\eta| < 3.2$), και περιοχές ακόμα περισσότερο κοντά στη δέσμη ($3.1 < |\eta| < 3.9$) [3, 4]. Κάθε τμήμα έχει και λίγο διαφορετική κατασκευή: για παράδειγμα το καλορίμετρο που περιβάλλει το βαρέλι είναι φτιαγμένο από ατσάλι (που λειτουργεί ως απορροφητής) και σπινθηριστές που παράγουν τα σήματα ανίχνευσης [5]. Τα υπόλοιπα αδρονικά καλορίμετρα είναι κατασκευασμένα από υγρό Αργό (Liquid-Argon (LAr)). Το ηλεκτρομαγνητικό καλορίμετρο από την άλλη, αποτελείται από δύο ομόκεντρους κυκλικούς δίσκους στα καπάκια, έναν εξωτερικό και έναν εσωτερικό που καλύπτουν $1.375 < |\eta| < 2.5$ και $2.5 < |\eta| < 3.2$ αντίστοιχα. Το ηλεκτρομαγνητικό καλορίμετρο περικλείει το βαρέλι και καλύπτει εύρος $|\eta| < 1.475$. Οι συσκευές αυτές είναι κατασκευασμένες από μολύβδινες πλάκες που απορροφούν τα σωματίδια, από LAr, και από ηλεκτρόδια τύπου Kapton σε σχήμα ακκορντεόν υπεύθυνα για τη συλλογή των ηλεκτρικών σημάτων [3–5].

Τα σωματίδια που δεν έχουν απορροφηθεί από τα δύο μεγάλα στρώματα ανιχνευτών που αναφέρθηκαν παραπάνω, είναι συνήθως μιόνια (μ), τα οποία ανήκουν στην

[5] Το "σημείο" αλληλεπίδρασης ονομάζεται vertex.

[6] Η ακτινοβολία που εκπέμπεται από ένα σωματίδιο όταν αυτό αλλάζει μέσα διάδοσης τα οποία έχουν διαφορετικές διηλεκτρικές ιδιότητες μεταξύ τους [1].

οικογένεια των λεπτονίων δεύτερης γενιάς. Είναι είτε θετικά είτε αρνητικά φορτισμένα (τα αντιμιόνια είναι θετικά φορτισμένα) και είναι αρκετά βαρύτερα από τα συγγενικά τους ηλεκτρόνια[7]. Λόγω της διεισδυτικότητας που τα χαρακτηρίζει, γενικά σε ανιχνευτικές διατάξεις, τα μέρη εκείνα που είναι υπεύθυνα για την ταυτοποίησή τους, τοποθετούνται στα εξωτερικά τμήματα του ανιχνευτή.

Το *Μιονικό Φασματόμετρο* (στο οποίο θα δοθεί και περισσότερη έμφαση καθώς η αναβάθμιση την οποία πραγματεύεται το παρών μέρος αυτής της εργασίας αφορά αυτό το κομμάτι του ATLAS) περιβάλλει τα καλορίμετρα και ορίζει ουσιαστικά το μέγεθος ολόκληρου του ανιχνευτή. Τα συστήματα του μιονικού φασματόμετρου, περιλαμβάνουν ανώτερες τεχνολογίες για τον σκανδαλισμό[8] και για την ανακατασκευή των τροχιών των μιονίων [3, 4]. Στο Σχ. I.5 είναι φανερή η διάταξη του συστήματος εντοπισμού των μιονίων του ATLAS.

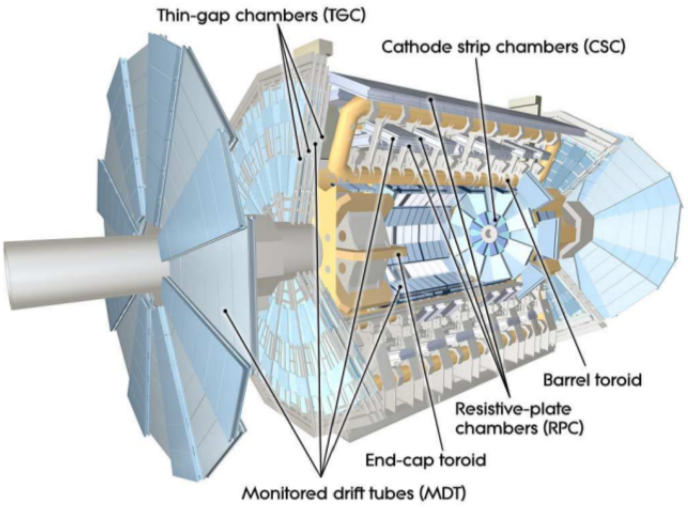


Figure I.5: Άποψη του μιονικού φασματομέτρου του ATLAS [5].

Η λειτουργία του φασματομέτρου μιονίων βασίζεται στην κάμψη των τροχιών των μιονίων λόγω του μαγνητικού πεδίου που δημιουργούν οι υπεραγώγιμοι τοροειδείς μαγνήτες μέσα στον ανιχνευτή. Στο εύρος $|\eta| < 1.4$, η κάμψη των τροχιών παρέχεται από τον τοροειδή μαγνήτη που περιβάλλει το βαρέλι, ενώ για το εύρος $(1.6 < |\eta| < 2.7)$ οι τροχιές κάμπτονται από δύο τοροειδείς μαγνήτες που βρίσκονται στα καπάκια του ανιχνευτή. Το προκύπτον μαγνητικό πεδίο είναι ως επί το πλείστον κάθετο στις τροχιές των μιονίων [5], όποια και να είναι η γωνία διαφυγής τους σε σχέση με τη

[7] Τα μιόνια (μ^-) έχουν μάζα περίπου $106\,\mathrm{MeV/c^2}$, ενώ τα ηλεκτρόνια (e^-) $511\,\mathrm{keV/c^2}$ [6].

[8] "Triggering" στα Αγγλικά. Πρόκειται για ένα σήμα που υποδεικνύει ότι ένα πραγματικό γεγονός έχει συμβεί και πρέπει να καταγραφεί από τα ηλεκτρονικά συστήματα του ανιχνευτή.

δέσμη. Η ανίχνευση στα μικρά $|\eta|$, γίνεται από τρεις θαλάμους ανίχνευσης, που είναι ταξινομημένοι σε τρία κυλινδρικά στρώματα γύρω από τον άξονα της δέσμης. Στην περιοχή με μεγάλα $|\eta|$, οι θάλαμοι καλύπτουν τα καπάκια πλήρως, είναι κάθετοι στον άξονα της δέσμης, και είναι και αυτοί οργανωμένοι σε τρία στρώματα.

Σε όλα τα εύρη των $|\eta|$, η τροχιά των μιονίων ανακατασκευάζεται από τα σήματα που αφήνουν τα μιόνια σε μία σειρά από Monitored Drift Tubes (MDTs). Στα καπάκια, εκτός από MDTs, υπάρχουν και Cathode Strip Chambers (CSCs), που είναι πολυσυρματικοί αναλογικοί ανιχνευτές (multiwire proportional chambers), σχεδιασμένοι να υποβοηθούν την ανακατασκευή τροχιών των μιονίων, καθώς η περιοχή εκείνη χαρακτηρίζεται από αυξημένο ρυθμό γεγονότων και ακτινοβολία υποβάθρου [5]. Η αρχή λειτουργίας τους είναι συγγενική με εκείνη των MDTs, όμως χαρακτηρίζεται από εντελώς διαφορετική γεωμετρία και μείγμα αερίου. Το σύστημα του σκανδαλισμού από την άλλη, αποτελείται από Resistive Plate Chambers (RPCs) που περιβάλλουν το βαρέλι, και από Thin Gap Chambers (TGCs), στα καπάκια.

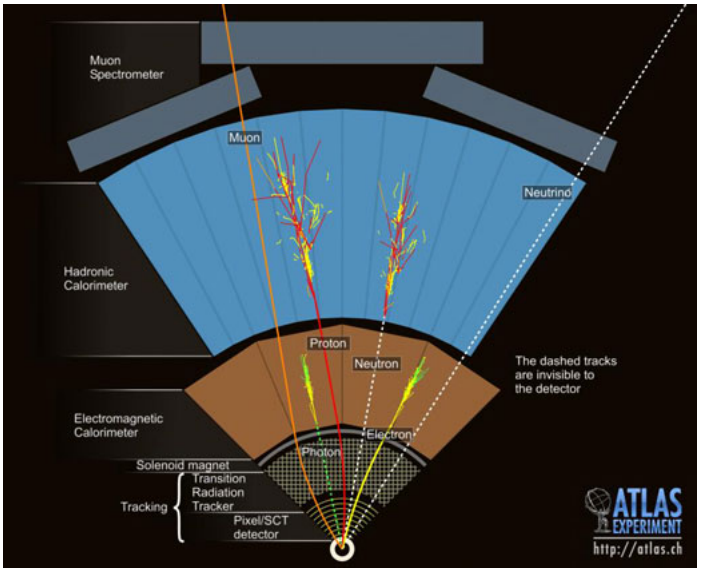


Figure I.6: Αναπαράσταση ενός γεγονότος στον ATLAS [3, 4].

I.3 Η Αναβάθμιση του New Small Wheel

Η κατασκευή του LHC ολοκληρώθηκε με επιτυχία το 2008 και η μέγιστη ενέργεια του επιταχυντή έφτασε περίπου τα 1 TeV, και μέχρι το πρώτο κλείσιμο το 2010, προσέφερε συνολικά $\int \mathscr{L} = 29\,\text{fb}^{-1}$. Μετά τη πρώτη μεγάλη παύση λειτουργίας (Long Shutdown 1, LS1) το 2013-2014 για αναβαθμίσεις, η ενέργεια του επιταχυντή αυξήθηκε στα

7 TeV ανά δέσμη, με τη φωτεινότητα να φτάνει στα $\mathscr{L} = 1 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$. Το 2020 έχει προγραμματιστεί η δεύτερη μεγάλη παύση (LS2), όπου η φωτεινότητα θα αυξηθεί ακόμα περισσότερο ($\mathscr{L} = 2-3\times10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$), η ενέργεια θα φτάσει τα $13-14$ TeV, και ο ανιχνευτής ATLAS αναμένεται να συλλέγει περίπου $100\,\mathrm{fb}^{-1}$ ολοκληρωμένη φωτεινότητα ανά έτος [7]. Το τελικό στάδιο της αναβάθμισης της δέσμης αναμένεται το 2022, (LS3), όπου η φωτεινότητα θα αγγίξει τα $\mathscr{L} = 5 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$. Στο Σχ. I.7 δίνεται ένα χρονοδιάγραμμα των αναβαθμίσεων στον LHC.

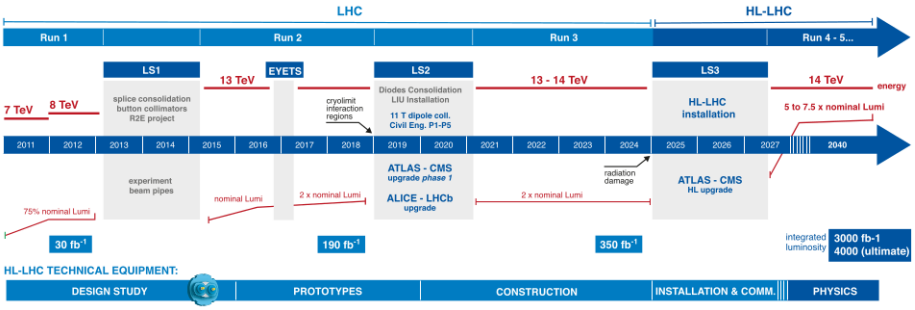


Figure I.7: Χρονοδιάγραμμα των αναβαθμίσεων του LHC [4].

Για τον ανιχνευτή ATLAS, οι διαδοχικές αυξήσεις στη φωτεινότητα συνεπάγονται και αύξηση στο ρυθμό με τον οποίο τα σωματίδια θα τον διαπερνούν. Προκειμένου να εξαχθούν όλα τα επιθυμητά αποτελέσματα από την επικείμενη αναβάθμιση του LHC το 2018, ο ανιχνευτής ATLAS πρέπει να αναβαθμιστεί και αυτός. Πιο συγκεκριμένα, ιδιαίτερη προσοχή έχει δοθεί στη βελτίωση του πρώτου επιπέδου σκανδαλισμού και της ανακατασκευής τροχιών στο σύστημα μιονίων, καθώς [7]:

- Η απόδοση των ανιχνευτών που είναι υπεύθυνοι για την ανακατασκευή των τροχιών των μιονίων, κυρίως σε τμήματα με μεγάλα $|\eta|$ στα καπάκια του ανιχνευτή, προβλέπεται ότι θα υποβαθμιστεί με την αύξηση της φωτεινότητας, βάσει των μέχρι τώρα μετρήσεων. Μεγαλύτερο πρόβλημα θα παρουσιαστεί κυρίως στο εσωτερικό τμήμα των δίσκων/τροχών (Small Wheels), που έχουν απόσταση μέχρι 7 m από τον άξονα της δέσμης.

- Αναλύσεις των δεδομένων που πραγματοποιήθηκαν το 2012 και συγκέντρωσαν τα δεδομένα του σκανδαλισμού από την έναρξη λειτουργίας του LHC, κατέδειξε ότι περίπου το $90\,\%$ των μιονικών στοιχείων σκανδαλισμού στα καπάκια του ATLAS ήταν ψεύτικα. Αυτό συνέβαινε καθώς πρωτόνια χαμηλής ενέργειας, που παράγονταν από δευτερεύουσες αντιδράσεις στα υλικά του ανιχνευτή πριν το μιονικό φασματόμετρο, στα καπάκια, εισέρχονταν στους ανιχνευτές TGCs (που είναι υπεύθυνοι για το μιονικό σκανδαλισμό), με τέτοιες γωνίες και ορμές που αναγνωρίζονταν από αυτούς ως μιόνια, φέροντας έτσι το σύστημα σε σύγχυση.

Με την αύξηση της ροής των σωματιδίων (περίπου $15\,\mathrm{kHz/cm^2}$ για $|\eta| = 2.7$ [3, 4]) μετά την πρώτη αναβάθμιση, τα προαναφερθέντα προβλήματα αναμένεται να οξυνθούν σημαντικά. Ως εκ τούτου, η ομάδα του ATLAS πρότεινε την πλήρη αντικατάσταση του προβληματικού τμήματος στα καπάκια (του εσωτερικού δίσκου με μεγάλα $|\eta|$ που είναι πιο κοντά στη δέσμη), από ένα νέο ανιχνευτικό σύστημα, το *New Small Wheel* (NSW). Το νέο τμήμα θα καλύπτει ένα εύρος $1.3 < |\eta| < 2.7$, και με τους νέους ανιχνευτές που θα είναι εγκατεστημένοι πάνω του, προβλέπεται ότι θα προσφέρει άριστο χωρικό και χρονικό προσδιορισμό των τροχιών σε πραγματικό χρόνο, καθώς και πιο συνεπές σκανδαλισμό πρώτου επιπέδου, αφού θα μπορεί πλέον να διακρίνει τα μιόνια χαμηλής ενέργειας από τα πρωτόνια που παράγονται από δευτερεύουσες αντιδράσεις και αποτελούν για το σύστημα μιονίων ακτινοβολία υποβάθρου [3,4]. Ένα νέο σύστημα σκανδαλισμού που θα συνδυάζει δεδομένα από το ηλεκτρομαγνητικό καλορίμετρο μαζί με δεδομένα από το NSW, θα μειώσει την αναμενόμενη (για $\mathscr{L} = 3 \times 10^{34}\,\mathrm{cm^{-2}s^{-1}}$) συχνότητα μιονικών σημάτων σκανδαλισμού πρώτου επιπέδου, από την προβλεπόμενη τιμή των $100\,\mathrm{kHz}$, που θα επικρατούσε αν η ανιχνευτική διάταξη παρέμενε ως έχει, σε πιο διαχειρίσιμα επίπεδα της τάξης των $20\,\mathrm{kHz}$, με την εγκατάσταση του NSW [3, 4].

Οι ανιχνευτές που θα αποτελούν το NSW θα είναι δύο ειδών και θα προέρχονται από την γενικότερη κατηγορία των ανιχνευτών αερίου. Ο πρώτος είναι ένας *Πολυσυρματικός Θάλαμος (Multiwire Chamber)*, που ονομάζεται *small strip Thin Gap Chamber (sTGC)*, ενώ ο δεύτερος ανήκει στην οικογένεια των *Micro-Pattern Gaseous Detectors*, και ονομάζεται *Micromesh Gaseous Structure (Micromegas)*. Οι sTGC θα χρησιμοποιούνται για τον σκανδαλισμό, ενώ οι ανιχνευτές Micromegas (MM) θα χρησιμεύσουν κυρίως στην ανακατασκευή τροχιών, αλλά θα συμμετέχουν και στο σκανδαλισμό. Οι δύο αυτοί τύπου ανιχνευτών θα καλύπτουν συνολικά μία έκταση $1200\,\mathrm{m^2}$, και θα καταλαμβάνουν όλη την εσωτερική περιοχή στα καπάκια του ανιχνευτή. Θα είναι τοποθετημένοι ακτινικά σε οκτώ πλήρως αλληλοκαλυπτόμενα επίπεδα, όπου οι MM θα καταλαμβάνουν το εσωτερικό του δίσκου, ενώ οι sTGC θα βρίσκονται στα εξωτερικά μέρη. Οι ανιχνευτές θα είναι τοποθετημένοι με τέτοιο τρόπο ώστε να μην υπάρχουν νεκρές ζώνες[9], αλλά μόνο περιοχές ελαττωμένης ανιχνευτικής απόδοσης. Μία γενική άποψη από τη μορφή του NSW μπορεί να δει κανείς στο Σχ. I.8.

[9]Πρόκειται για περιοχές που δεν καλύπτονται από ανιχνευτές και αν περάσει κάποιο σωματίδιο από εκεί, θα μείνει απαρατήρητο.

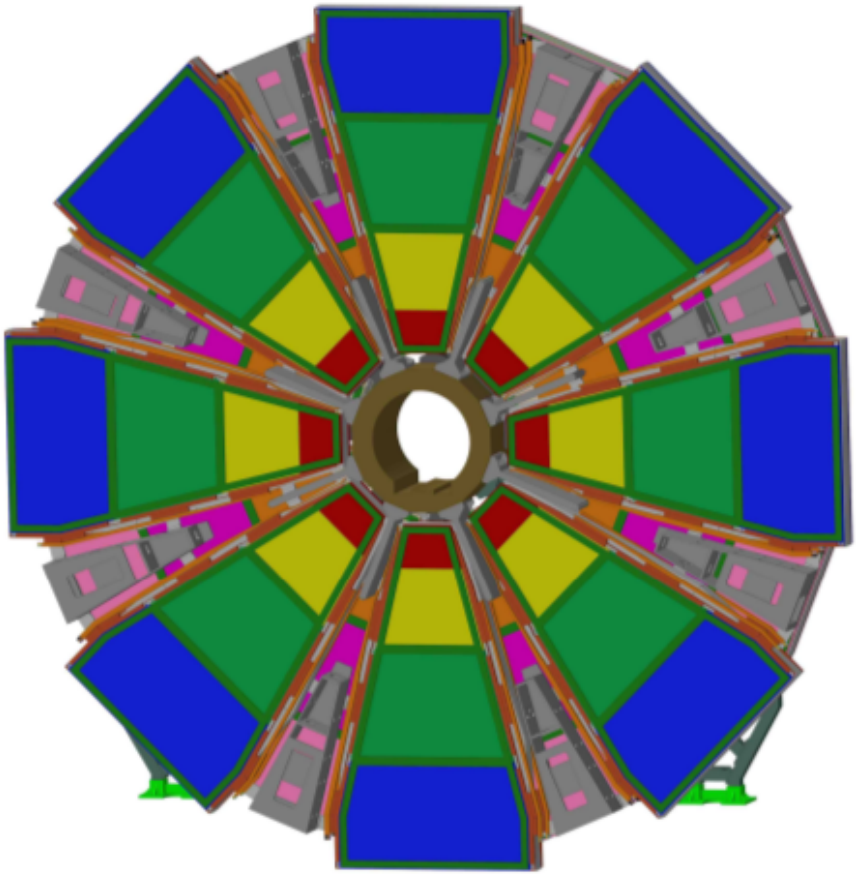


Figure I.8: Σχηματική αναπαράσταση του NSW. Οι ανιχνευτές MM και sTGC είναι διατεταγμένοι σε αλλεπάλληλα στρώματα σε κάθε ”φέτα” (wedge) [3, 4].

I.4 Τα Ηλεκτρονικά Συστήματα του New Small Wheel

Σε αυτή την ενότητα, θα μελετηθεί η γενικότερη δομή που διέπει ολόκληρο το σύστημα ηλεκτρονικών που θα υποστηρίξει την αναβάθμιση του New Small Wheel (NSW). Μέσω παρατηρήσεων και υπολογισμών με τα μέχρι τώρα δεδομένα του ATLAS, έχει προβλεφθεί ότι μετά την αναβάθμιση του LHC, που συνεπάγεται αύξηση της φωτεινότητας άρα και της ροής των σωματιδίων στους ανιχνευτές, η γενική απόδοση του μιονικού φασματόμετρου στα καπάκια του ανιχνευτή θα εκφυλιστεί. Πιο συγκεκριμένα, αναμένεται ότι με την αύξηση του ρυθμού αλληλεπιδράσεων λόγω των διαδοχικών αναβαθμίσεων, η χωρική διακριτική ικανότητα του παρόντος ανιχνευτή θα υποβαθμιστεί, ενώ το σύστημα του μιονικού σκανδαλισμού, που ήδη παρουσιάζει προβλήματα, σίγουρα θα κριθεί ως παραπάνω από ανεπαρκές. Επομένως,

με την αναβάθμιση των ανιχνευτών του NSW, απαιτείται και ένα εξελιγμένο σύστημα ηλεκτρονικών το οποίο σε γενικές γραμμές οφείλει:

- Να συλλέγει τα δεδομένα του σκανδαλισμού και να τα στέλνει στο τμήμα επεξεργασίας σημάτων σκανδαλισμού του CERN γρήγορα και αποδοτικά.
- Να αποθηκεύει την ενέργεια που εναποθέτουν τα σωματίδια στο σύστημα ανίχνευσης, μαζί με το *πότε* συνέβη αυτό, με ικανοποιητική ακρίβεια, προκειμένου να βελτιστοποιηθεί η ανακατασκευή των τροχιών.

Έτσι ορίζονται οι βασικοί άξονες λειτουργίας του συστήματος ηλεκτρονικών που βρίσκεται πάνω στους ανιχνευτές *(Εμπρόσθια Ηλεκτρονικά - Front-end Electronics, FE)*: γρήγορη συλλογή δεδομένων για το σκανδαλισμό, ακριβής μέτρηση ενέργειας και χρόνου, και διανομή των δεδομένων αυτών στο σύστημα ηλεκτρονικών που παρεμβάλλεται μεταξύ του ανιχνευτή και του κέντρου υπολογιστών του CERN *(Όπισθεν Ηλεκτρονικά - Back-end Electronics, BE)*. Επιπλέον, θα πρέπει να λαμβάνουν εντολές διαμόρφωσης της λειτουργίας τους από το κέντρο ελέγχου του ATLAS, και να στέλνουν πίσω δεδομένα μετρήσεων εξωτερικών παραγόντων (π.χ. θερμοκρασία) που επικρατούν στους ανιχνευτές *(configuration and monitoring)*.

Ως προς τη διαδικασία του σκανδαλισμού, το νέο σύστημα που θα παράγει τα εν λόγω σήματα του NSW, σε συνεργασία με το σύστημα του μεγάλου τροχού, πρέπει να λειτουργεί με τέτοιο τρόπο ώστε να αποκλείει τα ψεύτικα σήματα σκανδαλισμού που προέρχονται από τροχιές που δε συμβαδίζουν με το σημείο αλληλεπίδρασης στο κέντρο του ανιχνευτή [7]. Η όλη διαδικασία απεικονίζεται στο Σχ. I.9:

Η διαδικασία επεξεργασίας των σημάτων σκανδαλισμού περιλαμβάνει συλλογή δεδομένων για τα σημεία αλληλεπίδρασης και υπολογισμός των γωνιών (αζιμουθιακή ϕ και πολική $\Delta\theta$) που έχουν οι τροχιές σε σχέση με το σύστημα συντεταγμένων των ανιχνευτών (βλ. Σχ. I.9). Αυτές οι γωνίες, υπολογίζονται τόσο από τους sTGC όσο και από τους MM, οι οποίοι μέσω των λωρίδων διαβάσματός τους που έχουν πλάτος μόλις $0.5\,\text{mm}$, μπορούν να παρέχουν άμεσες και ακριβείς πληροφορίες που οδηγούν στον υπολογισμό γωνιών (με ακρίβεια < 1 mrad) των τροχιών. Η διαδικασία υπολογισμού βέβαια, πρέπει να ολοκληρώνεται μέσα σε χρόνο $1025\,\text{ns}$ από τη στιγμή που εντοπίζεται η πρώτη αλληλεπίδραση, ώστε να προλαμβάνει τα δεδομένα που θα προκύψουν από την ανάλυση του μεγάλου τροχού [7]. Με το συνδυασμό των πληροφοριών για τις τροχιές και από τα δύο συστήματα, το μιονικό σύστημα σκανδαλισμού θα μπορεί πλέον αν ανταπεξέρχεται στις περιστάσεις. Πέρα από τη βελτίωση του σκανδαλισμού, οι νέες τεχνολογίες των ανιχνευτών που θα χρησιμοποιηθούν στο NSW, αυξάνουν τη χωρική και διακριτική ικανότητα και βελτιστοποιούν τις μετρήσεις για την ενέργεια που εναποθέτουν τα σωματίδια στους ανιχνευτές. Λόγω αυτού του γεγονότος, η

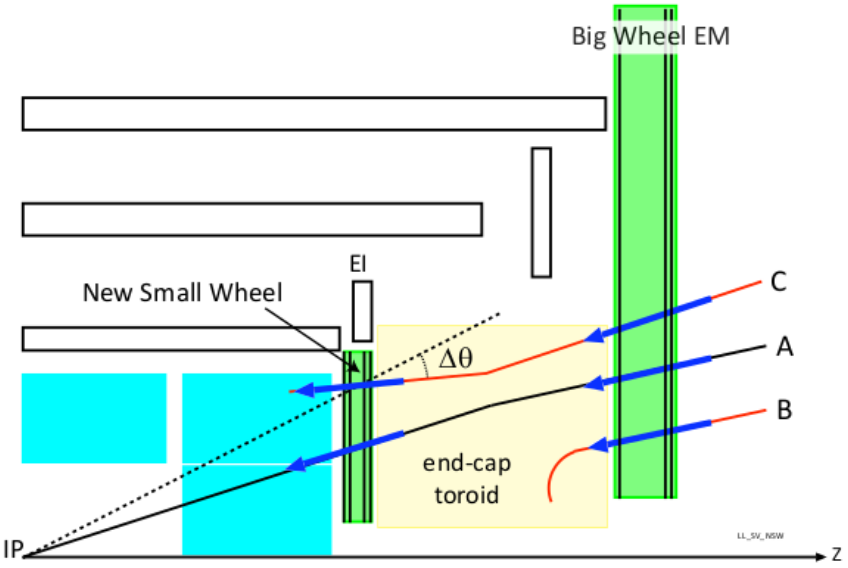


Figure I.9: Σχηματική αναπαράσταση του συστήματος σκανδαλισμού στα καπάκια του ATLAS μετά την αναβάθμιση του NSW. Ο ήδη υπάρχον Big Wheel, δέχεται όλες τις τροχιές σωματιδίων που απεικονίζονται. Με την εγκατάσταση του NSW, μόνο η περίπτωση 'A' γίνεται δεκτή καθώς μόνο αυτή περνάει και από τα δύο συστήματα σκανδαλισμού. Η περίπτωση 'B' θα απορριφθεί καθώς ο NSW δε θα βρει την τροχιά του σωματιδίου που πέρασε από εκείνο το σημείο του μεγάλου τροχού. Η τροχιά 'C' θα απορριφθεί επειδή συνδυάζοντας τις γωνίες και από τα δύο συστήματα, το σύστημα επεξεργασίας δεν καταλήγει σε μία ευθεία που να οδηγεί στο σημείο αλληλεπίδρασης [7].

λεπτομερής ανακατασκευή των τροχιών, και ο υπολογισμός των ορμών (τα οποία πραγματώνονται σε μεταγενέστερα στάδια μετά την αποθήκευση των δεδομένων), θα γίνεται με ακόμα μεγαλύτερη ακρίβεια μετά την αναβάθμιση του NSW.

Αντιλαμβάνεται λοιπόν κανείς το κεντρικό ρόλο των ηλεκτρονικών σε όλη αυτή την περίπλοκη διαδικασία ακριβούς και γρήγορου υπολογισμού του σκανδαλισμού και στην ανακατασκευή των τροχιών, τα οποία ηλεκτρονικά οφείλουν όχι μόνο να μεταδώσουν γρήγορα και βέλτιστα τις σχετικές πληροφορίες, αλλά και να αντέξουν τις αντίξοες συνθήκες ακτινοβόλησης που θα επικρατούν πάνω στους ανιχνευτές όπου και θα βρίσκονται. Το κύριο σύστημα των εμπρόσθιων ηλεκτρονικών των ανιχνευτών Micromegas, απαρτίζεται από *τρεις* ηλεκτρονικές πλακέτες. Σε αυτό το σημείο θα δοθεί η γενική ιδέα της λειτουργίας των καρτών αυτών [3, 4, 7], ενώ παρακάτω παρατίθεται και μία σχηματική αναπαράσταση της μεταξύ τους συνδεσμολογίας (βλ. Σχ. I.10):

- **Micromegas Front-End Board** *(MMFE, ή και MMFE8)*. Πρόκειται για την ηλεκτρονική πλακέτα που βρίσκεται πάνω στους θαλάμους Micromegas και

είναι υπεύθυνη για τη συλλογή των πρωταρχικών σημάτων από τις λωρίδες διαβάσματος του ανιχνευτή. Κάθε κάρτα MMFE8 περιλαμβάνει οκτώ *ASIC*[10] υπεύθυνα για το διάβασμα, που ονομάζονται *VMM*. Κάθε VMM έχει 64 κανάλια, όπου το κάθε ένα αντιστοιχεί σε μία λωρίδα διαβάσματος του ανιχνευτή Micromegas. Εκτός από τα VMM ASIC, στην πλακέτα αυτή βαίνουν και άλλα δύο ASIC υπεύθυνα για την διανομή σημάτων διαμόρφωσης και την εξαγωγή των ψηφιακών δεδομένων που προκύπτουν από την ανάλυση των παλμών από τα VMM. Αυτά τα ASIC είναι το *Slow Control Adapter (SCA)* και το *Read-Out Controller (ROC)* αντίστοιχα. Τα πρωτότυπα των MMFE8 όμως, δεν διέθεταν αυτά τα δύο βοηθητικά ASIC, αλλά ένα *FPGA*[11], το οποίο υποκαθιστούσε τις λειτουργίες αυτών των μονάδων. Για το τελικό πείραμα, θα χρησιμοποιηθούν 4096 πλακέτες MMFE8.

- **Level-1 Data Driver Card** *(L1DDC)*. Πρόκειται για την ηλεκτρονική πλακέτα που δέχεται τα σειριακά δεδομένα από οκτώ MMFE, και τα συμπτύσσει σε μία ζεύξη οπτικής ίνας η οποία καταλήγει στο σύστημα των όπισθεν ηλεκτρονικών και συγκεκριμένα στις μονάδες *FELIX*[12]. Αυτό το επιτυγχάνει με τη χρήση ενός ASIC ονόματι *GBTx*. Εκτός από τη συλλογή και αποστολή δεδομένων για τους παλμούς που συλλέγει ο ανιχνευτής, η πλακέτα αυτή είναι υπεύθυνη και στο να αποστέλλει τα δεδομένα για τη διαμόρφωση της λειτουργικότητας των MMFE8, που προέρχονται από το κέντρο ελέγχου, και να παρέχει στις MMFE8 το ρολόι χρονισμού διασταύρωσης της δέσμης *(BC clock)*. Η κάθε κάρτα L1DDC όμως, δε συνδέεται μόνο με τις οκτώ MMFE8, αλλά και με την τρίτη πλακέτα, την ADDC που θα περιγραφεί στη συνέχεια. Συνολικά θα κατασκευαστούν 1024 κάρτες L1DDC για την αναβάθμιση του NSW (512 για τους Micromegas και 512 για τους sTGC). Η έρευνα-ανάπτυξη και σχεδίαση για τις κάρτες L1DDC πραγματοποιήθηκε από το Εθνικό Μετσόβιο Πολυτεχνείο.

- **ART Data Driver Card** *(ADDC)*. Πρόκειται για την ηλεκτρονική πλακέτα η οποία δέχεται τα δεδομένα ART (Address-in-real-time Data) από τις οκτώ MMFE με τις οποίες συνδέεται. Διαθέτει και αυτή με τη σειρά της δύο GBTx

[10] Application-Specific Integrated Circuit: Πρόκειται για μία ευρεία κατηγορία ηλεκτρονικών μονάδων, που μπορούν να λειτουργούν αναλογικά, ψηφιακά, ή και με τους δύο τρόπους ταυτόχρονα. Σχεδιάζονται με εξειδικευμένες γλώσσες περιγραφής ηλεκτρονικών πυλών, ή και με άλλα εργαλεία σχεδιασμού κυκλωμάτων. Τελούν πολύ συγκεκριμένες λειτουργίες, εξ ου και η αγγλική τους ονομασία.

[11] Field-Programmable Gate Array: Πρόκειται για μία οικογένεια ηλεκτρονικών μονάδων που λειτουργούν στο ψηφιακό πεδίο σημάτων, και είναι πλήρως επαναπρογραμματιζούμενα. Η εν λόγω διαδικασία τελείται από γλώσσες περιγραφής ψηφιακών πυλών, οι οποίες συνθέτουν το υλικολογισμικό που φιλοξενεί η μονάδα του FPGA.

[12] Front End LInk eXchange: Πρόκειται για ένα εξειδικευμένο σύστημα του ATLAS, που αποτελείται από FPGA και συμβατικούς ηλεκτρονικούς υπολογιστές που λειτουργούν συνδυαστικά προκειμένου να δρομολογούν μεγάλο όγκο δεδομένων με υψηλή απόδοση.

ASIC και ένα ART ASIC, τα οποία συλλέγουν τα δεδομένα ART, και τα πακετάρουν για να τα στείλουν μέσω οπτικής ίνας στα όπισθεν ηλεκτρονικά, και συγκεκριμένα στο κομμάτι που επεξεργάζεται τα δεδομένα του σκανδαλισμού. Η πλακέτα αυτή επικοινωνεί και με μία L1DDC, η οποία της στέλνει σήματα διαμόρφωσης λειτουργικότητας από το κέντρο ελέγχου, και αυτή της παρέχει σήματα χρονισμού.

Η επικοινωνία μεταξύ όλων των πλακετών γίνεται σειριακά, μέσω ζευγών διπολικών γραμμών (καλωδίων) που ονομάζονται *E-links*. Το κάθε E-link, αποτελείται από τρία ζεύγη διπολικών γραμμών. Στη μία γραμμή μεταδίδεται το σήμα του ρολογιού (Clk+, Clk-), στην άλλη πραγματοποιείται η μετάδοση των δεδομένων (Dout+, Dout-), και στην τελευταία η λήψη των δεδομένων (Din+, Din-). Το κάθε E-link υποστηρίζει τρεις διαφορετικές ταχύτητες μετάδοσης δεδομένων $(80,\ 160,\ 320\,\text{Mbps})$ οι οποίες μπορούν να οριστούν από το χρήστη. Το επίπεδο τάσεων των E-link ακολουθεί ως επί το πλείστον τα πρότυπο LVDS και SLVS [3, 4, 8].

Στη συνέχεια θα μελετηθεί το υλικολογισμικό που έχει αναπτυχθεί για την MMFE8, αλλά πριν γίνει αυτό, θα περιγραφεί η συγκεκριμένη πλακέτα εκτενέστερα, όπως επίσης και το VMM ASIC, οκτώ από τα οποία βαίνουν απάνω στην MMFE8.

I.4.1 Micromegas Front-End Board

Η πλακέτα *Micromegas Front-End Board (MMFE8)*, είναι εκείνη που συνδέεται με τον ανιχνευτή Micromegas, και συλλέγει τα ηλεκτρικά σήματα τα οποία παράγονται από αυτόν (που είναι ουσιαστικά μία συγκέντρωση των φορτίων που προέρχονται από αλλεπάλληλους ιονισμούς από τις λωρίδες διαβάσματος). Έχει διαστάσεις $215 \times 60\,\text{mm}^2$. Πρόκειται κατά κάποιον τρόπο για τον ενδιάμεσο μεταξύ του ανιχνευτή και των δύο καρτών που διαχειρίζονται τα δεδομένα σκανδαλισμού (ADDC) και τα δεδομένα για την ενέργεια και το χρόνο εμφάνισης των παλμών που θα χρησιμεύσουν αργότερα στην ανακατασκευή των τροχιών και των ορμών (L1DDC). Το νούμερο ”8” στο όνομα υπονοεί ότι κάθε MMFE8 έχει πάνω της οκτώ VMM ASIC, το κάθε ένα εκ των οποίων συνδέεται με 64 κανάλια (λωρίδες διαβάσματος - *strips*) του Micromegas, και προβαίνει σε λειτουργίες ανάλυσης σημάτων, όπως ενίσχυση, διαμόρφωση σχήματος, εύρεση κορυφών, και ψηφιοποίηση. Εκτός από τα οκτώ VMM ASIC, η τελική κάρτα MMFE8, θα διαθέτει και άλλα δύο ASIC: Το *SCA (Slow Control Adapter)*, και το *ROC (Read-Out Controller ASIC)*. Το SCA δέχεται τα σήματα της διαμόρφωσης λειτουργίας των μονάδων ASIC της MMFE και στέλνει σήματα για την παρακολούθηση των συνθηκών που επικρατούν στον ανιχνευτή. Η διασύνδεση αυτή γίνεται με την κάρτα L1DDC μέσω των E-link. Το ROC, συλλέγει τα ψηφιοποιημένα δεδομένα από τα VMM, τα συμπτύσσει, και τα αποστέλλει

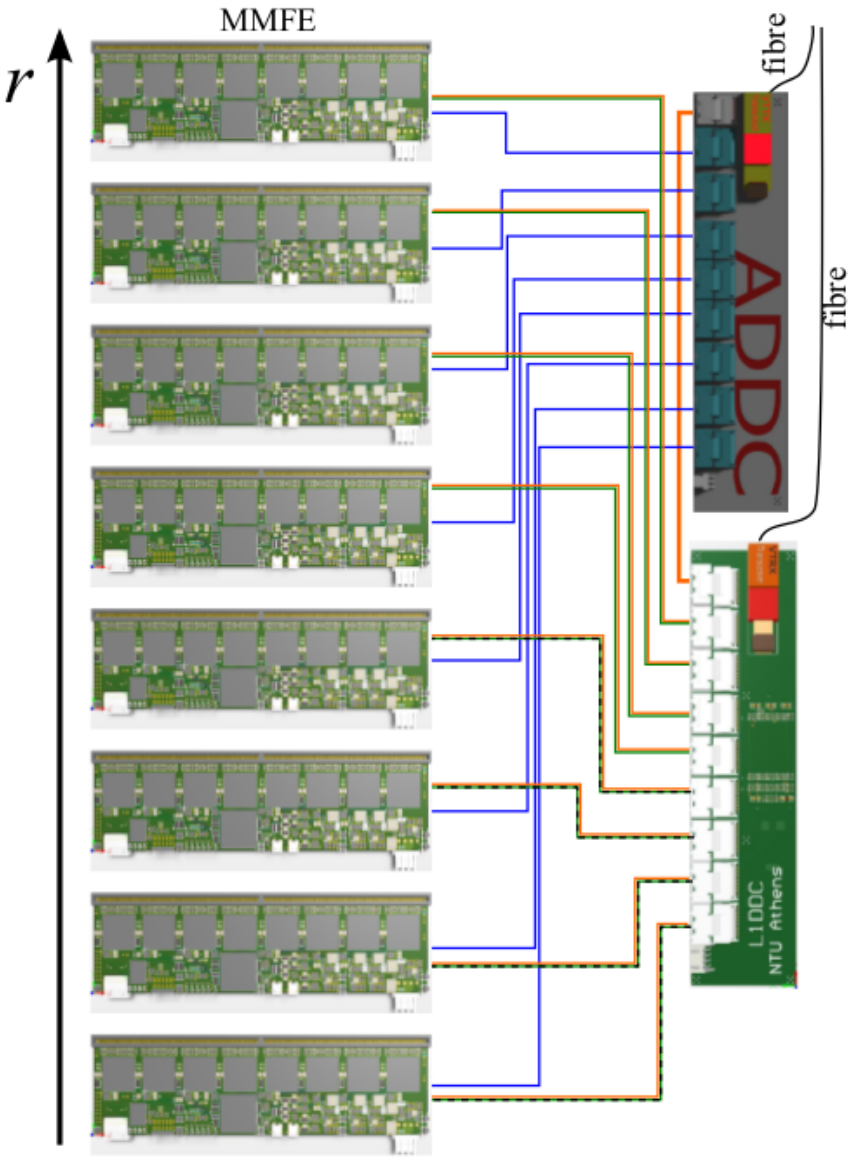


Figure I.10: Σχηματική αναπαράσταση της συνδεσμολογίας μεταξύ των εμπρόσθιων ηλεκτρονικών του NSW. Αριστερά είναι οι οκτώ πλακέτες MMFE, και στα δεξιά η κάρτα L1DDC συνδέεται με τις MMFE και την ADDC η οποία με τη σειρά της είναι και αυτή συνδεδεμένη με τις MMFE. Οι κάρτες L1DDC και ADDC συνδέονται μέσω οπτικής ίνας με το σύστημα των όπισθεν ηλεκτρονικών, και συγκεκριμένα με το δίκτυο FELIX και τον επεξεργαστή σκανδαλισμού αντίστοιχα [8].

σειριακά στην L1DDC μέσω των E-link. Ο ρυθμός μετάδοσης/λήψης δεδομένων είναι σταθερός στα 80 Mbps για τα E-link του SCA. Από την άλλη, αν η MMFE βρίσκεται στο εσωτερικό τμήμα του Micromegas (βλ. Σχ. I.8), ο ρυθμός μετάδοσης δεδομένων του ROC είναι στα 320 Mbps ενώ αν βρίσκεται στο εξωτερικό, είναι στα

160 Mbps. Ο λόγος για τον οποίο συμβαίνει αυτό είναι επειδή η υψηλότερη ροή σωματιδίων στο εσωτερικό τμήμα (μεγαλύτερες τιμές του $|\eta|$) απαιτεί και υψηλότερο εύρος ζώνης μετάδοσης δεδομένων για τις συγκρούσεις [8]. Τα πρώτα πρωτότυπα της MMFE8 που κατασκευάστηκαν δεν διέθεταν τα ROC/SCA ASIC, αλλά στη θέση τους υπήρχε ένα Xilinx® FPGA (Artix XC7A200T-2FBG484), για το οποίο αναπτύχθηκε υλικολογισμικό που προσομοίαζε τη λειτουργικότητα των δύο συνοδευτικών ASIC. Η έρευνα-ανάπτυξη και σχεδίαση για τις κάρτες MMFE8 πραγματοποιήθηκε στο πανεπιστήμιο της Αριζόνα των ΗΠΑ.

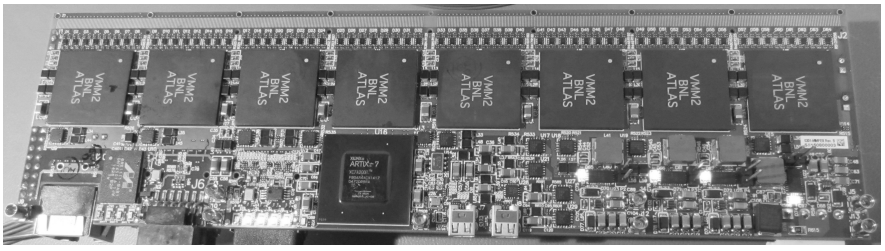


Figure I.11: Φωτογραφία ενός πρωτότυπου MMFE8. Διακρίνονται τα οκτώ VMM ASIC στο πάνω μέρος της πλακέτας, και το Xilinx® FPGA στο κάτω.

I.4.2 Μονάδα VMM ASIC

Το VMM, είναι ένα εμπρόσθιο ASIC διαβάσματος με 64 κανάλια εισόδου, που αναπτύχθηκε για την αναβάθμιση του μιονικού φασματόμετρου του ATLAS από το Brookhaven National Laboratory (BNL) στο Upton, NY των ΗΠΑ. Κάθε πλακέτα MMFE8 έχει πάνω της οκτώ VMM ASIC, και κατά συνέπεια συλλέγει δεδομένα από 512 κανάλια του Micromegas. Το VMM, έχει διαστάσεις $13.5 \times 8.4\,\text{mm}^2$ και περιέχει περίπου πέντε εκατομμύρια τρανζίστορ σχεδιασμένα με τεχνολογία CMOS στα 130 nm [3, 4]. Κάθε κανάλι που είναι συνδεδεμένο με μία λωρίδα διαβάσματος, υλοποιεί έναν ενισχυτή φορτίου *(charge amplifier)*, έναν ενισχυτή διαμόρφωσης *(shaping amplifier)* ο οποίος δρα ως φίλτρο, έναν διευκρινιστή *(discriminator)* με κατώτερο κατώφλι και εντοπισμό κορυφής, και ένα κύκλωμα μέτρησης χρόνου *(Time to Amplitude Converter - TAC)*. Τα αναλογικά σήματα που παράγονται από αυτά τα ηλεκτρονικά υποσυστήματα, ψηφιοποιούνται από τρεις διαφορετικούς ψηφιοποιητές *(Analog to Digital Converter - ADC)*, με εξόδους 6,8 και 10 bit. Τα (ψηφιακά πλέον) δεδομένα που προκύπτουν από την ανάλυση των παλμών, αποθηκεύονται σε μία μνήμη *(First-In First-Out - FIFO)* η οποία αποστέλλει τα δεδομένα στη μονάδα ROC της MMFE8 [9–11]. Σε γενικές γραμμές, το VMM προσφέρει ακριβείς μετρήσεις για το φορτίο που εναποτίθεται στα κανάλια από τα μιόνια, αλλά και για το πότε συνέβη αυτή η συλλογή φορτίου, σε

σχέση με το ρολόι διασταύρωσης της δέσμης[13]. Επίσης, παρέχει και άμεσα δεδομένα που χρησιμεύουν για το σκανδαλισμό, ενώ η διαδικασία του διαβάσματος από την L1DDC είναι πιο σπάνια, και εξαρτάται από το αν ένα γεγονός θεωρηθεί έγκυρο από τον σύστημα του σκανδαλισμού. Η διαδικασία αυτή θα ξεκαθαριστεί στη συνέχεια, μετά από μία σχηματική αναπαράσταση των ηλεκτρονικών υποσυστημάτων του VMM στο Σχ. I.12):

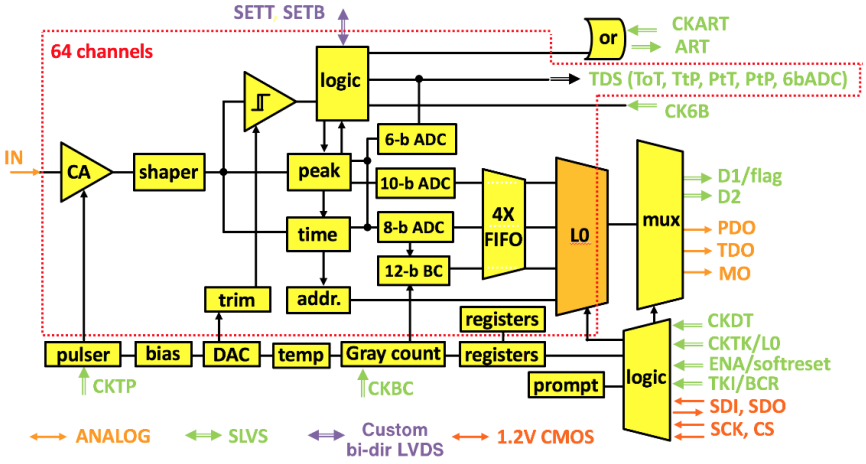


Figure I.12: Σχηματική αναπαράσταση της αρχιτεκτονικής ενός καναλιού του VMM. [11]

Όταν ένας παλμός εισέλθει στο VMM, ο προενισχυτής μετατρέπει το συνολικό φορτίο σε έναν αναλογικό παλμό τάσης ο οποίος μορφοποιείται από έναν ημι-Γκαουσιανό διαμορφωτή/φίλτρο, που παράγει το τελικό σήμα τάσης. Το μέγιστο πλάτος του σήματος τάσης είναι ανάλογο του φορτίου που εναποτέθηκε στο κανάλι. Το σχεδιάγραμμα σε επίπεδο CMOS του προενισχυτή παρατίθεται στο Σχ. I.13.

Το επεξεργασμένο σήμα από το φίλτρο εισέρχεται στη συνέχεια στο διευκρινιστή, ο οποίος έχει ρυθμιζόμενο κατώτατο κατώφλι. Μόλις κάποιος παλμός ξεπεράσει το κατώφλι αυτό, ενεργοποιείται μία μέθοδος εντοπισμού της κορυφής του παλμού. Τη στιγμή που εντοπίζεται η κορυφή και καταγράφεται το ύψος της, ενεργοποιείται το κύκλωμα μέτρησης χρόνου, το οποίο λαμβάνει το σήμα έναρξης και ξεκινάει να εκφορτίζει έναν πυκνωτή. Το σήμα άρσης έρχεται με την επόμενη αρνητική ακμή του ρολογιού αναφοράς, και ο πυκνωτής παύει να εκφορτίζεται. Με αυτό τον τρόπο, υπολογίζεται το ύψος του παλμού, άρα και το φορτίο που συλλέχθηκε, που υποδεικνύει το πότε εμφανίστηκε ο παλμός, με ακρίβεια της τάξης των ps. Αξίζει επίσης να

[13] Το Bunch Crossing (BC) clock είναι ένα ρολόι με περίοδο 25 ns η θετική ακμή του οποίου ταυτίζεται με τη στιγμή που τα πρωτόνια της δέσμης του LHC διασταυρώνονται μεταξύ τους. Ταυτίζεται επομένως με τη στιγμή που δημιουργούνται τα σωματίδια υπό ανίχνευση στο κέντρο του ATLAS.

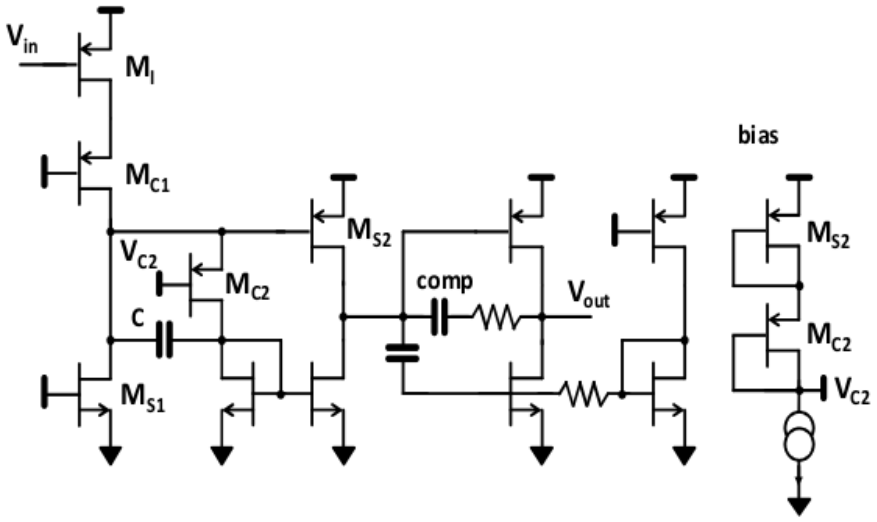


Figure I.13: Σχεδιάγραμμα του ενισχυτή του VMM σε επίπεδο τρανζίστορ [9].

σημειωθεί πως όταν ένα σήμα ξεπερνάει το κατώφλι του διευκρινιστή, ενεργοποιείται και ένα κύκλωμα που ειδοποιεί τα γειτονικά κανάλια να αγνοήσουν τον δικό τους διευκρινιστή, και να καταγράψουν έτσι και αλλιώς τις ενέργειες των παλμών τους, όσο μικροί και αν είναι αυτοί. Αυτή η έξυπνη μεθοδολογία επιτρέπει στο να τεθεί ψηλά το επίπεδο του κατωφλίου, χωρίς ουσιαστικά να χάνονται πληροφορίες, αφού όταν καταγραφεί μεγάλος παλμός σε ένα κανάλι, τα διπλανά θα αποθηκεύσουν με τη σειρά τους την ενέργεια που συνέλεξαν, η οποία δεν θα πρέπει να αγνοηθεί, όσο μικρή και αν είναι αυτή, καθώς θα αντιστοιχεί λογικά σε ένα συνονθύλευμα φορτίου *(cluster)* του ανιχνευτή Micromegas. [3, 4, 7].

Τρία διαφορετικά κυκλώματα μετατροπής από αναλογικό σε ψηφιακό (ADC), δέχονται τα αποτελέσματα της ανάλυσης των σημάτων, και τα ψηφιοποιούν. Πιο συγκεκριμένα, για τη μέτρηση του χρόνου, ο 8-bit ADC, ψηφιοποιεί τον παλμό του πυκνωτή από το κύκλωμα μέτρησης χρόνου (TAC), και συμπτύσσει το ένα byte που παράγει, με ένα αλφαριθμητικό μήκους 12-bit, το οποίο δημιουργείται από έναν μετρητή Gray Code που αυξάνεται από το ρολόι αναφοράς. Ουσιαστικά, η έξοδος του TAC, παράγει την πληροφορία της ακριβούς μέτρησης του χρόνου, και ο μετρητής Gray, ο οποίος φτάνει στη μέγιστη τιμή του και μηδενίζεται σε πιο αραιά χρονικά διαστήματα, παρέχει μία σχετικά πιο χονδροειδή εκτίμηση του χρόνου. Συνδυάζοντας όμως και τα δύο αποτελέσματα, κατασκευάζεται ένα αλφαριθμητικό μήκους 20 bit, που φέρει την πληροφορία του ακριβούς χρόνου που εμφανίστηκε ένας παλμός με διακριτική ικανότητα. Για τη μέτρηση της ενέργειας, ο 10-bit ADC λαμβάνει το σήμα που φέρει την πληροφορία του μέγιστου ύψους του παλμού από τον ανιχνευτή κορυφής, και το ψηφιοποιεί με μεγάλη ακρίβεια σε περίπου 200 ns. Ο ADC των 6 bit από την

άλλη, προβαίνει και αυτός σε μετατροπή του σήματος του ανιχνευτή κορυφής, αλλά με λιγότερη ακρίβεια σε σχέση με τον ADC των 10 bit. Η όλη διαδικασία ολοκληρώνεται σε 400 ns. Το αν θα χρησιμοποιηθεί ο πιο ακριβής ADC ή ο λιγότερο ακριβής για την κατασκευή του ψηφιακού πακέτου που θα εμπεριέχει την πληροφορία της συνολικής ενέργειας, εξαρτάται από τις προτιμήσεις του χρήστη.

Το τρίτο πρωτότυπο του VMM, υποστηρίζει δύο τύπους διαβάσματος των ψηφιακών δεδομένων από το FPGA/ROC [10, 11]. Ο πρώτος, που υποστηρίζεται και από τις παλαιότερες εκδόσεις του VMM, ονομάζεται *συνεχές διάβασμα (continuous readout)*, και η πληροφορία για την ενέργεια και το χρόνο εμφάνισης ενός παλμού για κάποιο κανάλι μορφοποιείται σε ένα πακέτο από 38 bit. Ανάμεσα σε αυτά, υπάρχουν τα 6 bit της διεύθυνσης του καναλιού, μετά τα 10 bit του ύψους του παλμού (δηλαδή η ενέργεια που εναποτέθηκε από το σωματίδιο), και τέλος τα 20 bit του χρόνου. Η τελική συμβολοσειρά αποθηκεύεται σε μία μνήμη η οποία έχει τέσσερις διαθέσιμες θέσεις. Όταν η εξωτερική μονάδα θελήσει να διαβάσει αυτά τα δεδομένα, θα αδειάσει όλα τα περιεχόμενα αυτής της μνήμης. Ο δεύτερος τρόπος διαβάσματος των δεδομένων του VMM, ονομάζεται *διάβασμα Level-0 (Level-0 readout)*, και είναι ο τρόπος με τον οποίο θα αποσπούνται τα δεδομένα στο τελικό πείραμα. Στο διάβασμα Level-0, η επικοινωνία μεταξύ των δύο μονάδων υλοποιείται μέσω του πρωτοκόλλου *8b/10b*, όπου το VMM στέλνει *comma characters*[14] όταν δεν έχει δεδομένα να στείλει. Η μονάδα που θέλει να διαβάσει δεδομένα από το VMM, πρέπει να στείλει ένα σήμα, το *Level-0* στο VMM. Το σήμα αυτό, θα λάβει ένα στοιχείο χρόνου από το VMM, όπως και τα γεγονότα που καταγράφει από τον ανιχνευτή. Βάσει αυτού του στοιχείου χρόνου και του πως έχει διαμορφωθεί η λειτουργία του VMM από το χρήστη, το VMM θα ”ψάξει” σε μία συγκεκριμένη περιοχή της μνήμης του για κάποιο γεγονός. Αν αυτό βρεθεί, τότε θα στείλει ένα συγκεκριμένο πακέτο δεδομένων (παρόμοιο με αυτό που περιγράφηκε πριν), που περιέχει την πληροφορία που συνέλεξαν όλα τα κανάλια για εκείνη την περιοχή της μνήμης. Αυτή η λειτουργία είναι ιδιαίτερα χρήσιμη, καθώς αναπόφευκτα το VMM θα συλλέγει ηλεκτρονικό θόρυβο, αλλά και γεγονότα που δεν αντιστοιχούν σε μιόνια. Μόνο λίγα από αυτά τα δεδομένα θα αντιπροσωπεύουν χρήσιμα γεγονότα πραγματικών μιονίων, και με αυτό τον τρόπο μπορεί κανείς να επιλέξει ακριβώς τα δεδομένα που θέλει από το VMM. Στο τελικό πείραμα, οι πληροφορίες από τους επεξεργαστές σκανδαλισμού θα συνδυάζονται προκειμένου να αποφανθεί σε ποια χρονική στιγμή αντιστοιχεί η πραγματική τροχιά ενός μιονίου, και έτσι θα στέλνεται την κατάλληλη στιγμή το σήμα Level-0 στα VMM ώστε να εξαχθεί αυτή την πληροφορία και μόνο.

Αξίζει επίσης να γίνει αναφορά σε διάφορες δευτερεύουσες λειτουργίες που διαθέτει

[14]Πρόκειται για προσυμφωνημένους χαρακτήρες μεταξύ πομπού και δέκτη, οι οποίοι ανταλλάσσονται προκειμένου να επιτευχθεί συγχρονισμός και ευθυγράμμιση φάσης.

το VMM. Μια τέτοια επιπρόσθετη λειτουργία, είναι η ύπαρξη κυκλωμάτων για τη βαθμονόμηση του κάθε καναλιού. Συγκεκριμένα, κάθε κανάλι του VMM παρουσιάζει διακυμάνσεις ως προς την ενίσχυση κατά τη μετατροπή των παλμών φορτίου σε παλμούς τάσης. Προκειμένου να γίνουν σωστές μετρήσεις στις κυματομορφές των παλμών, οι σχετικές αυτές διακυμάνσεις πρέπει να είναι γνωστές. Για αυτό το λόγο, το κάθε κανάλι του VMM διαθέτει ένα εξωτερικό κύκλωμα που διοχετεύει με παλμούς το κανάλι *(Test Pulser)*, ώστε να μετρηθεί η ακριβής απόκριση του καναλιού [7]. Εκτός από την ενίσχυση, είναι επίσης σημαντικό να γίνει βαθμονόμηση και του επιπέδου του κατωφλίου του διευκρινιστή[15], γεγονός που επιτυγχάνεται επίσης με τη χρήση του κυκλώματος δοκιμαστικών παλμών. Βαθμονόμηση επίσης επιδέχεται και ο TAC [3,4], ο οποίος παρουσιάζει μικροδιαφορές ως προς τη λειτουργικότητά του ανά κανάλι. Με τη χρήση της λειτουργίας των δοκιμαστικών παλμών, διοχετεύονται στο σύστημα παλμοί με συγκεκριμένο χρονικό προφίλ, που μπορεί να οριστεί από το χρήστη. Στη συνέχεια τα αποτελέσματα του TAC επιστρέφονται πίσω, όπου υπολογίζονται κάποιες χρονικές σταθερές διόρθωσης για κάθε κανάλι. Αυτές οι σταθερές, θα ληφθούν υπόψιν αργότερα στην ανάλυση του χρόνου όπως αυτός δίνεται από το κάθε κύκλωμα μέτρησης χρόνου, ώστε οι μετρήσεις κατά τη διάρκεια του πειράματος να είναι ακριβέστερες. Το αποτέλεσμα της χρονικής βαθμονόμησης απεικονίζεται στο Σχ. I.14.

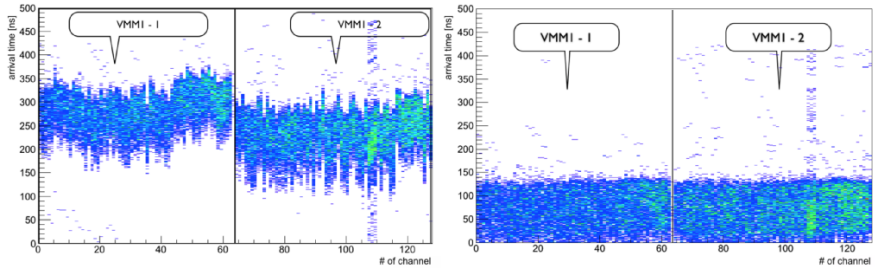


Figure I.14: Η κατανομή του υπολογισμένου χρόνου για δύο μονάδες VMM, πριν (αριστερά) και μετά (δεξιά) τη βαθμονόμηση. Είναι φανερό πως πριν τη βαθμονόμηση, παρουσιάζονται σημαντικές διακυμάνσεις στο χρόνο εμφάνισης των παλμών, όπως αυτός δίνεται από τον TAC. Μετά τη βαθμονόμηση, η κατανομή είναι σαφώς πιο ισορροπημένη [3,4].

Όλες αυτές οι λειτουργίες, μαζί με τη δυνατότητα διαμόρφωσης των επιμέρους λεπτομερειών του τρόπου λειτουργίας του VMM, υλοποιούνται ουσιαστικά μέσω της επικοινωνίας με την L1DDC. Όταν η L1DDC λάβει εντολή από το κέντρο ελέγχου του ATLAS ότι ένα συγκεκριμένο VMM πρέπει να αλλάξει κάτι στον τρόπο λειτουργίας του (π.χ. χρήση του 6-bit ADC αντί του 10-bit για πιο γρήγορη ψηφιοποίηση

[15] Ώστε να μην επηρεάζει τις μετρήσεις ο εγγενής θόρυβος κάθε καναλιού.

δεδομένων, αλλαγή του επιπέδου κατωφλίου του διευκρινιστή, κ.ά.), ή πρέπει να προβεί σε βαθμονόμηση, στέλνει την εντολή αυτή στο SCA της αντίστοιχης MMFE μέσω του E-link, και το SCA διοχετεύει τα ψηφιακά δεδομένα σε ένα μετατροπέα από ψηφιακό σε αναλογικό *(Digital-to-Analog Converter - DAC)* μέσα στο VMM. Ο αποκωδικοποιητής DAC, μετατρέπει το ψηφιακό σήμα της εντολής σε αναλογικό παλμό, ο οποίος αποστέλλεται στο αντίστοιχο ηλεκτρονικό υποσύστημα που πρέπει να γίνει η διαμόρφωση της λειτουργίας. Με αυτό τον τρόπο, υπάρχει μία πλήρης επικοινωνία μεταξύ του κέντρου ελέγχου και του VMM.

Δεδομένα ART και ADDC

Μαζί με την ανάλυση των σημάτων, το VMM παράγει και τα δεδομένα που χρησιμεύουν για το σκανδαλισμό. Πρόκειται για τα δεδομένα *Address in Real Time (ART)*. Καθ' όλη τη διάρκεια της λειτουργίας του, το VMM εποπτεύει τα αποτελέσματα από τους διευκρινιστές όλων των καναλιών του. Όταν κάποιος παλμός ξεπεράσει το κατώφλι του διευκρινιστή του, ή όταν βρεθεί η πρώτη κορυφή ενός παλμού, τότε το VMM καταγράφει αμέσως τη διεύθυνση του καναλιού που εντοπίστηκε το γεγονός, και τη στέλνει στην ADDC. Τα σήματα ART, αποστέλλονται στην ADDC μέσω ενός E-link σε κάθε κάθε $25\,\text{ns}$. Στην ADDC, βρίσκονται τέσσερα ASIC: δύο ART και δύο GBTx ASIC. Οι δύο πρώτες μονάδες, επεξεργάζονται τα δεδομένα ART που τους στέλνονται από τις 8 MMFE μέσω των E-link, και επιλέγουν τα σήματα που θα στείλουν στα GBTx. Τα GBTx, που είναι ουσιαστικά μονάδες που προβαίνουν σε σύμπτυξη πολλών σημάτων σε μία έξοδο οπτικής ίνας μεγάλης ταχύτητας, στέλνουν τα δεδομένα ART στα όπισθεν ηλεκτρονικά όπου γίνεται η επεξεργασία των σημάτων σκανδαλισμού. Το εν λόγω σύστημα, που αποτελείται κυρίως από FPGA, συνδυάζει τα δεδομένα του σκανδαλισμού από τους sTGC και τους MM, αλλά και το σύστημα του μεγάλου τροχού. Για να συγχρονιστούν τα δεδομένα και των τριών ανιχνευτικών υποσυστημάτων, τα ηλεκτρονικά του κάθε ανιχνευτή πρέπει να αποστείλουν τα δεδομένα τους μέσα σε ένα αυστηρά καθορισμένο χρονικό παράθυρο. Για τον MM για παράδειγμα, από τη στιγμή που θα γίνει μία αλληλεπίδραση στο κέντρο του ATLAS, μέχρι να φτάσουν τα δεδομένα του σκανδαλισμού της αλληλεπίδρασης αυτής στο κομμάτι εκείνο του όπισθεν επεξεργαστή που συνδυάζει τα δεδομένα και των υπολοίπων θαλάμων, δύναται να παρέλθει ένα μέγιστο χρονικό διάστημα της τάξης των $t_{max} = 1025\,\text{ns}$. Αυτή η καθυστέρηση *(latency)* εξαρτάται από πολλούς παράγοντες, όπως για παράδειγμα: το χρόνο απόκρισης του VMM, την ταχύτητα με την οποία το GBTx στην ADDC συμπτύσσει τα δεδομένα από τα οκτώ E-links και τα αποστέλλει σειριακά στην οπτική ίνα που οδηγεί στον επεξεργαστή σκανδαλισμού, ή και την ταχύτητα μετάδοσης της πληροφορίας στις οπτικές ίνες που μόλις αναφέρθηκαν. Έχει υπολογιστεί πως το σύστημα ηλεκτρονικών του

Micromegas, προσφέρει ελάχιστη καθυστέρηση $t_{min} = 876\,\text{ns}$, και μέγιστη $t_{max} = 1018\,\text{ns}$, που είναι ένα χρονικό πλαίσιο πλήρως αποδεκτό βάσει του σχεδιασμού ολόκληρου του πειράματος [3, 4]. Τελικά, με τη σύγκριση όλων των δεδομένων, ο επεξεργαστής αποφαίνεται για το ποια γεγονότα είναι έγκυρα (βλ. Σχ. I.9), και αποστέλλει μέσω της L1DDC το σήμα του Level-0, επιλέγοντας έτσι μόνο τα χρήσιμα δεδομένα από τις μνήμες του VMM, όπως περιγράφηκε παραπάνω.

I.5 Το Υλικολογισμικό για το Διάβασμα του VMM

Σε αυτό το κομμάτι, θα παρουσιαστεί το υλικολογισμικό που έχει αναπτυχθεί για τα FPGA των πλακετών στα οποία βαίνει το VMM3[16]. Το εν λόγω έργο έχει δημιουργηθεί για να ελέγξει τις λειτουργίες του VMM, προκειμένου να υπάρχει πλήρης επίγνωση για τον τρόπο που συμπεριφέρεται η μονάδα κάτω από όλες τις δυνατές συνθήκες. Έτσι, οι σχεδιαστές του VMM μπόρεσαν να προβλέψουν τι αλλαγές έπρεπε να γίνουν στην αρχιτεκτονική του, πριν την τελική τοποθέτηση των ηλεκτρονικών στους ανιχνευτές του NSW. Εκτός από τη μελέτη των λειτουργιών του VMM όμως, το υλικολογισμικό κρίνεται απαραίτητο για υποστήριξη σε συνθήκες *τεστ δέσμης*. Οι ανιχνευτές Micromegas, από τους οποίους θα αποσπά ηλεκτρικά σήματα το VMM, δοκιμάστηκαν σε ακτινοβόληση δέσμης αρκετές φορές πριν εγκατασταθούν στο τελικό πείραμα. Μόνο έτσι μπόρεσε να γίνει χαρακτηρισμός της λειτουργίας του ανιχνευτή, καθώς από τα δεδομένα του VMM, μπόρεσε να γίνει η ανάλυση που θα υποδείκνυε την ποιότητα της διάταξης.

Η ευελιξία της μονάδας FPGA την καθιστά ιδανική για αυτό το σκοπό, καθώς ο τρόπος με τον οποίο λειτουργεί μπορεί να αλλάξει δυναμικά ώστε να τεστάρει διαφορετικές πτυχές του VMM, ή να διαμορφωθεί προκειμένου να αντεπεξέλθει στις συνθήκες πειραμάτων τεστ δέσμης. Το υλικολογισμικό αυτό, υποστηρίζει αρκετές πλακέτες[17] οι οποίες φιλοξενούν το VMM. Οι MMFE8 έχουν παραχθεί για το NSW, όμως το VMM δύναται να καταγράψει δεδομένα και από άλλους τύπους ανιχνευτών. Οι άλλες πλακέτες είναι συμβατές με τον Micromegas αλλά και με άλλους θαλάμους ανίχνευσης, και διαθέτουν συνήθως μόνο ένα VMM (αντί για οκτώ όπως στην MMFE8), όμως έχουν και αυτές διεπαφή Ethernet, κοννέκτορα miniSAS, και FPGA. Έτσι, το υλικολογισμικό που χρησιμοποιείται για διαφορετικές πλακέτες, είναι πρακτικά το ίδιο, αφού τα πρωτόκολλα επικοινωνίας και οι μονάδες που βαίνουν στις εκάστοτε πλακέτες δεν αλλάζουν.

[16] Πρόκειται για το τρίτο πρωτότυπο της μονάδας. Πολλές από τις λειτουργίες του είναι διαθέσιμες και στις προηγούμενες εκδόσεις του.

[17] MMFE8, MMFE1, MDT_446, MDT_MU2E, GPVMM.

Figure I.15: Τρεις πλακέτες που φιλοξενούν VMM, και το FPGA για το οποίο έχει σχεδιαστεί το υλικολογισμικό που θα περιγραφεί σε αυτό το Κεφάλαιο. Πάνω δεξιά είναι η MDT_446, πάνω αριστερά η MMFE1, και κάτω η MMFE8. Στις πλακέτες διακρίνεται το FPGA, το VMM και οι κοννέκτορες της διεπαφής Ethernet.

Το υλικολογισμικό, κατά κύριο λόγο, υλοποιεί δύο διαφορετικές λειτουργίες:

- *Διαμόρφωση:* Πρόκειται για τις εντολές ελέγχου που προέρχονται από το χρήστη, και έχουν σαν σκοπό να διαμορφώσουν δυναμικά τις λειτουργίες του FPGA ή/και του VMM. Υλοποιείται μέσω του πρωτοκόλλου Ethernet/UDP, όπου εξειδικευμένο λογισμικό[18] στέλνει τις εντολές στο FPGA, το οποίο στη συνέχεια τις αποκωδικοποιεί και δρα αναλόγως. Η συγκεκριμένη λειτουργία στην MMFE8 θα υλοποιείται στο τελικό πείραμα από το SCA, και οι εντολές θα προέρχονται από το κέντρο ελέγχου του ATLAS, που δρομολογούνται στην MMFE8 μέσω της L1DDC και των αντίστοιχων ζεύξεων E-link.
- *Λήψη Δεδομένων:* Η διαδικασία απόσπασης της ψηφιακής πληροφορίας από αναλογικούς παλμούς που λαμβάνει το VMM, η οποία υλοποιείται στην τελική MMFE8 από το ROC. Όπως αναφέρθηκε και στο προηγούμενο Κεφάλαιο, υπάρχει το συνεχές διάβασμα και το διάβασμα Level-0. Ο πρώτος τρόπος είναι όμοιος με εκείνον του VMM2, ενώ ο δεύτερος υφίσταται μόνο στο VMM3, το οποίο εν προκειμένω θα στέλνει τα δεδομένα σε πακέτα κωδικοποιημένα με το πρωτόκολλο *8b/10b*, και θα χρησιμοποιείται στο τελικό πείραμα του ATLAS.

[18] Θα καλείται από εδώ και στο εξής ως λογισμικό λήψης δεδομένων, ή λογισμικό DAQ.

Το FPGA δύναται να υλοποιήσει και τις δύο περιπτώσεις στη μέγιστη ταχύτητα των 320 Mbps. Στο τελικό πείραμα τα δεδομένα θα προωθούνται στην L1DDC μέσω του ROC και των E-link. Εν προκειμένω, το FPGA στέλνει τα δεδομένα στο λογισμικό DAQ μέσω του πρωτοκόλλου Ethernet/UDP, και το λογισμικό με τη σειρά του κατασκευάζει αρχεία .ROOT για ανάλυση των δεδομένων.

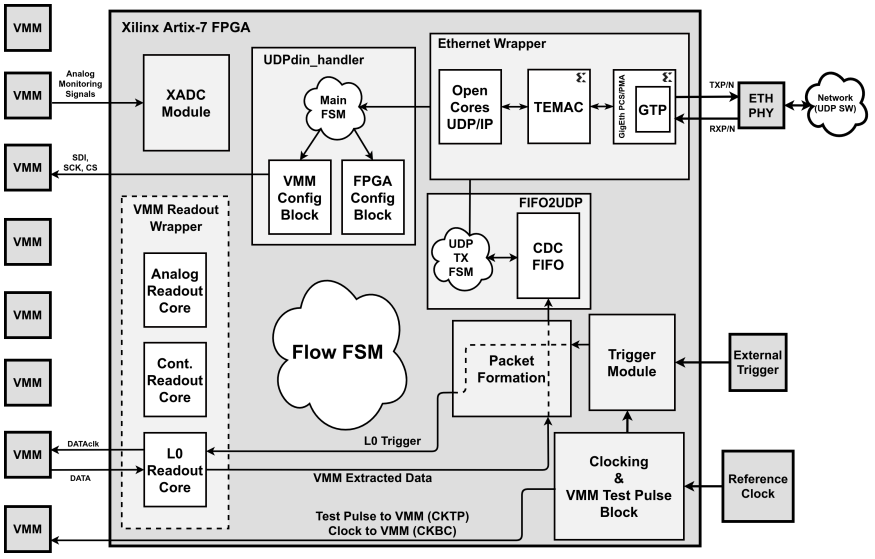


Figure I.16: Γενικό Block Diagram του υλικολογισμικού.

I.5.1 Διεπαφή Ethernet - Διαμόρφωση

Σε αυτή την ενότητα θα περιγραφούν όλα τα κομμάτια του υλικολογισμικού που έχουν να κάνουν με τη διαμόρφωση της λειτουργίας του FPGA και του VMM, από τα λογικά κομμάτια που είναι υπεύθυνα για την λήψη των πακέτων Ethernet (PCS/PMA, TEMAC), στις υπο-μονάδες που αναγνωρίζουν (UDP/ICMP κομμάτι) το UDP πακέτο και το προωθούν στη λογική του χρήστη (UDP_din_handler).

Ethernet/UDP Διεπαφές

Προκειμένου να πραγματοποιηθεί η επικοινωνία του FPGA με τον ηλεκτρονικό υπολογιστή, απαιτείται η υλοποίηση ενός τμήματος των επιπέδων OSI μέσα στο FPGA (βλ. Παράρτημα Β). Τα OSI (Open Systems Interconnection), είναι διάφορα αλληλοεπικαλυπτόμενα επίπεδα που προτυποποιούν και κατηγοριοποιούν τις λειτουργίες που πρέπει να επιτελέσει ένας κόμβος, προκειμένου αυτός να μπορεί να λάβει μέλος σε ένα δίκτυο επικοινωνιών. Τα επίπεδα αυτά είναι: Application, Presentation, Session, Transport, Network, Data Link και Physical, με σειρά από το ανώτερο στο κατώτερο [12]. Στο υλικολογισμικό της προκείμενης εφαρμογής, τα τμήματα των επιπέδων που

πρέπει να μοντελοποιηθούν μέσα στο FPGA είναι τα: Transport, Network, Data Link και Physical.

Όπως έχει ήδη γίνει σαφές, τα δεδομένα από τα VMM, θα συγκεντρώνονται από το FPGA, και θα πακετάρονται προκειμένου να σταλούν μέσω Ethernet στον υπολογιστή που θα είναι συνδεδεμένος με την πλακέτα. Για το σκοπό αυτό τα δεδομένα αρχικά πακετάρονται σύμφωνα με το πρότυπο που ορίζει το πρωτόκολλο UDP (επίπεδο Transport). Στη συνέχεια αυτά τα πακέτα προωθούνται στο επίπεδο Network, όπου υλοποιείται το πρωτόκολλο IPv4. Στο υλικολογισμικό της MMFE8, αυτές οι λειτουργίες τελούνται από κοινούς κώδικες VHDL που μπορούν να κατεβαστούν ελεύθερα από το Opencores.org. Τα πράγματα γίνονται πιο περίπλοκα στην υλοποίηση των Data Link και Physical επιπέδων, όπου είναι απαραίτητη η χρήση εξεζητημένων πυρήνων IP που διαθέτει η Xilinx® για τέτοιες εφαρμογές. Πρόκειται για τρεις μονάδες IP: *Tri-mode Ethernet MAC, Ethernet SGMII PCS/PMA* και *7-Series GTP Transceiver*. Η εφαρμογή των τριών αυτών πυρήνων, σε συνδυασμό με το υλικολογισμικό από το OpenCores, επιτρέπει στον χρήστη να κατασκευάσει ένα πλήρες σύστημα δικτύωσης στο FPGA, το οποίο μπορεί και υλοποιεί πρωτόκολλο Gigabit Ethernet μέσω χάλκινου καλωδίου.

PCS/PMA & TEMAC

Ο τρόπος με τον οποίο δομείται αυτό το σύμπλεγμα διαφορετικών πυρήνων μέσα στο υλικολογισμικό είναι ο εξής: Κατ' αρχάς, η διασύνδεση των πυρήνων IP, έχει ως εξής: TEMAC $\leftrightarrow$ PCS/PMA-SGMII $\leftrightarrow$ Transceiver. Ο Transceiver, είναι ένα ηλεκτρονικό υποσύστημα που μπορεί να μεταδίδει σειριακά δεδομένα με πολύ υψηλές ταχύτητες. Τα περισσότερα FPGA σήμερα, διαθέτουν ενσωματωμένα εξειδικευμένα υπο-κυκλώματα που υλοποιούν Transceivers[19], οι ταχύτητες των οποίων φτάνουν στην τάξη των Gbps. Το σύστημα αυτό, γενικά μπορεί να επικοινωνήσει μόνο με άλλες ανάλογες μονάδες που βρίσκονται εκτός του FPGA. Ένας Transceiver για παράδειγμα, μπορεί να επικοινωνήσει με συστήματα που εφαρμόζουν τις διασυνδέσεις με οπτικές ίνες, να υλοποιήσει PCI Express πρωτόκολλα, και πρωτόκολλα Ethernet. Εν προκειμένω, στην MMFE8 υλοποιείται το πρωτόκολλο Ethernet, και ο Transceiver του FPGA επικοινωνεί με ένα αντίστοιχο εξωτερικό κύκλωμα της Marvell®. Πρόκειται για το *Marvell 88E1111 Integrated Ultra Gigabit Ethernet Transceiver*, το οποίο είναι ένα εξωτερικό Ethernet PHY, που επικοινωνεί αμφίδρομα με το FPGA, και συγκεκριμένα με τον Transceiver. Η διεπαφή αυτή επιτρέπει στα πακέτα Ethernet του δικτύου να εισέρχονται στο FPGA, και στο FPGA να στέλνει πακέτα στο

[19]Το FPGA που χρησιμοποιείται στην MMFE8 και στις άλλες πλακέτες που φιλοξενούν το υλικολογισμικό, διαθέτει 16 Transceivers, όπου ο κάθε ένας μπορεί να υλοποιήσει διεπαφές Gigabit Ethernet μέσω χάλκινου καλωδίου, Gigabit Ethernet μέσω οπτικής ίνας, PCIe κ.λπ.

δίκτυο. Στις μεγάλες ταχύτητες, όπου απαιτείται εξισορρόπηση του επιπέδου τάσης συνεχούς ρεύματος για να μην αλλοιώνονται τα σήματα στις γραμμές, εφαρμόζεται συνήθως 8b/10b κωδικοποίηση (βλ. αντίστοιχο Παράρτημα) η οποία προσφέρει και ευκολότερη ανάκτηση ρολογιού στα κυκλώματα που συμμετέχουν στη διεπαφή από τη πλευρά της λήψης. Υπενθυμίζεται επίσης ότι η 8b/10b κωδικοποίηση, διαθέτει εξειδικευμένους χαρακτήρες, που στέλνονται συνεχώς από τον κόμβο που θέλει να μεταδώσει δεδομένα. Αυτοί έχουν το προσόν να διευκολύνουν την ανάκτηση του ρολογιού και την ευθυγράμμιση φάσης. Όταν αυτά λαμβάνονται σωστά από τον δέκτη, εκείνος γνωρίζει ότι είναι συμφασικός με τη ροή δεδομένων και μπορεί να κάνει σωστό δειγματοληψία των δυφίων.

Εν προκειμένω, ο χρονισμός ολόκληρου του κυκλώματος που υλοποιεί τη διεπαφή Ethernet, γίνεται από ένα ρολόι το οποίο ανακτά ο Transceiver από τη γραμμή επικοινωνίας με το Marvell ETH PHY[20]. Σαν ρολόι αναφοράς ($f = 125\,\text{Mhz}$), ο Transceiver λαμβάνει ένα διαφορικό σήμα *(MGTREFCLK)*, που προέρχεται από έναν κρύσταλλο υψηλής ποιότητας μέσα στο ETH PHY, και μέσω αυτού, τα εξειδικευμένα PLL του Transceiver συνθέτουν ρολόι από την εξωτερική ροή δεδομένων. Αυτό το ανακτημένο ρολόι, περνάει μέσα από ένα εξωτερικό MMCM (ανάλογο του PLL, βλ. υποεν. 3.1.4). Αυτό συνθέτει δύο ρολόγια (userclk, userclk2 με συχνότητες $62.5\,\text{Mhz}$ και $125\,\text{Mhz}$ αντίστοιχα) τα οποία χρονίζουν όλη τη λογική του κυκλώματος.

Όταν το UDP/IP κομμάτι, που υλοποιεί το επίπεδο Transport και Network, θελήσει να στείλει δεδομένα ένα επίπεδο κάτω, στο επίπεδο Link, θα λάβει τον χρονισμό του από τα ρολόγια του MMCM (δηλαδή ουσιαστικά από το ανακτημένο ρολόι), και θα προωθήσει το πακέτο που έχει κατασκευάσει στον TEMAC. Ο TEMAC, είναι ένα αυστηρά ορισμένο υποσύστημα, το οποίο τελεί τη διασύνδεση του Physical Layer με τα ανώτερα επίπεδα OSI. Πιο συγκεκριμένα, ο TEMAC, λαμβάνει τα πακέτα από το Network Layer (εν προκειμένω πακέτα IPv4 μαζί με UDP)[21], και τα ”τυλίγει” με το πακέτο Ethernet (βλ. Σχ. B.1).

Ο TEMAC προωθεί το πακέτο Ethernet που κατασκεύασε, στο επίπεδο Physical, που υλοποιείται από τον PCS/PMA. Ο τρόπος μετάδοσης των δεδομένων στο υλικολογισμικό, γίνεται σύμφωνα με το πρότυπο *SGMII (Serial Gigabit Media Independent Interface)*, το οποίο είναι μία υποπερίπτωση του GMII (που είναι παράλληλο) [13]. Το Marvell PHY, όντας μονάδα που υλοποιεί διεπαφή Ethernet μέσω χάλκινου καλωδίου και όχι οπτικής ίνας, θέτει άνω όριο στην ταχύτητα στα $1\,\text{Gbps}$, και υποχρεώνει τον PCS/PMA να εφαρμόζει το πρωτόκολλο 1000BASE-T μαζί με το SGMII (το 1000

[20] Πρόκειται για ένα ολοκληρωμένο κύκλωμα που υλοποιεί την διεπαφή του Transceiver με τη γραμμή του Ethernet.

[21] Το UDP προτιμάται στην παρούσα εφαρμογή, επειδή δεν έχουν μεγάλη επιβάρυνση στο εύρος ζώνης

υποδεικνύει ταχύτητα 1 Gbps, ενώ το -T διασύνδεση μέσω χαλκού).

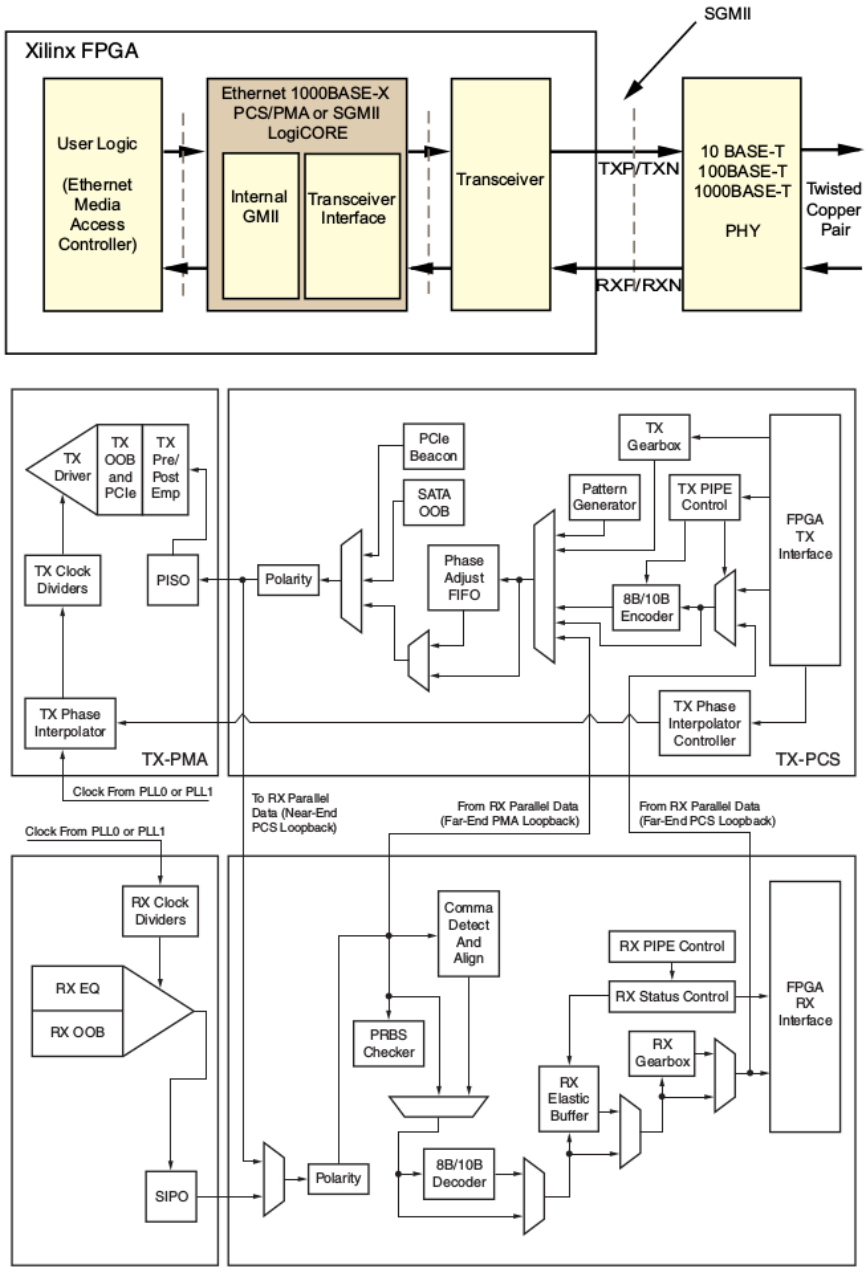


Figure I.17: Πάνω: Η διασύνδεση των τριών IP πυρήνων, τόσο με τα ανώτερα στρώματα OSI, όσο και με το εξωτερικό Ethernet PHY. Κάτω: Τα διάφορα υποσυστήματα του Xilinx® GTP Transceiver που χρησιμοποιεί το FPGA της MMFE8 [13].

OpenCores UDP/ICMP Stack

Πρόκειται για τον κώδικα VHDL *1G ETH UDP/IP Stack* από το *Opencores.org*. Το συγκεκριμένο κομμάτι λογικής υλοποιεί το *Internet (IPv4)* και το *Transport (UDP και ICMP)* επίπεδο. Η διεπαφή του είναι από τη μία με τον TEMAC και από την άλλη με το *UDP_din_handler*. Όταν ο TEMAC λάβει ένα πακέτο, τότε το αποστέλλει στον πυρήνα UDP/ICMP, σύμφωνα με το διάγραμμα του Σχ. I.18.

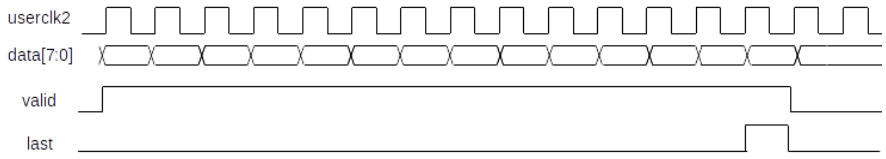


Figure I.18: Διάγραμμα χρονισμού των δεδομένων του Ethernet, όπως τα προωθεί ο TEMAC στο UDP/ICMP. Τα δεδομένα στέλνονται ανά byte, σε κάθε περίοδο του userclk2. Το σήμα *valid* παίρνει τη λογική τιμή ένα στο πρώτο byte του πακέτου, και μένει ψηλά καθ'όλη τη διάρκεια μετάδοσής του, ενώ το *last* θα μείνει ψηλά για έναν κύκλο του ρολογιού, σηματοδοτώντας έτσι το τελευταίο byte του πακέτου.

Το UDP/ICMP διακρίνει τα διαφορετικά μέρη του πακέτου Ethernet, και ξεχωρίζει το UDP δεδομενόγραμμα από το πακέτο του IPv4 (βλ. αντίστοιχο Παράρτημα και Σχ. Β.2), και προωθεί στις εξόδους του αρχικά τα δεδομένα του UDP Header (θύρες πομπού/δέκτη, μήκος πακέτου, checksum) και ύστερα τα περιεχόμενά του, μαζί με τα σήματα *valid* και *last* (παρόμοια με το Σχ. I.18, μόνο που αυτή τη φορά τα valid/last δε συνοδεύουν ολόκληρο το πακέτο Ethernet, αλλά μόνο το περιεχόμενο του UDP. Αυτά τα σήματα θα τα επεξεργαστεί ο *UDP_din handler*, προκειμένου να αποφανθεί τι εντολή εστάλη στο FPGA από το λογισμικό DAQ.

Ο αρχικός κώδικας του *1G ETH UDP/IP* δεν παρέχει υποστήριξη για το πρωτόκολλο ICMP, αλλά μόνο για το UDP. Αυτό σημαίνει ότι το κομμάτι λογικής που είναι υπεύθυνο για την αναγνώριση του πρωτοκόλλου (στο *Internet Layer*), ξεσκαρτάρει τα πακέτα που στο πεδίο *Upper Layer Protocol*, δεν έχουν την τιμή 0x11 που είναι ο δεκαεξαδικός κωδικός του UDP. Προκειμένου το FPGA να μπορεί να ανταποκρίνεται στα *Echo Requests (ή Ping Request)*[22] του υπολογιστή με τον οποίο είναι συνδεδεμένος, κρίθηκε απαραίτητη η προσθήκη λογικής που να αναγνωρίζει το πρωτόκολλο ICMP, και στην περίπτωση που αυτό το πακέτο ICMP αντιστοιχεί σε Echo Request, το υλικολογισμικό να απαντάει με Echo Reply. Για το λόγο αυτό, προστέθηκαν τρία κομμάτια λογικής:

[22]Πρόκειται για ένα χρήσιμο και δημοφιλές εργαλείο, το οποίο υλοποιείται μέσω του πρωτοκόλλου ICMP. Όταν ο πομπός στείλει Ping Request, τότε ο δέκτης λαμβάνει το πακέτο και απαντάει με Ping Reply. Χρησιμοποιείται για επιβεβαίωση της ορθής επικοινωνίας μεταξύ δύο κόμβων σε ένα δίκτυο.

- *ICMP_RX:* Αναγνωρίζει τα πακέτα που προέρχονται από το IP Layer με κωδικό πρωτοκόλλου 0x01 (δεκαεξαδικός κωδικός του ICMP), και τα προωθεί στον *ping_reply_processor*.
- *ping_reply_processor:* Διαβάζει το πακέτο ICMP που προέρχεται από το *ICMP_RX* και αν πρόκειται για Echo Request, τότε τελεί τις απαραίτητες διεργασίες για να κατασκευάσει το Ping Reply. Συγκεκριμένα, ο πυρήνας αυτός, ελέγχει τα πεδία *Type* και *Code* του πακέτου ICMP, και αν αυτά έχουν δεκαεξαδικούς κωδικούς 0x08 και 0x00 αντίστοιχα[23], τότε προωθεί στο ICMP_TX το πακέτο της απάντησης.
- *ICMP_TX:* Αναγνωρίζει τα πακέτα τύπου Ping Reply όπως του τα στέλνει ο προαναφερθέντας επεξεργαστής, και τα προωθεί στο επίπεδο IP, που με τη σειρά του τα στέλνει στον TEMAC, και από εκεί τελικά στη γραμμή του Ethernet. Έτσι ο κόμβος που έκανε το αρχικό αίτημα, θα λάβει την απάντηση του.

Στο Σχ. I.19 μπορεί κανείς να δει την αναπαράσταση των κομματιών λογικής που μόλις περιγράφηκαν.

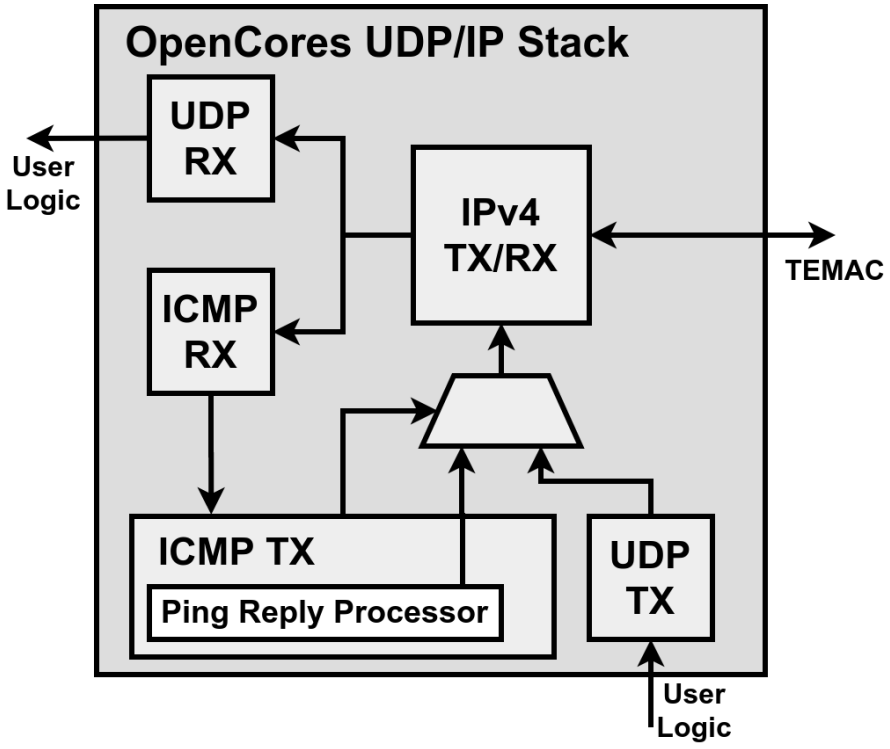


Figure I.19: Η δομή του πυρήνα UDP/ICMP.

[23]Πρόκειται για τον κωδικό αναγνώρισης πακέτων που προβαίνουν σε Echo Request.

I.5.2 Λογική Χρήστη Διαμόρφωσης

Σε αυτή την υποενότητα θα περιγραφούν τα κομμάτια του υλικολογισμικού που έχουν κατασκευαστεί έτσι ώστε να διαμορφώνουν τη λειτουργία του FPGA και του VMM.

UDP Data_In Handler

Τα δεδομένα που προέρχονται από το UDP, προωθούνται στο κομμάτι λογικής που είναι υπεύθυνο για την αναγνώριση και επεξεργασία των πακέτων αυτών. Το πρωτόκολλο επικοινωνίας μεταξύ του λογισμικού διαμόρφωσης και του υλικολογισμικού, υποδεικνύει ότι υπάρχουν δύο περιπτώσεις διαμόρφωσης: η μια αφορά τη διαμόρφωση λειτουργιών του FPGA, και η άλλη του VMM. Το αν ένα ληφθέν πακέτο προορίζεται για τη διαμόρφωση του VMM ή του FPGA, εξαρτάται από την τιμή της θύρας δέκτη του πακέτου. Η κυρίως λογική ελέγχει κατ'αρχάς αν το σήμα *valid* που προέρχεται από τον πυρήνα UDP/ICMP είναι στο λογικό ένα. Αν κάτι τέτοιο ισχύει, τότε σημαίνει ότι έχει ληφθεί ένα πακέτο UDP. Στη συνέχεια ελέγχει το πεδίο της θύρας, και ανάλογα την τιμή του, ενεργοποιεί την αντίστοιχη λογική που θα επεξεργαστεί το περαιτέρω το πακέτο. Η ιεραρχία των κομματιών λογικής απεικονίζεται στο Σχ. I.20.

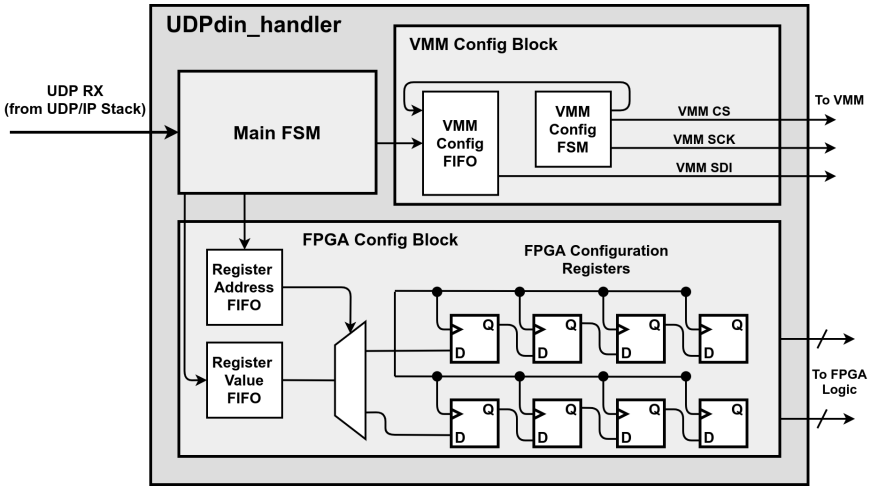


Figure I.20: Αρχιτεκτονική του UDP Data_In Handler. Η κύρια λογική ελέγχει δύο άλλες που είναι υπεύθυνες είτε για τη διαμόρφωση των λειτουργιών του FPGA, είτε του VMM.

Το FPGA διαθέτει πολυάριθμους καταχωρητές διαμόρφωσης οι οποίοι συνδέονται με τα περισσότερα κομμάτια του υλικολογισμικού, ούτως ώστε να μπορεί ο χρήστης μέσω του λογισμικού να αλλάζει δυναμικά τον τρόπο με τον οποίο λειτουργεί η ηλεκτρονική μονάδα. Το χρονοδιάγραμμα ενός πακέτου που στέλνεται από το λογισμικό για να διαμορφώσει τις λειτουργίες του FPGA μπορεί να βρεθεί στο Σχ. I.21. Υπάρχουν δύο

οικογένειες τιμών που βρίσκονται μέσα στο πακέτο, οι διευθύνσεις και οι τιμές. Το πρωτόκολλο επικοινωνίας μεταξύ του λογισμικού και του υλικολογισμικού υποδεικνύει τις θέσεις των τιμών αυτών μέσα στο πακέτο. Η λογική καταχωρεί αυτές τις τιμές καθώς το πακέτο διαβάζεται, και τις γράφει σε δύο μνήμες, μία για κάθε τύπο τιμής (βλ. Σχ. I.20). Μόλις εντοπιστεί το σήμα *last*, τότε οι μνήμες διαβάζονται προκειμένου να περάσουν τα δεδομένα στους αντίστοιχους καταχωρητές. Ουσιαστικά, η τιμή της διεύθυνσης, παίζει το ρόλο του σήματος *sel* ενός πολυπλέκτη, ο οποίος δρομολογεί τα δεδομένα διαμόρφωσης στον αντίστοιχο καταχωρητή μέσα στο FPGA[24].

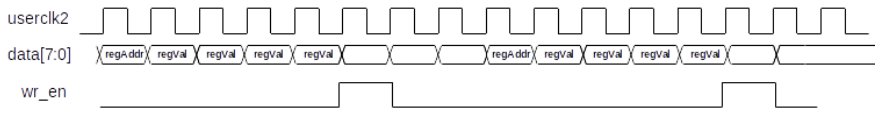


Figure I.21: Χρονοδιάγραμμα ενός πακέτου που προορίζεται για τη διαμόρφωση των λειτουργιών του FPGA. Πρώτα στέλνεται η διεύθυνση (ένα byte) και μετά η τιμή του καταχωρητή που αντιστοιχεί σε αυτή τη διεύθυνση (τέσσερα byte). Η λογική καταχωρεί τα δεδομένα, και την κατάλληλη στιγμή τις γράφει στις δύο αντίστοιχες μνήμες (σήμα *wr_en*).

Στην περίπτωση που ένα πακέτο προερχόμενο από το λογισμικό έχει σαν σκοπό να διαμορφώσει τις λειτουργίες του VMM, τότε θα δρομολογηθεί κατ' αρχάς σε μία μνήμα *(VMM Config FIFO)* η οποία θα αποθηκεύσει ολόκληρο το πακέτο. Μετά τον εντοπισμό του σήματος *last*, τότε η λογική του *VMM Config Block* θα διαβάσει τη μνήμη, ανά δυφίο, προκειμένου να σειριοποιήσει τα δεδομένα που πρέπει να περαστούν στο VMM. Η διαμόρφωση του VMM3 τελείται σύμφωνα με το πρωτόκολλο SPI (βλ. Σχ. I.22), και η λογική οδηγεί τα απαραίτητα σήματα *(SCK* και *CS)* στο VMM, ενώ τα δυφία διαμόρφωσης *(SDI)* έρχονται απευθείας από τη μνήμη. Το λογισμικό στέλνει αυτούσια τα 1728 bit που πρέπει να σταλούν στο VMM, επιτρέποντας έτσι στο χρήστη να έχει πλήρη έλεγχο πάνω στις λειτουργίες της μονάδας[25].

Γεννήτρια CKBC/CKTP - Βαθμονόμηση

Προκειμένου να μελετηθεί η συμπεριφορά του VMM κάτω από διάφορες συνθήκες, είναι απαραίτητη η ύπαρξη ενός κυκλώματος μέσα στη μονάδα που να εναποθέτει δοκιμαστικούς παλμούς στις εισόδους των προενισχυτών, οι οποίοι κανονικά δέχονται σήματα από τις λωρίδες διαβάσματος του θαλάμου με τον οποίο είναι συνδεδεμένοι.

[24] Τα δεδομένα περνάνε ανά δυφίο στον καταχωρητή (ο οποίος έχει τη μορφή ενός *Shift Register*)

[25] Μπορεί κανείς για παράδειγμα να επιλέξει το πόσο θα ενισχύει το σήμα εισόδου ο προενισχυτής του VMM ή να ενεργοποιήσει το κύκλωμα που στέλνει δοκιμαστικούς παλμούς σε συγκεκριμένα κανάλια.

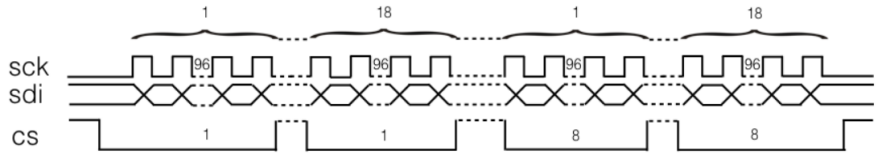


Figure I.22: Η επικοινωνία του VMM3 με τη μονάδα που το διαμορφώνει (Εν προκειμένω το FPGA, στο τελικό πείραμα, το SCA) τελείται μέσω του πρωτοκόλλου SPI. Το *CS* ενεργοποιείται και απενεργοποιείται μετά από κάθε 96 SCK. Έτσι στέλνονται τμηματικά τα 1728 δυφία διαμόρφωσης.

Η ανάγκη για την ύπαρξη ενός τέτοιου μηχανισμού είναι διττή: Επιτρέπει κατ' αρχάς στον χρήστη να δοκιμάσει τη μονάδα κανάλι προς κανάλι, ανεξάρτητα από το αν το VMM είναι προσαρτημένο απάνω σε κάποιο ανιχνευτή. Οι καταχωρητές διαμόρφωσης του VMM δίνουν τη δυνατότητα ενεργοποίησης του κυκλώματος δοκιμαστικών παλμών σε συγκεκριμένα κανάλια. Κάθε κανάλι λειτουργεί λίγο-πολύ ανεξάρτητα από τα άλλα (για μία γενική άποψη του κυκλώματος κάθε καναλιού βλ. Σχ. I.12), και ένας μηχανισμός που θα οδηγεί παλμούς στο κανάλι κατά τη βούληση του χρήστη, επιτρέπει τον πλήρη έλεγχο της ορθής λειτουργίας του καναλιού.

Ο άλλος λόγος για τον οποίο οι δοκιμαστικοί παλμοί είναι αναγκαίοι, είναι ότι κάθε κανάλι πρέπει να υποστεί βαθμονόμηση. Ως γνωστόν, το VMM ψηφιοποιεί την πληροφορία του ύψους του παλμού και του χρόνου εμφάνισής του, με τη χρήση κάποιων μετατροπέων, οι οποίοι όμως δεν θα έχουν όλοι την ίδια απόκριση ως προς τους παλμούς που ψηφιοποιούν. Ανατρέχοντας για παράδειγμα στην υποενότητα I.4.2, μπορεί κανείς να δει πως το VMM μετράει επακριβώς το χρόνο εμφάνισης ενός παλμού. Έχοντας σα ρολόι αναφοράς το ρολόι διασταύρωσης των δεσμών, που καλείται και *CKBC*[26], το VMM μετράει με μεγάλη ακρίβεια το χρόνο εμφάνισης της κορυφής ενός παλμού σε σχέση με την επόμενη αρνητική ακμή του CKBC. Ουσιαστικά αυτό που εισέρχεται στον ψηφιοποιητή μέτρησης του χρόνου είναι ένας παλμός, το ύψος του οποίου είναι ανάλογο της απόστασης της κορυφής του παλμού στην είσοδο του VMM[27] από την επόμενη αρνητική ακμή του CKBC. Η ψηφιακή πληροφορία του ύψους του παλμού, που αντιστοιχεί ουσιαστικά στο *χρόνο*, ονομάζεται *TDO* και έχει μήκος 8 bit. Ο άλλος μηχανισμός που πρέπει να βαθμονομηθεί αφορά την πληροφορία του *ύψους* του παλμού εισόδου, που είναι ανάλογος του φορτίου που εναποτέθηκε στην είσοδο του VMM, και σχετίζεται με την ενέργεια του σωματιδίου που εντοπίστηκε από το ανιχνευτικό σύστημα. Προφανώς, ένας άλλος ψηφιοποιητής/ADC αναλαμβάνει τη μετατροπή του ύψους του αναλογικού παλμού

[26]Το οποίο εν προκειμένω παρέχεται από το FPGA, και έχει $f = 40\,\text{MHz}$, ομοίως με το ρολόι που προέρχεται από τον ATLAS.

[27]Που μπορεί να είναι είτε δοκιμαστικός παλμός, είτε το φορτίο που εναποτέθηκε σε ένα κανάλι λόγω της ανίχνευσης ενός σωματιδίου.

εισόδου, και η 10-bit ψηφιακή λέξη που θα παράξει, ονομάζεται *PDO*. Οι τιμές των *PDO* και *TDO* λοιπόν, πρέπει να αντιστοιχούν στην πραγματικότητα, καθώς μόνο έτσι η μονάδα θα παρέχει αξιόπιστες πληροφορίες που θα οδηγούν στην αναγνώριση σωματιδίων και ανακατασκευή τροχιών στο τελικό πείραμα. Η απόκριση κάθε καναλιού σε παλμούς εισόδου θα είναι διαφορετική, όμως με τη βαθμονόμηση του συστήματος, παράγοντες διόρθωσης μπορούν να υπεισέρχονται στις τιμές των *PDO* και *TDO* για κάθε κανάλι όταν γίνεται η ανάλυση των δεδομένων που προέρχονται από το VMM.

Το κύκλωμα των δοκιμαστικών παλμών ενεργοποιείται για κάθε κανάλι ξεχωριστά από το λογισμικό. Ο χρήστης μπορεί μέσω του λογισμικού να στείλει ένα UDP πακέτο διαμόρφωσης λειτουργίας του VMM, και το υλικολογισμικό, μέσω του vmm_config_block, το οποίο έχει περιγραφεί παραπάνω, θα στείλει αυτά τα δεδομένα στο VMM μέσω του πρωτοκόλλου SPI. Το επόμενο στάδιο, είναι να διεγερθεί το κανάλι, αλλά προκειμένου συμβεί αυτό, πρέπει να σταλεί ο παλμός *CKTP (CK = Clock, TP = Test Pulse)* στο VMM. Αυτό γίνεται μέσω μίας γεννήτριας παλμών που βρίσκεται μέσα στη λογική του FPGA[28], που στέλνει κατ' αρχάς το CKBC στο VMM, ενώ αν ο χρήστης θέλει να ελέγξει την απόκριση του VMM, μπορεί να στείλει και το CKTP. Το CKBC παράγεται από ένα PLL. Το CKTP παράγεται από πύλες flip-flop με τη βοήθεια μετρητών. Έτσι, ο χρήστης, μέσω του λογισμικού μπορεί να αλλάξει τη συχνότητα του CKBC $(40,\ 20$ ή $10\,\mathrm{MHz})$, την περίοδο του CKTP (από μερικές εκατοντάδες ns μέχρι μερικά ms), και το χρονικό διάστημα που το CKTP θα έχει την τιμή του λογικού ένα. Μπορεί επίσης να ελέγξει και το πλήθος των δοκιμαστικών παλμών που θα εναποτεθούν στα κανάλια του VMM. Όλες αυτές οι παράμετροι ανήκουν στην γενικότερη κατηγορία των καταχωρητών διαμόρφωσης FPGA, αν και επηρεάζουν το VMM.

Με την άφιξη του μετώπου του CKTP, και υπό την προϋπόθεση ότι το κύκλωμα δοκιμαστικών παλμών μέσα στο VMM έχει ενεργοποιηθεί από το χρήστη, θα εναποτεθεί ένας δοκιμαστικός παλμός στο κανάλι, το ύψος του οποίου ελέγχει ο χρήστης μέσω της διαμόρφωσης της λειτουργίας του VMM. Μέσω ενός DAC μέσα στο VMM, μετατρέπεται η ψηφιακή πληροφορία του ύψους του παλμού που επιθυμεί ο χρήστης να στείλει στο κανάλι σε πραγματικό αναλογικό ύψος. Προφανώς, όσο μεγαλύτερη τιμή ύψους παλμού στείλει ο χρήστης στον DAC[29], άρα και στο κανάλι, τόσο μεγαλύτερη θα είναι και η τιμή του PDO αυτού του παλμού. Γνωρίζοντας a priori το ύψος του παλμού σε mV όμως, ο χρήστης μπορεί να μελετήσει την απόκριση του καναλιού,

[28] Στο υλικολογισμικό ονομάζεται *clk_gen_wrapper* και πρόκειται ουσιαστικά για ένα ειδικά διαμορφωμένο PLL.

[29] Ο ίδιος DAC, που είναι κοινός για όλο το VMM, πρέπει να υποστεί και αυτός με τη σειρά του βαθμονόμηση. Αυτό γίνεται με τη βοήθεια του xADC, το οποίο μελετάται σύντομα παρακάτω.

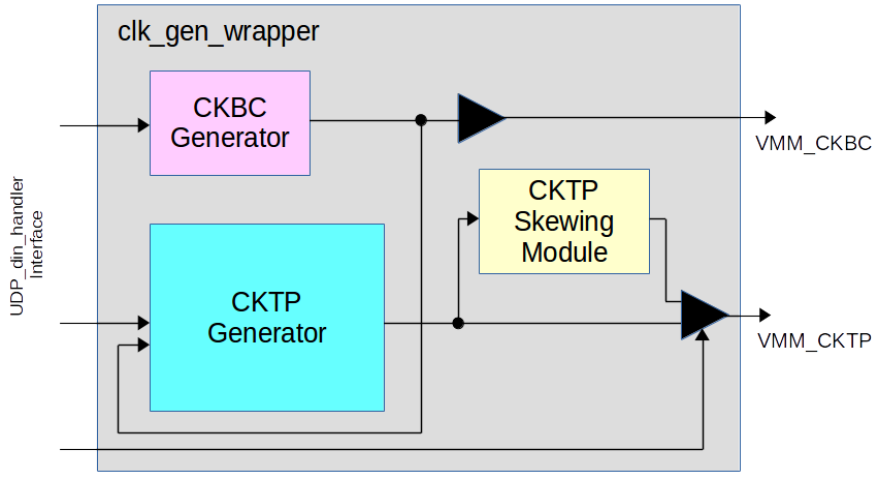


Figure I.23: Αρχιτεκτονική της γεννήτριας CKBC/CKTP. Ο χρήστης ελέγχει τα χαρακτηριστικά των δύο παλμών, όπως και τη μεταξύ τους διαφορά φάσης.

και να αντιστοιχήσει τις ψηφιακές τιμές του PDO σε πραγματικό ύψος παλμού. Έτσι βαθμονομείται το PDO.

Η βαθμονόμηση του TDO απαιτεί λίγη περισσότερη προσπάθεια, καθώς το VMM δεν διαθέτει κάποιον εγγενή μηχανισμό που να ελέγχει το χρόνο άφιξης του CKTP σε σχέση με το CKBC. Για αυτό το λόγο, σχεδιάστηκε ένα ειδικό κύκλωμα στο FPGA, που να αλλάζει δυναμικά το χρονικό προφίλ του CKTP σε σχέση με το CKBC. Ο πυρήνας αυτός, ονομάζεται *CKTP Skewing Module* μέσα στο υλικολογισμικό, και ελέγχει τη διαφορά φάσης μεταξύ των δύο παλμών. Έτσι λοιπόν, η τρίτη παράμετρος του CKTP που μπορεί να ελέγξει ο χρήστης δυναμικά, είναι το *CKTP Skew*, και το διάγραμμα των παλμών σε σχέση με την τιμή αυτής της παραμέτρου απεικονίζεται στο Σχ. I.24.

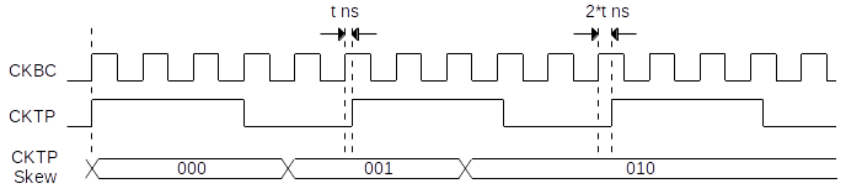


Figure I.24: Ο χρήστης έχει τη δυνατότητα να αλλάξει τη διαφορά φάσης μεταξύ των δύο παλμών. Αν η τιμή της διαφοράς φάσης είναι μηδέν, τότε οι ακμές τους ”συμπίπτουν”. Αν η τιμή είναι ένα, τότε το CKTP θα καθυστερήσει κατά t ns. Αν είναι δύο, τότε η καθυστέρηση του CKTP θα είναι $2t$ ns, όπου t το βήμα του χρόνου της διαφοράς φάσης *(skew step)*.

Η αρχιτεκτονική του πυρήνα χρονικής βαθμονόμησης που επιτρέπει ουσιαστικά τη δυναμική αλλαγή της διαφοράς φάσης μεταξύ των δύο σημάτων (CKBC/CKTP),

απεικονίζεται στο Σχ. I.25, ενώ η λογική του έχει ως εξής: Κατ' αρχάς, η λογική που παράγει το CKTP, θα το παράξει ευθυγραμμισμένο με το CKBC. Το ευθυγραμμισμένο CKTP, που ονομάζεται *CKTP Unskewed* στο Σχ. I.25, θα είναι η μία είσοδος ενός *BUGMUX*[30] η έξοδος του οποίου θα οδηγείται στην είσοδο του VMM για το CKTP. Όταν ο χρήστης θέλει ευθυγραμμισμένα τα δύο σήματα, η τιμή της παραμέτρου του skewing θα είναι μηδέν και θα επιλέγεται αυτό το σήμα. Όταν όμως αυτή η τιμή είναι διάφορη του μηδενός, τότε επιλέγεται η έξοδος του *CKTP Skewing Module*. Αυτό που κάνει η συγκεκριμένη λογική, είναι να προσθέτει διαδοχικές καθυστερήσεις στο σήμα του CKTP, αλλάζοντας έτσι τη φάση του σε σχέση με το CKBC. Αυτό το επιτυγχάνει περνώντας το CKTP από μια συστοιχία καταχωρητών. Μετά από κάθε καταχωρητή, το CKTP θα καθυστερεί *τόσα ns όσο η περίοδος του ρολογιού που χρονίζει τους καταχωρητές*, το οποίο ονομάζεται *clk_skew* στο Σχ. I.25. Κάθε διαδοχική διασύνδεση μεταξύ των καταχωρητών οδηγείται στην είσοδο ενός πολυπλέκτη, ο οποίος ανάλογα την τιμή της παραμέτρου της διαφοράς φάσης, θα οδηγήσει και διαφορετικό στάδιο καθυστέρησης στη δεύτερη είσοδο του BUFMUX, άρα και στο VMM.

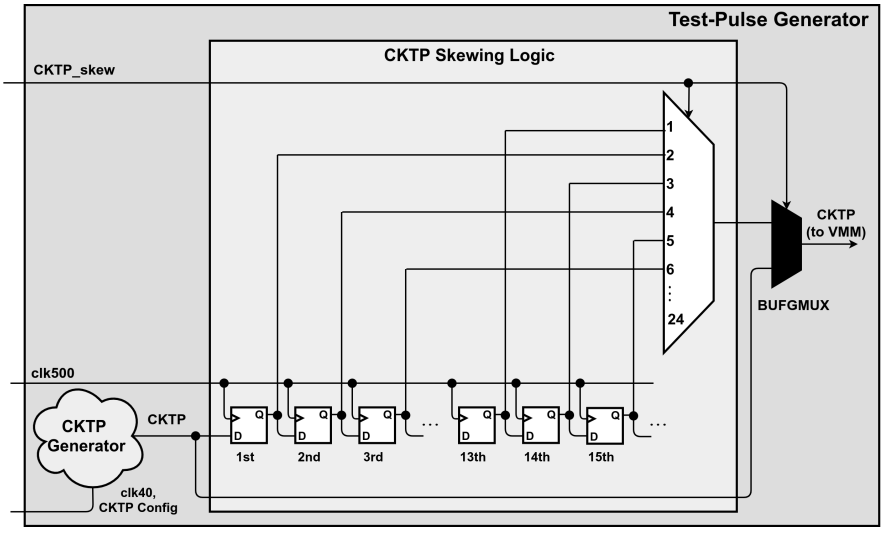


Figure I.25: Αρχιτεκτονική της γεννήτριας CKBC/CKTP. Ο χρήστης ελέγχει τα χαρακτηριστικά των δύο παλμών, όπως και τη μεταξύ τους διαφορά φάσης. Η περίοδος του ρολογιού *clk_skew* θα επηρεάζει άμεσα το βήμα της διαφοράς φάσης (συμβολίζεται με t στο Σχ. I.24).

Στη λογική χρησιμοποιείται ρολόι *clk_skew* με συχνότητα $f = 500\,\text{MHz}$, δηλαδή με περίοδο $T = 2\,\text{ns}$. Λαμβάνοντας πάντα υπόψιν τη περίοδο του CKBC, που είναι συνήθως $T_{CKBC} = 25\,\text{ns}$, ο αναγνώστης μπορεί να υποθέσει ότι *δώδεκα* επίπεδα

[30]Η γενική περίπτωση του BUFG (βλ. Κεφάλαιο 3), με δύο εισόδους και ένα σήμα επιλογής που επιτρέπει στο χρήστη να επιλέξει ποιο σήμα να οδηγήσει στην έξοδο του ενισχυτή.

καθυστέρησης αρκούν για να καλυφθεί μία περίοδος CKBC και να βαθμονομηθεί η απόκριση των ADCs ως προς το χρόνο. Θα υλοποιηθούν έτσι δώδεκα αλληλοσυνδεδεμένοι καταχωρητές και θα έχει κανείς $t = 2$ ns στο Σχ. I.24.

Ένας δέκατος-τρίτος καταχωρητής θα είχε νόημα, από τη στιγμή που η έξοδός του θα παρήγαγε ένα σήμα CKTP η καθυστέρηση του οποίου θα ξεπερνούσε την περίοδο του CKBC; Η απάντηση είναι πως θα *είχε νόημα*, καθώς έτσι, εν δυνάμει μπορεί κανείς να συρρικνώσει το βήμα t σε 1 ns. Ο επιπλέον καταχωρητής θα δημιουργήσει καθυστέρηση 26 ns, δηλαδή ουσιαστικά καθυστέρηση ενός ns μετατοπισμένο στην επόμενη περίοδο του ρολογιού αναφοράς (CKBC). Τοποθετώντας λοιπόν 24 καταχωρητές, λαμβάνει κανείς έναν πυρήνα λογικής που δύναται να δημιουργεί διαφορά φάσης μεταξύ των ακμών των CKBC και CKTP με βήμα *ενός* ns. Αυτό είναι που χρησιμοποιείται και στο προκείμενο υλικολογισμικό, επιτρέποντας έτσι την πολύ ακριβή μέτρηση της σχέσης μεταξύ *TDO* και *CKTP Skew* (βλ. Σχ. I.26). Με αυτόν τον τρόπο βελτιστοποιείται η βαθμονόμηση της χρονικής απόκρισης των καναλιών.

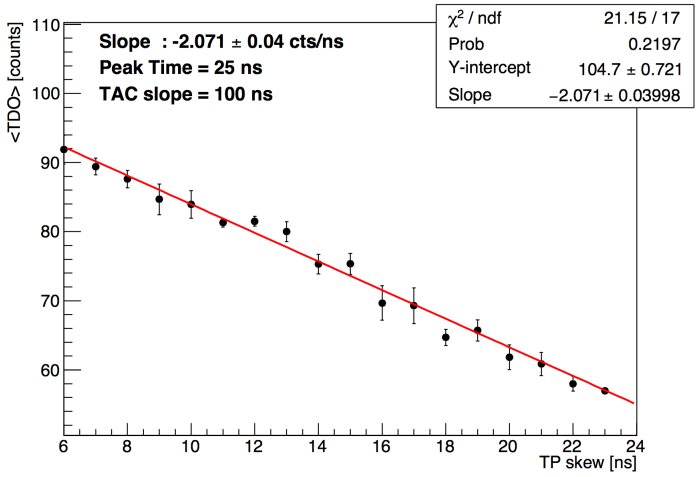


Figure I.26: Ευθεία που προκύπτει από τη βαθμονόμηση του TDO ενός καναλιού.

I.6 Διάβασμα του VMM - Απόσπαση/Λήψη Ψηφιακών Δεδομένων

Μέχρι τώρα έχει μελετηθεί για το πως μπορεί κανείς να διαμορφώσει τη λειτουργικότητα του VMM και του FPGA, και το πως δύναται να βαθμονομηθεί το VMM, μέσω του υλικολογισμικού. Πως όμως μπορεί κανείς να αποσπάσει τα δεδομένα από τα κανάλια του VMM με τη βοήθεια του FPGA; Η περιγραφή του διαβάσματος θα απαντήσει σε αυτή την ερώτηση.

Σε γενικές γραμμές, αυτό που οφείλει να κάνει το υλικολογισμικό, είναι να αποσπάσει τα ψηφιακά δεδομένα του VMM, να τα *πακετάρει* σε μία προσυμφωνημένη μορφή, και τελικά να τα προωθήσει στο υλικολογισμικό. Το τελευταίο θα αναλύσει αυτά τα δεδομένα, κατασκευάζοντας ιστογράμματα που περιέχουν όλες τις απαραίτητες πληροφορίες - όπως π.χ. το φορτίο (PDO), ή το χρόνο (TDO) ενός γεγονότος. Αν αναλυθούν αυτά τα δεδομένα αργότερα, μπορεί να χαρακτηριστεί η αποδοτικότητα του VMM και το ανιχνευτικού συστήματος.

I.6.1 Διάβασμα του VMM

Η διαδικασία που ακολουθείται για να διαβαστούν τα ψηφιακά δεδομένα από το VMM, εξαρτάται από το πως ο χρήστης έχει διαμορφώσει το VMM προηγουμένως. Πιο συγκεκριμένα, μέσω του λογισμικού, ο χρήστης μπορεί να δώσει εντολή στο VMM να υλοποιήσει τη διεπαφή του Level-0 με το FPGA, προκειμένου να διαβαστεί το VMM με τον αντίστοιχο τρόπο. Αν αυτοί οι καταχωρητές είναι απενεργοποιημένοι, τότε το VMM θα στέλνει τα ψηφιακά δεδομένα στο FPGA με το συνεχή τρόπο. Το υλικολογισμικό, διαθέτει δύο ανεξάρτητους πυρήνες, όπου ο κάθε ένας έχει σχεδιαστεί ώστε να εξυπηρετεί το διάβασμα του VMM, ανάλογα με το πως αυτό έχει διαμορφωθεί από το χρήστη.

Συνεχές Διάβασμα

Σε αυτό τον τρόπο διαβάσματος, κάθε κανάλι του VMM αποθηκεύει τα δεδομένα που προέκυψαν από την ψηφιοποίηση των παλμών σε μία μνήμη που διαθέτει τέσσερις θέσεις. Αν ανατρέξει κανείς στο παρών Κεφάλαιο, θα δει ότι κάθε γεγονός καναλιού αποτελείται από 38 bit, τα οποία περιέχουν τις πληροφορίες για το ύψος (PDO), το χρόνο (TDO), και φυσικά τη διεύθυνση του καναλιού. Για να διαβάσει το FPGA κάποιο γεγονός, πρέπει να οδηγήσει στο VMM έναν παλμό, μέσω του διαύλου *CKTK*. Αν το VMM έχει καταγεγραμμένο γεγονός σε κάποιο κανάλι, τότε η γραμμή *DATA0* θα πάει στο λογικό ένα. Το FPGA τότε θα στείλει 19 παλμούς μέσω του διαύλου *CKDT*, και θα περάσει σε έναν εσωτερικό καταχωρητή τα δεδομένα από τις δύο γραμμές των δεδομένων που προέρχονται από το VMM, ονόματι *DATA0* και *DATA1*. Με τους 19 παλμούς CKDT, αποθηκεύονται $2 \times 19 = 38$ bit, όσο δηλαδή ένα γεγονός. Η λέξη αυτή θα αποθηκευτεί μέσα στο FPGA, το οποίο στη συνέχεια θα στείλει πάλι το έναν παλμό μέσω του *CKTK* στο VMM, ζητώντας του νέα δεδομένα. Η λογική του VMM υποδεικνύει ότι μόλις ληφθεί ο νέος παλμός από το FPGA στην σύνδεση του *CKTK*, θα οδηγηθεί στις γραμμές DATA0/1 το γεγονός από το επόμενο κανάλι του οποίου η μνήμη δεν είναι άδεια. Με αυτόν τον τρόπο, το FPGA διαβάζει "κυκλικά" και σειριακά όλα τα κανάλια της μονάδας. Όταν δεν υπάρχουν πλέον γεγονότα στις μνήμες του

VMM, τότε το FPGA δεν θα λάβει ανταπόκριση όταν στείλει πάλι παλμό στο *CKTK*, δηλαδή δεν θα δει τη γραμμή *DATA0* να έχει τιμή λογικού ένα. Αυτό σημαίνει ότι όλες οι πληροφορίες αποσπάστηκαν από το VMM, και το η λογική του FPGA θα προωθήσει όλα τα αποθηκευμένα δεδομένα μέσω της διεπαφής Ethernet στο λογισμικό.

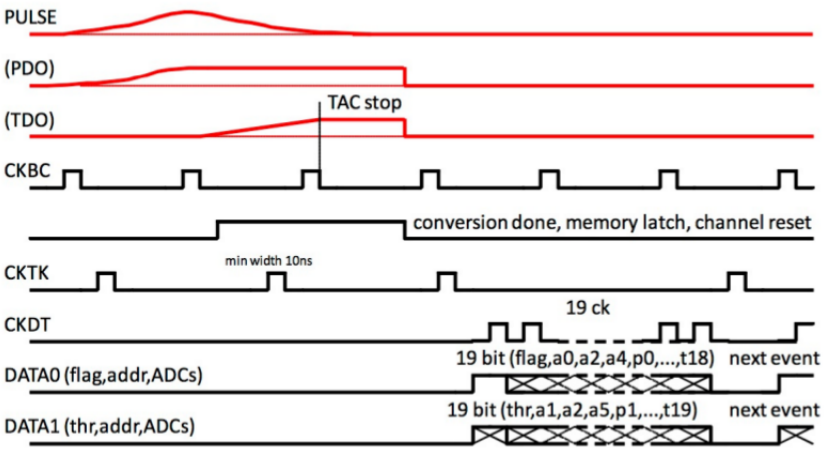


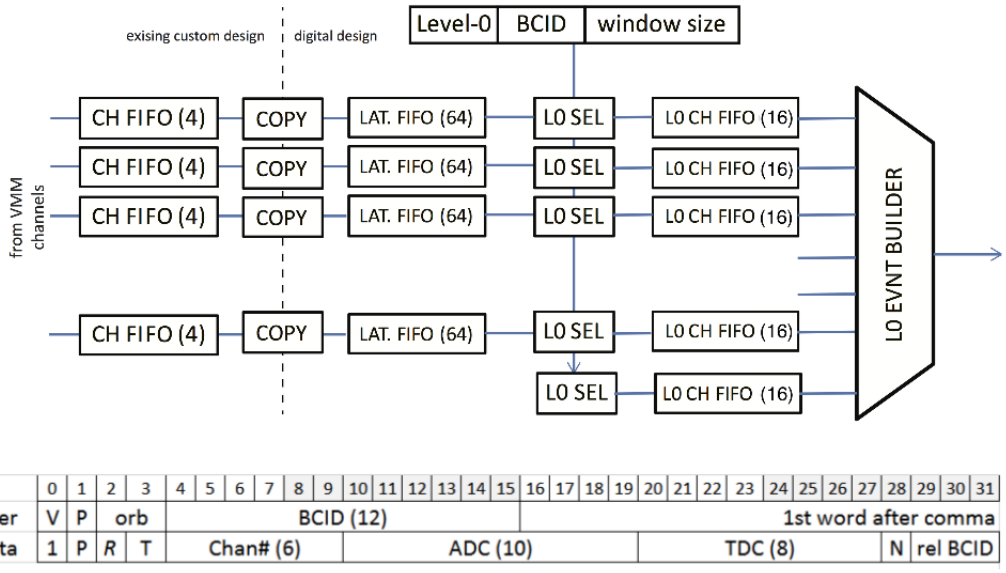
Figure I.27: Χρονοδιάγραμμα του συνεχούς διαβάσματος.

Διάβασμα Level-0

Το διάβασμα Level-0 είναι αυτό που θα χρησιμοποιηθεί και στο τελικό πείραμα του ATLAS, καθώς προσφέρει στο χρήστη τη δυνατότητα επιλογής ενός συγκεκριμένου γεγονότος που θα διαβαστεί από το VMM [10, 11]. Αυτό έχει μεγάλη σημασία για την ακεραιότητα των δεδομένων, καθώς οι ανιχνευτές Micromegas, εκτός από μιόνια, θα ανιχνεύουν και άλλα σωματίδια που ουδεμία σχέση έχουν με τα μιόνια που προέρχονται από το σημείο αλληλεπίδρασης στο κέντρο του ανιχνευτή, όπως έχει ήδη αναφερθεί στο παρών Κεφάλαιο. Η λογική του σκανδαλισμού όμως, θα μπορεί να εντοπίζει τα μιόνια που έχουν ενδιαφέρον για την τελική ανάλυση των δεδομένων, και έτσι μέσω της λογικής του Level-0, μόνο οι ψηφιακές πληροφορίες που είναι συνυφασμένες με τα έγκυρα γεγονότα θα οδηγούνται έξω από το VMM. Τα υπόλοιπα δεν θα διαβάζονται ποτέ, και τελικά θα απορρίπτονται.

Όταν ένας παλμός εισόδου ξεπεράσει το κατώφλι του διευκρινιστή του VMM, τότε θα λάβει ένα χρονικό χαρακτηρισμό, σύμφωνα με την τιμή του μετρητή Gray που αυξάνεται με κάθε ακμή του CKBC (βλ. Σχ. I.12), δηλαδή αυξάνει κατά ένα ανά 25 ns. Αυτό το γεγονός θα προωθηθεί στη συνέχεια σε μία μνήμη βάθους τεσσάρων γεγονότων. Μέχρι αυτό το σημείο η λογική είναι ίδια με του συνεχούς διαβάσματος, όμως στο Level-0, αυτό το γεγονός θα προωθηθεί σε μία δεύτερη μνήμη, όπου θα περιμένει μέχρι να επιλεγεί για να προωθηθεί εκτός του VMM. Προκειμένου να

επιλεγεί ένα γεγονός, πρέπει να έρθει στο VMM το σήμα ονόματι *Level-0* που οδηγείται μέσω του διαύλου CKTK. Όταν το VMM λάβει το Level-0, θα του δώσει και αυτό έναν χρονικό χαρακτηρισμό, όπως πριν με το γεγονός του καναλιού. Μόλις συμβεί αυτό, το VMM θα ”ψάξει” για ένα γεγονός με αυτήν την τιμή, μείον μία προκαθορισμένη τιμή που διαμορφώνεται από το χρήστη. Το Level-0 επομένως, ”ψάχνει” για ένα συγκεκριμένο γεγονός που έχει καταγραφεί στο παρελθόν από το VMM. Θέτοντας σωστά την τιμή της αφαιρούμενης ποσότητας από τη χρονική τιμή του Level-0 για κάθε VMM στο NSW, θα επιλέγονται μόνο τα σωστά γεγονότα από τη μονάδα. Αν το Level-0 που εστάλη στο VMM βρει κάποιο γεγονός, τότε στο FPGA προωθείται κατ’ αρχάς ένα κομμάτι πληροφορίας που περιέχει τα δεδομένα του χρόνου, και διαδοχικά τα γεγονότα από όλα τα κανάλια που έχουν αυτό το χρονικό προφίλ.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
header	V	P	orb		BCID (12)												1st word after comma															
hit data	1	P	*R*	T	Chan# (6)						ADC (10)										TDC (8)								N	rel BCID		

Figure I.28: Πάνω: Αρχιτεκτονική της λογικής του Level-0 μέσα στο VMM. Κάτω: Δεδομενόγραμμα του Level-0 [11].

Η επικοινωνία του Level-0 υλοποιείται μέσω του πρωτοκόλλου *8b/10b*. Όταν το VMM δεν στέλνει δεδομένα στο FPGA (ή στο ROC, για το τελικό πείραμα), τότε στέλνει τους λεγόμενους *Comma Characters*, που είναι προκαθορισμένες 10-bit λέξεις που χρησιμοποιούνται για να φέρουν σε φάση τον πομπό και τον δέκτη (βλ. αντίστοιχο Παράρτημα και το παρών Κεφάλαιο). Η λογική του FPGA έχει ως εξής: Προωθείται κατ’ αρχάς συνεχόμενα στο VMM το ρολόι *CKDT* (σε αντίθεση με το συνεχές διάβασμα, όπου το FPGA έστελνε παλμούς στο δίαυλο *CKDT* μόνο όταν ήξερε ότι υπήρχε γεγονός), και το FPGA προβαίνει σε δειγματοληψία των δεδομένων από τις γραμμές *DATA0/1*. Τα δεδομένα αυτά, όταν το VMM βρίσκεται σε αδρανή κατάσταση, είναι ένα προκαθορισμένο μοτίβο, που περιέχει τους χαρακτήρες ευθυγράμμισης

φάσης. Ένας καταχωρητής δέχεται αυτά τα δεδομένα, και μία λογική ελέγχει συνεχώς τα περιεχόμενα του. Όταν αναγνωρίζονται οι χαρακτήρες μέσα στον καταχωρητή, τότε ο μηχανισμός ”κλειδώνει” και η λογική του FPGA/δέκτη βρίσκεται πλέον σε φάση με τα δεδομένα του VMM/πομπού. Όταν το VMM στείλει πραγματικά δεδομένα, τότε η λογική της ευθυγράμμισης θα γνωρίζει ακριβώς πότε ο καταχωρητής θα περιέχει μία έγκυρη λέξη, προωθώντας τη στη συνέχεια στον αποκωδικοποιητή του 8b/10b. Αυτός ο πυρήνας δέχεται 10-bit λέξεις και σύμφωνα με το πρωτόκολλο 8b/10b τις μετατρέπει σε byte. Αυτά τα byte γράφονται σε μία μνήμη. Όταν λήξει η μετάδοση του πακέτου (η μορφή του οποίου απεικονίζεται στο Σχ. I.28), τότε το VMM ξεκινάει πάλι να στέλνει comma characters. Τότε η λογική του FPGA αντιλαμβάνεται ότι το διάβασμα του γεγονότος τελείωσε, και τα δεδομένα προωθούνται στο λογισμικό μέσω της διεπαφής Ethernet, όπως περιγράφηκε προηγουμένως.

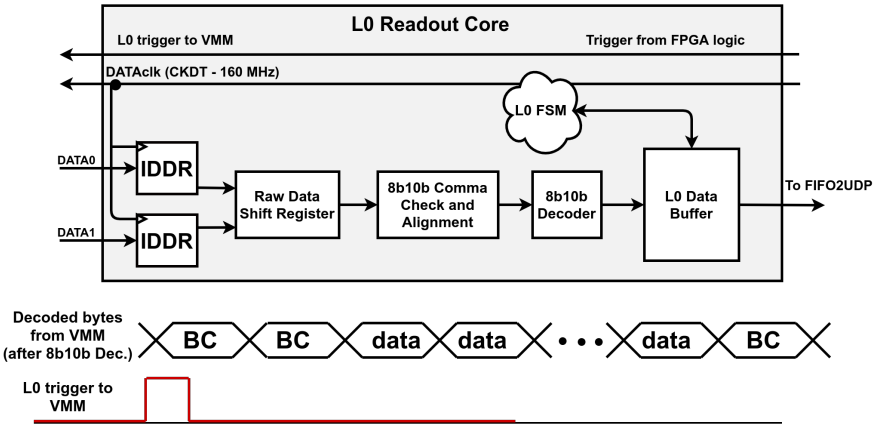


Figure I.29: Αρχιτεκτονική του Level-0 Readout Module του FPGA. Διακρίνονται όλα τα υποσυστήματα που έχουν περιγραφεί παραπάνω.

Part I

Introduction to the New Small Wheel Upgrade, the Micromegas Detector and the VMM ASIC, and to Field-Programmable Gate Arrays

Chapter 1

The Large Hadron Collider and the ATLAS Experiment

In this Chapter, a short introduction to the Large Hadron Collider (LHC) and the Toroidal LHC ApparatuS (ATLAS) experiment will be made. After this, the New Small Wheel (NSW) upgrade of the ATLAS detector will be described, alongside the motivation behind its development and intent of deployment.

1.1 The Large Hadron Collider

For a high energy physicist, the accelerator is the equivalent of a microscope [1]. The first accelerator ever constructed, may very well have been the cathode ray tube, that Röntgen built in 1895 in order to generate X-rays. The said device was accelerating electrons using a voltage difference at energies of a few keV[1]. Today, the energy levels that these devices can reach, are in the order of a few TeV, and do not only accelerate electrons, but also protons or heavy ions. Modern apparatuses accelerate a *beam* of particles either in a straight line, or around a circle (or ring), with the help of electromagnets that produce strong enough fields to provide the required amount of energy to help the particles reach unprecedented speeds, and at the same time guide them with significant accuracy. These energetic beams either collide with a fixed target, or with another beam, in order to produce other, much heavier particles, that researchers study.

[1] The *electronvolt*, or eV, is an energy measurement widely used in high energy physics, and is the amount of energy than an electron has, when it is being accelerated by a voltage difference of one volt (1 V).

The Large Hadron Collider (LHC) [14] is the world's largest and most powerful particle accelerator. It was built by the European Organization for Nuclear Research (CERN) between 1998 and 2008, in collaboration with over 10000 scientists and hundreds of universities and laboratories, from more than 100 countries. It is of circular design, accelerates protons or heavy ions, and lies in a tunnel 27 kilometers in circumference, as deep as 175 meters beneath the France–Switzerland border near Geneva. The first beam of the LHC was delivered in September 2008 and the first collisions were produced about a year later. The aim of the LHC is to reveal physics processes beyond the Standard Model with proton-proton collision energies of up to 14 TeV. A general overview of the entire complex is shown in Figure 1.1.

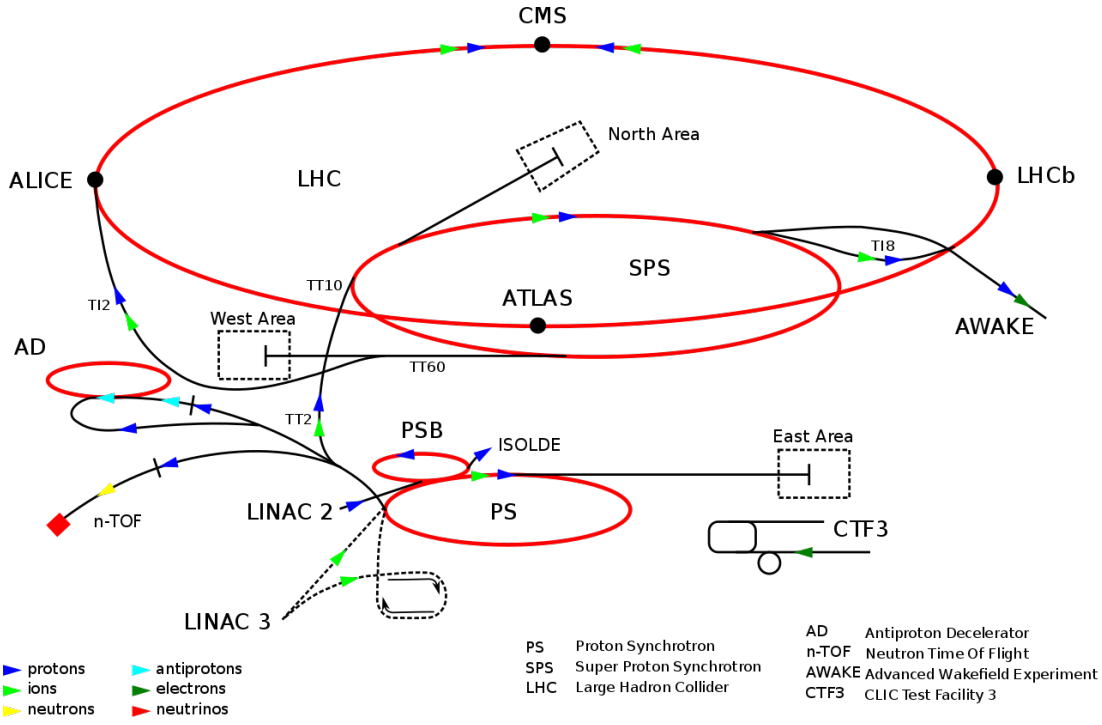


Figure 1.1: Drawing of the LHC complex. The protons are first being accelerated in the Proton Synchrotron (PS) ring, before entering the Super Proton Synchrotron (SPS). After the SPS, they enter the LHC. The beams collide in four dedicated areas, where the main experiments (ATLAS, CMS, ALICE, LHCb) are installed, in order to detect the collision fragments.

A series of progressively larger ring-shaped accelerators boost the speed of the protons, which originate from a simple hydrogen bottle. Across all accelerating stages, superconducting magnets guide and provide energy to the protons, in order to eventually reach the desired energy. The protons finally end up in the large collider of the LHC, which splits the proton beam in two, and circulates both beams, one clockwise and the other anticlockwise for several hours. These two beams, collide in

four designated places, and due to the energetic proton-proton interactions, heavy short-lived particles are produced, which provide valuable information about the structure of matter to the researchers at CERN. In order to study the behavior of the collision fragments, *detectors* are placed in the positions where the beams collide. These apparatuses are sophisticated devices that are able to analyze the trajectory and the energy of particles that traverse them. The two general-purpose detectors of the LHC are the Toroidal LHC ApparatuS (ATLAS) [5], and Compact Muon Solenoid (CMS) [15], which announced the discovery of the Higgs boson in 2012 [16, 17].

Since the high energy physics experiments rely heavily on statistics, the increase of *reaction rate* is considered to be of utmost importance. The higher the reaction rate, the more data are recorded. Therefore, events-of-interest that may occur rarely, because they are generally superseded by other more frequent processes, can be detected. For fixed target experiments, the reaction rate ϕ depends on the rate n that the beam particles collide with the target, the cross section[2] σ of the reaction in question, and the target's thickness d [1]. By combining these, one gets the unit of *Luminosity*:

$$\phi = \sigma \mathscr{L} \tag{1.1}$$

Another way to define the concept of luminosity is as follows [2]:

$$\frac{dR}{dt} = \mathscr{L}\sigma_p \tag{1.2}$$

Where $\frac{dR}{dt}$ is the number of events per second, and σ_p is the cross section of a given type of interaction. The luminosity's unit is therefore $\mathrm{cm}^{-2}\mathrm{s}^{-1}$. In modern high energy physics experiments, the method of colliding an energetic beam on a stationary target is generally not used, since with two colliding beams one can achieve much higher energies and therefore can explore a larger variety of natural processes. Hence, the definition of luminosity will be different than the one provided in equation 1.1. For colliding beams, it represents a measure of how many particles are packed in the beam, per cm^2. The LHC machine for instance, is mostly a proton-proton accelerating apparatus, that packs its beam particles in *bunches*; groups of tightly packed protons, separated in space and time. Assuming the beam profiles are Gaussian-shaped, with N_1 and N_2 the amount of particles in each beam, σ_x, σ_y the traverse and longitudinal dimensions of the beam, f the revolution frequency of the beams[3], and N_b the number

[2]The cross section is a unit of measurement of the odds that a reaction can occur. It is measured in *barn* b, where $1\,\mathrm{b} = 10^{-28}\,\mathrm{m}^2 = 100\,\mathrm{fm}^2$.

[3]The LHC is operating at a frequency of $\sim 40\,\mathrm{MHz}$.

of bunches, one gets the following formula [2]:

$$\mathscr{L} = \frac{N_1 N_2 f N_b}{4\pi\sigma_x\sigma_y} \tag{1.3}$$

Which is the definition of instantaneous luminosity for two Gaussian beams that collide head-on[4]. Also, it is worth mentioning that the luminosity in the LHC does not remain constant during a physics run, but decays due to the degradation of intensities and radiation emittance from the circulating beams. The main cause of the luminosity decay during nominal LHC operation is the beam loss from collisions. If all degradation effects are taken into account, a luminosity lifetime of $\tau_L = 14.9\,\text{h}$ can be estimated [4]. In any case, it is evident from equations 1.2 and 1.3 that the more the protons in the bunch, and the more focused the beam is, more interactions per second can be achieved, which leads to a more efficient measurement of rare physics processes.

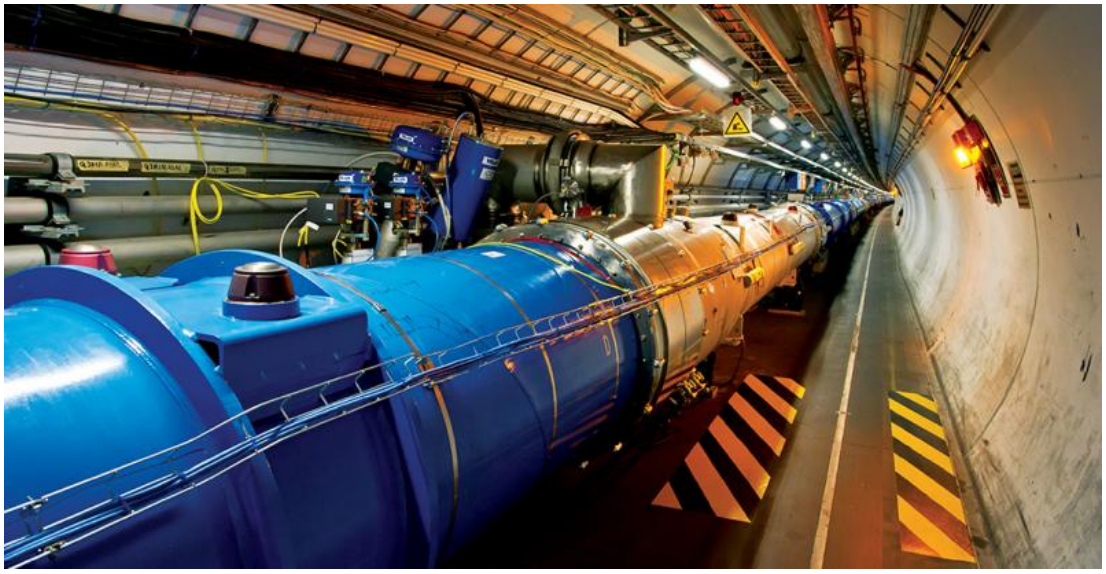


Figure 1.2: The LHC beam pipe, where the protons are being accelerated.

1.1.1 Increasing the LHC's Luminosity

In its first period of operation from 2010 to 2012, the LHC has delivered an integrated luminosity of $\sim$ 29 fb^{-1}, leading to extraordinary results from the experiments, including the discovery of the Higgs boson. During that period, also referred to as Run-1, the average instantaneous luminosity was about $7.5 \times 10^{33}\,\text{cm}^{-2}\text{s}^{-1}$, with a

[4]The reader should keep in mind that the LHC collides its bunches at a small angle, a fact that slightly changes the definition of luminosity for the said experiment.

center-of-mass energy of $\sim 7-8\,\text{TeV}$. After a two-year shutdown, the LHC restarted with Run-2, between 2015 and 2018, to deliver proton-proton collisions at nominal luminosity of $1 \times 10^{34}\,\text{cm}^{-2}\text{s}^{-1}$ with energies of 13 TeV.

The future of the LHC will yield larger luminosities and increases in energy. To be precise, Run-3 $(2021-2023)$, also referred to as *Phase-I*, will see an increase in instantaneous luminosity, with $\mathscr{L} = 2.5 \times 10^{34}\,\text{cm}^{-2}\text{s}^{-1}$. Run-4 and 5 $(2026-2038)$, or *Phase-II*, will see the LHC deliver a luminosity of $5-7.5 \times 10^{34}\,\text{cm}^{-2}\text{s}^{-1}$ and a center-of-mass energy of 14 TeV. The upgrade plans of the LHC are shown in Figure 1.3.

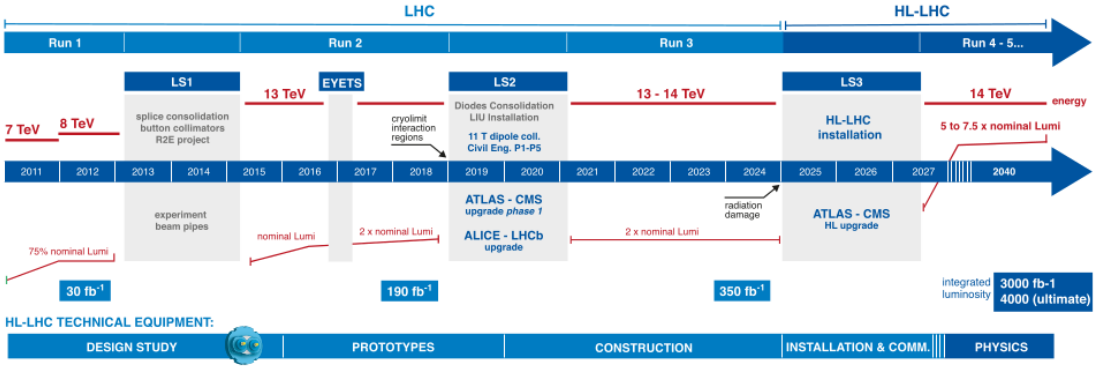


Figure 1.3: The plan towards the High Luminosity LHC.

All these are of course desirable from a physics point of view, since upgrades in luminosity and energy will lead to a better chance of observing rare physics events. However, since larger luminosities also mean an increased particle flux, the LHC experiments will have to be upgraded as well, in order to cope with the increased data-flow throughput (from the electronics and data acquisition point of view), and the increased background rate (from a detector point of view). This is exactly the motivation behind the New Small Wheel (NSW) upgrade of the ATLAS detector, which will now be described.

1.2 The ATLAS Detector

The ATLAS (**A** **T**oroidal **L**HC **A**pparatu**S**) general-purpose detector [5], has been designed to cover the needs of several experiments that take place in the LHC, such as the search (and eventual discovery in 2012) for the Higgs boson, studies on supersymmetry, or dark matter searches. The other similar apparatus of the LHC is the CMS experiment [5], situated on the opposite side of the LHC ring. The ATLAS detector is probing $p-p$ and $Pb-Pb$ or $p-Pb$ collisions. It is cylindrical in shape,

with a length of 45 m, and a diameter of about 25 m. It weighs about 7000 tons, and is the largest detector by volume every built. Since 2012, over 3000 researchers from 38 countries collaborate to support and upgrade the detector, and analyze its data to discover new physics processes. A graphical representation of ATLAS is shown in Figure 1.4.

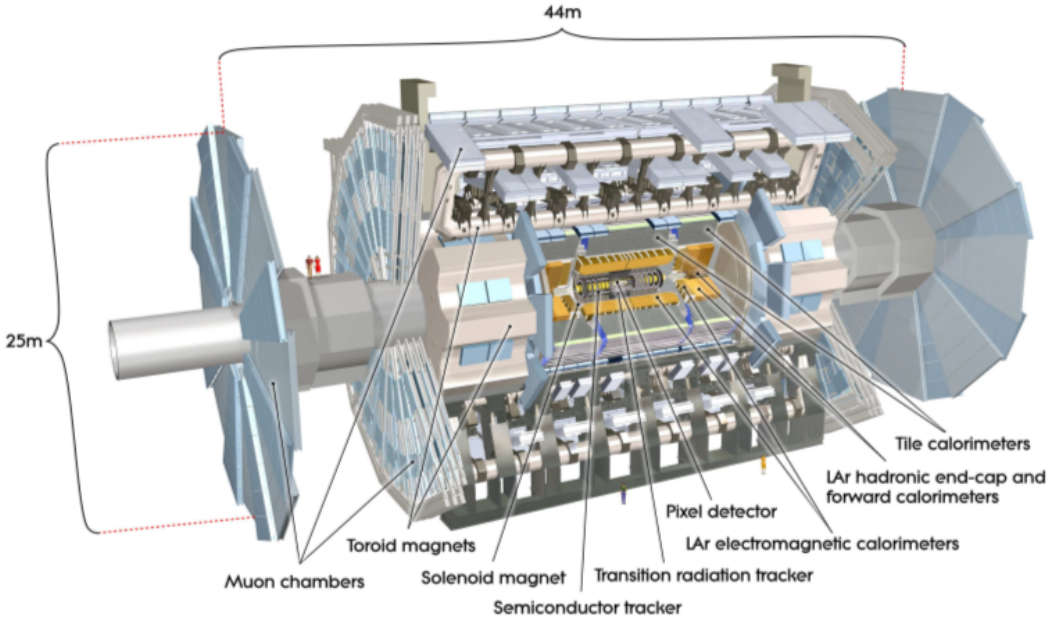


Figure 1.4: Overview of the ATLAS detector. The muon subsystem which is more closely related with this dissertation, lies in the outer layers of the device. The LHC's beam enters the detector through the center of its end-caps from both sides, and the bunches collide in the core of the detector.

The proton (or heavy ion) beams enter the detector through its end-caps, and collide almost head-on in its center, producing new particles. The different subsystems of ATLAS, organized in layers, measure the momenta of all particles that are being produced by the collisions; that is, they have been arranged and designed in such fashion as to provide precise energy and track estimations of the collision products. This is also assisted by the toroidal magnets, that bend the trajectory of charged particles, which are affected in a different way, depending on their momentum and charge. The entire detector resembles the shape of a barrel, and because of this, it is common to refer to specific areas of the detector as *barrel region*, or *end-cap region*. These two aforementioned regions, lie in so-called different *pseudorapidity* regions. Pseudorapidity is denoted as η. Given cylindrical coordinates, which are convenient to use because of the detector's shape, the longitudinal (or radial) angle with respect to the beam line is denoted as θ, whereas the azimuthal angle has the symbol of ϕ. Hence,

the pseudorapidity is defined as:

$$\eta \equiv -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \tag{1.4}$$

Thus, a particle that traverses an area of the detector with high pseudorapidity, is moving almost parallel to the beam line, while smaller values of $|\eta|$, correspond to larger escape angles ($90° - 45°$) with respect to the beam line. The end-cap region of the detector therefore lies in large pseudorapidity regions, and the barrel region in region with smaller $|\eta|$ values.

1.2.1 The Detector's Subsystems

The main subsystems of ATLAS are the following:

- The Magnet System
- The Inner Tracker
- The Calorimeters
- The Muon Spectrometer

The inner magnet system is a superconducting solenoid (also referred to as *Central Solenoid* [18]), which is situated around the inner tracker. It produces a magnetic field of $2\,\mathrm{T}$, and allows the inner tracker to analyze the tracks that have just escaped the interaction point. Besides the central solenoid, there is also the *air-core Barrel Toroid* [19], alongside the *End-Cap Toroids* [20], situated around the calorimeters, that provide the necessary magnetic fields (peaking at $4.1\,\mathrm{T}$ [4]) for the muon spectrometer to analyze the momenta of the incident muons. The *Inner Tracker* [21] has a length of $6\,\mathrm{m}$ and a diameter of $2\,\mathrm{m}$, covering areas with $|\eta| < 2.5$, and is the first detector subsystem which the particles that originate from the interaction point traverse. It is composed of three different detector technologies, namely the silicon Pixel detector (Pixel), the Semi-Conductor Tracker (SCT) and the Transition Radiation Tracker (TRT). Even though about 1000 particles are produced every $25\,\mathrm{ns}$ at the interaction point, the inner tracker is able to determine their exact point of origin[5], and their trajectory. The Pixel system is mainly in charge of the accurate measurement of vertices, whereas the SCT's primary functionality is the precise measurement of the particle momenta, with the two technologies providing at least three to four particle track reference points respectively [4]. The TRT system, makes use of the transition radiation, which is

[5]This point is also called a *vertex*.

emitted by a particle when it crosses into media with different dielectric properties [1], thus enhancing the tracking capabilities of the entire inner tracking system.

The *ATLAS Calorimeter* is responsible for the accurate measurement of the energy and trajectory of electrons and photons or jets. Due to its principle of operation, it is able to identify particles, and also contribute to the muon momentum reconstruction [22]. The calorimeter's tasks are rather challenging, given the fact that there are multiple collisions at each bunch-crossing (every $25\,\text{ns}$), and due to the energy of the beams themselves, as the particles that are being produced cover a wide range of energies (extending from a few GeV up to the TeV scale). Therefore, the calorimeter would not only have to be able to distinguish between two separate collision events, either in space or in time (i.e. be immune to *pile-up* effects), but also have the necessary space and energy resolution to identify particles and reconstruct their tracks. Usually, calorimeters are either hadronic (that is, they detect hadrons and baryons), or electromagnetic (they detect leptons and photons). The hadronic calorimeter is divided into three parts, depending on the pseudorapidity range it covers. There is the barrel region ($|\eta| < 1.7$), the end-cap region ($1.5 < |\eta| < 3.2$), and parts that are very close to the beam pipe ($3.1 < |\eta| < 3.9$). Each of the aforementioned parts is constructed in a different way. For example, the barrel calorimeter is made of steel (to absorb particles) and scintillators, which determine the incident radiation's energy [5]. The rest of the hadronic calorimeters are made of liquid argon (known as *LAr*), while the electromagnetic calorimeter is comprised of two disks in the end-caps which cover two pseudorapidity regions ($1.375 < |\eta| < 2.5$ and $2.5 < |\eta| < 3.2$). There is also the barrel electromagnetic calorimeter, that covers a region of $|\eta| < 1.475$. Both are made of lead plates which absorb the particles, and Kapton electrodes that contribute to track reconstruction.

The ATLAS Muon Spectrometer

The *ATLAS Muon Spectrometer* was designed to perform high-precision track reconstruction on final-state muons, which provide robust physics signatures, given the LHC's experimental conditions. The muon spectrometer features stand-alone triggering and adequate momentum measurement capabilities, over a wide range of transverse momentum, pseudorapidity, and azimuthal angle [23]. A graphical representation of the subsystems comprising the muon spectrometer are shown in Figure 1.5.

The particles that have not been absorbed by the calorimeters, are usually muons ($\mu^{\pm}$), which are second-generation leptons, and are either positively or negatively charged. The muon has a mass of about $106\,\text{MeV}/\text{c}^2$, while its closely-related electron,

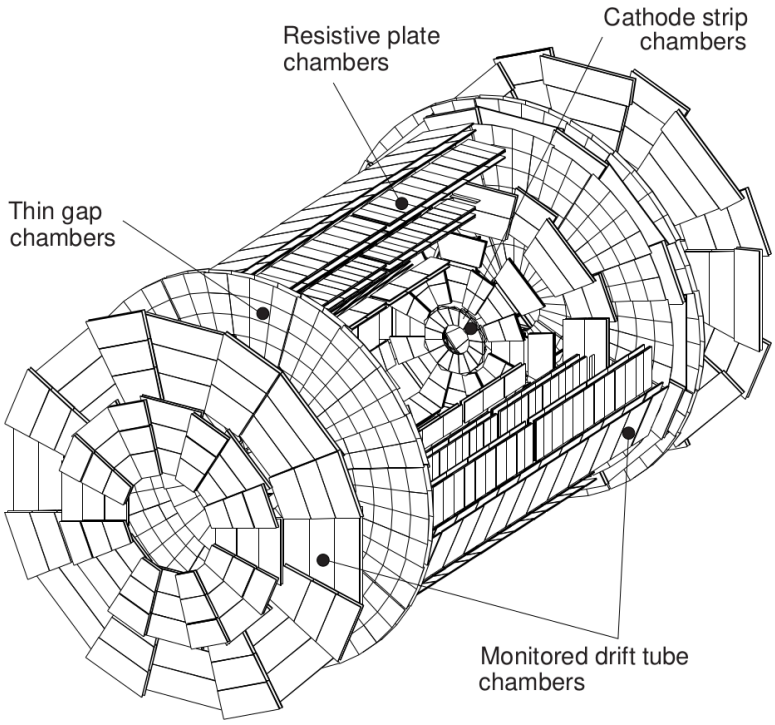


Figure 1.5: Overview of the ATLAS detector's muon subsystem. It is comprised of Cathode Strip Chambers (CSCs) in the inner end-cap region, Monitored Drift Tubes (MDTs) covering the barrel and end-cap region. CSCs contribute to tracking, while MDTs contribute to tracking and triggering. Resistive Plate Chambers (RPCs) are in the barrel region, and Thin Gap Chambers (TGCs) are at the end-caps, and provide trigger data to ATLAS. The outermost end-cap region is called Big Wheel (BW), while the innermost end-cap region, is called Small Wheel (SW). The muon system covers the entire outer layer of ATLAS, thus defining its final dimensions.

has a mass of just $511\,\mathrm{keV/c^2}$ [6]. Muons are generally able to penetrate into large volumes of matter (this is another reason why muons are detected at sea-level, apart from relativistic effects), therefore, in order to detect muons that originate from the interaction point, it is common to construct muon detecting systems on the outer parts of the detector, and this is why the muon spectrometer of ATLAS covers the entire outer layer of the detector. An overview of an event in ATLAS is shown in Figure 1.6.

The muon spectrometer makes use of the magnetic deflection of muon tracks induced by a system of three large superconducting air-core toroid magnets, that provide high-resolution muon momentum measurement. For $|\eta| < 1.4$, the bending of muons is being performed by the toroidal magnet that is around the barrel, while for regions $1.6 < |\eta| < 2.7$, the tracks are being bent by the two end-cap toroids. The detecting media comprising the muon spectrometer are the following:

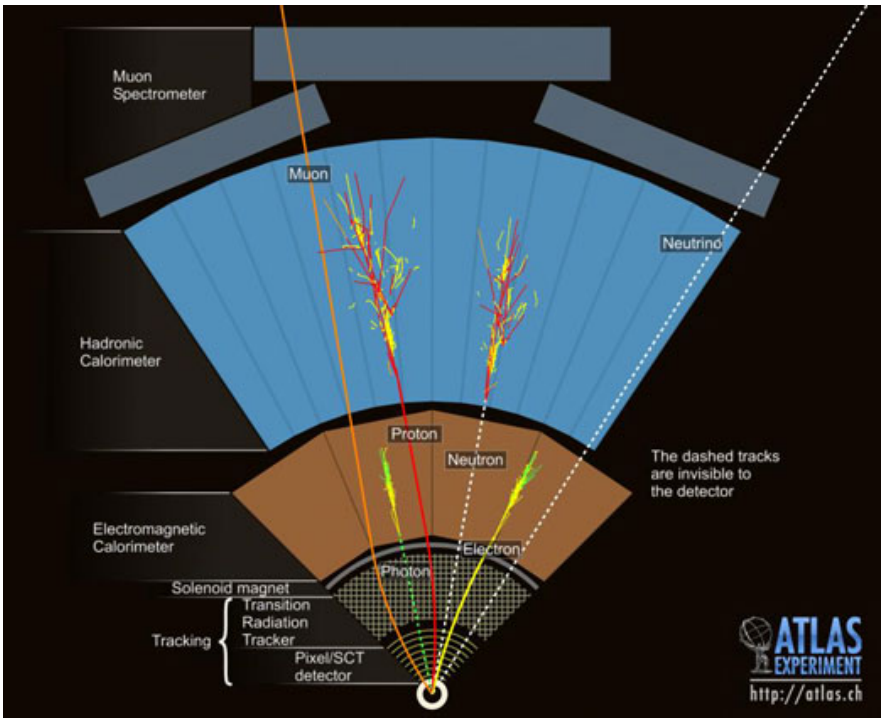


Figure 1.6: Graphical representation of an event occurring in ATLAS. Most particles stop at the calorimeters, but muons traverse the entire detector, and are being detected by the muon spectrometer.

- Cathode Strip Chambers (CSCs)
- Monitored Drift Tubes (MDTs)
- Resistive Plate Chambers (RPCs)
- Thin Gap Chambers (TGCs)

The CSCs are located in the innermost region of the end-caps (also referred to as *Small Wheels (SWs)*), covering the area of $2.0 < |\eta| < 2.7$, and are responsible for reconstructing the muon tracks with high precision. They are tilted with respect to the beam line (at about $11.59°$ angle), so that their planes are generally orthogonal to particles originating from the IP [4]. They are multi-wire proportional chambers, operating with a gas mixture of argon and carbon dioxide (20% of the gas), and a voltage of $1800 - 1900\,\text{V}$, in order to achieve an adequate gain throughout all the detector. The CSCs are comprised of two cathode layers of readout strips that perform charge interpolation, while a set of strips that run perpendicular with respect to the anodes provide the second coordinate measurement. The detector is able to reconstruct muon tracks with about $\sigma = 60\,\mu\text{m}$ precision. The small gas gap of the layers allows for fast

charge collection (less than 30 ns), which is highly desirable, because of the increased particle flux in the ATLAS high eta regions.

The MDTs cover the entire muon spectrometer of ATLAS, apart from the high pseudorapidity region of the SW, where the CSCs are installed. Each MDT chamber is comprised of a stack of tubes (arranged in two or sometimes even three layers), each filled with a non-flammable gas mixture of argon and carbon dioxide (7% of the gas). Each of the metal tubes is about 30 mm in diameter, and serves as an anode. Through the center of the tube, a slim cathode wire is kept at a potential of 3080 V, thus creating a high-enough electric field that produces avalanche effects necessary to collect charge related to muons passing through it. By combining the drift time measurements of the ionization electrons in each drift tube, a single MDT chamber creates a six-point muon segment, that can be reconstructed with high precision $\sigma = 40$ μm [4].

The two other detector technologies involved (i.e. RPCs and TGCs), cover different pseudorapidity regions, and due to their design, are able to provide fast and precise muon trigger information to ATLAS, thus completing the tracking and triggering scheme of the spectrometer.

1.3 The New Small Wheel Upgrade

As mentioned in Subsection 1.1.1, the luminosity of the LHC will be increased in the forthcoming upgrades, resulting in a significant increase in particle flux. This will pose a challenge to the detectors, which may be hampered by pile-up effects, thus degrading their performance. The front-end electronics will also be negatively impacted, as more events means more throughput, and if they are not able by-design to sustain a flawless data acquisition procedure given a more challenging event rate, they are deemed insufficient for future LHC runs.

To be precise, simulation studies [7] that have been performed on the muon end-cap region, alongside analysis of present data, have indicated that:

- The performance of the muon tracking chambers in the end-cap region degrades with the expected increase of cavern background rate. An extrapolation from the observed rates at the lower luminosity conditions of the 2012 run to high luminosity and high energy conditions indicates a substantial degradation of tracking performance, both in terms of efficiency and resolution in the inner end-cap station, which is comprised of CSCs and MDTs. In the SW region the luminosity increase will lead to a particle rate of up to $15\,\mathrm{kHz/cm^2}$, which the CSCs and MDTs will not be able to handle efficiently [4].

- The Level-1 muon trigger in the end-cap region is based on track segments in the TGC chambers located after the end-cap toroid magnet. The transverse momentum, p_T, of the muon is determined by the angle of the segment with respect to the direction pointing to the interaction point. A significant part of the muon trigger rate in the end-caps is background. Low energy particles, mainly protons, generated in the material located between the Small Wheel and other areas, produce fake triggers by hitting the end-cap trigger chambers at an angle similar to that of real high p_T muons. An analysis of 2012 data demonstrated that approximately 90% of the muon triggers in the end-caps are fake. In consequence, as the particle flux increases with the foreseen upgrades of the LHC, this effect will become even more pronounced.

Given these facts, the solution is to replace the existing small wheel region with the so-called *New Small Wheel (NSW)* during the second long shutdown, ending in 2021 (as reported at the time of writing of this dissertation). Comprised of two detector technologies, the *small-strip Thin Gap Chambers (sTGC)* and *Micromegas (MM)* detectors, the NSW will be able to maintain the current low-luminosity performance of the SW in the future high-luminosity environment imposed by the LHC's upgrades. It will be able to measure the transverse momentum (p_T) of 1 TeV muons with a precision of 10% for $|\eta| < 2.7$, and manage to provide precise enough trigger information to ATLAS, and suppress the fake triggers originating from the current end-cap region. An illustration of this desirable feature is shown in Figure 1.7

The NSW will be able to provide *online* trigger information with an angular resolution better than 2 mrad, in the full pseudorapidity region of $1.3 < |\eta| < 2.7$, with a latency of less than 1025 ns. This will be achieved by the detectors themselves, and the NSW electronics. Trigger primitives/data originating from the front-end chips will be processed by NSW-dedicated trigger processing electronics situated in the back-end, and trigger candidates (or segments) will be created fast enough to be combined with the BW's trigger information, in order to distinguish real muons originating from the interaction point from noise. Also, apart from the trigger, the detectors' design allows for precise muon track reconstruction, as they provide a position resolution in the precision coordinate better than 50 μm, when all the tracklets from the detector layers are combined [4].

All these will be achieved by a redundant triggering and tracking scheme, where the two detector technologies participate in both. The sTGCs are mainly triggering chambers, but they also contribute to precision tracking. The MMs on the other hand, are tracking detectors, but also assist in providing sufficient information to the NSW's triggering system. The detectors are segmented into sectors, sixteen per wheel, and are arranged

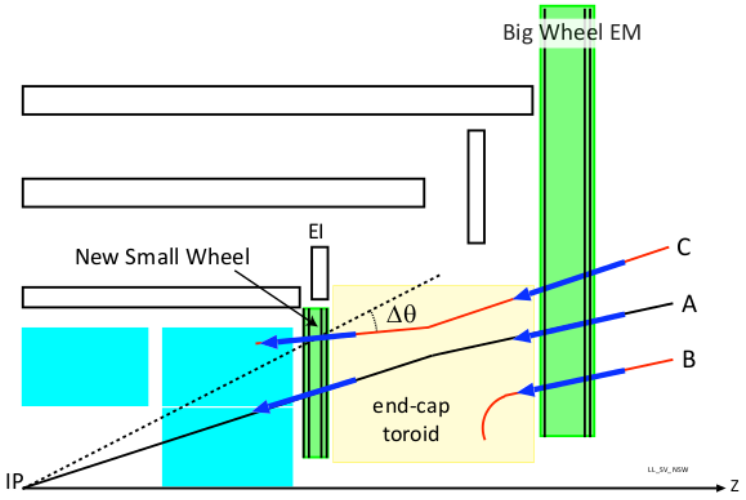


Figure 1.7: Graphical representation of the ATLAS end-cap muon trigger system, after the NSW upgrade. The already existing BW accepts all inbound particle tracks that are shown. With the installation of the NSW, only track *A* is accepted, since this is the only one that passes both sub-detector criteria. The *B* track will be rejected, since the NSW will not even detect it, and track *C* will be rejected due to the NSW's triggering scheme, since its vector at the NSW area does not point to the interaction point, a fact not true for the BW region.

in such a way as to create a disk with a diameter of about 9 m. Each "slice" is comprised of four quadruplets. The sTGCs are placed on the two outer parts of the sector, while the two MM quadruplets (or *wedges*) are on the inner part of the NSW sector. A graphical representation of the detector is shown in Figure 1.8.

A muon traversing the NSW will therefore excite the eight Micromegas detector layers, and the corresponding sTGC sectors, creating tracking and triggering data that will be mediated by the on-detector electronics to the back-end Data AcQuisition (DAQ) system of ATLAS. Both detector technologies are gaseous proportional chambers, operating under high voltage. The sTGC has three readout elements: wires, strips and pads. Each layer is filled with a mixture of carbon dioxide and n-pentane, and is operated under a voltage of 2.85 kV. Due to its design, the sTGC has excellent bunch-crossing (BC) identification capabilities, and by also taking its strip segmentation (3.2 mm) into account, the chamber is ideal for tagging an incident muon for triggering purposes. Its strip and wire pitch (1.8 mm for the wires), also contribute positively to the precision tracking properties of the detector. The Micromegas on the other hand, has only one readout element; its strips, with a pitch of 400 μm. Their finer segmentation makes the MM ideal for muon track reconstruction. The Micromegas detector will be described in greater detail in the following Chapter, as it is more closely related

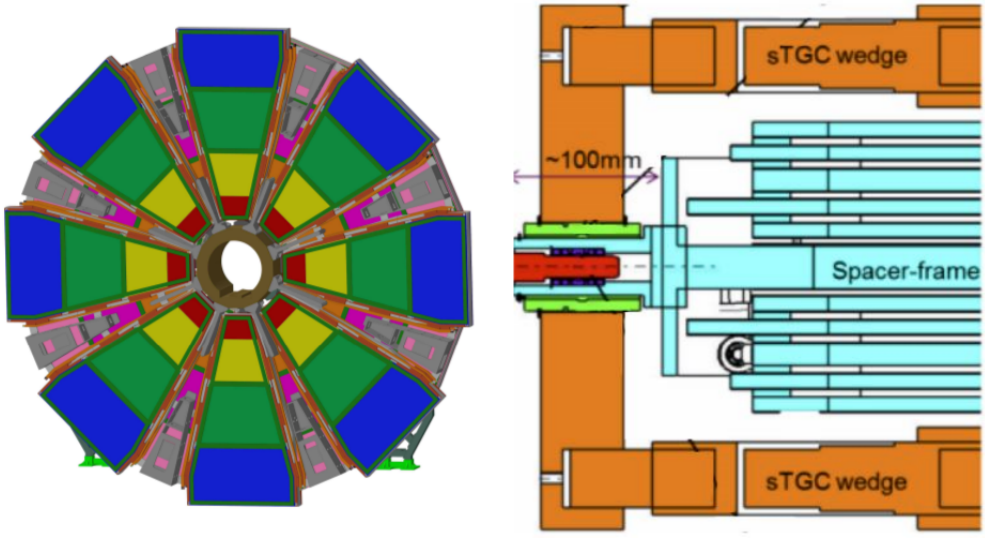


Figure 1.8: *Left:* One of the two NSW sides. The MM and sTGC sectors comprise each wedge. *Right:* Cross-section of an NSW sector. There is one sTGC quadruplet/wedge on each outer part of the sector, while the MM quadruplets are on the inside part [8].

to the work of this dissertation. A rough graphical representation of both detector technologies is provided in Figure 1.9.

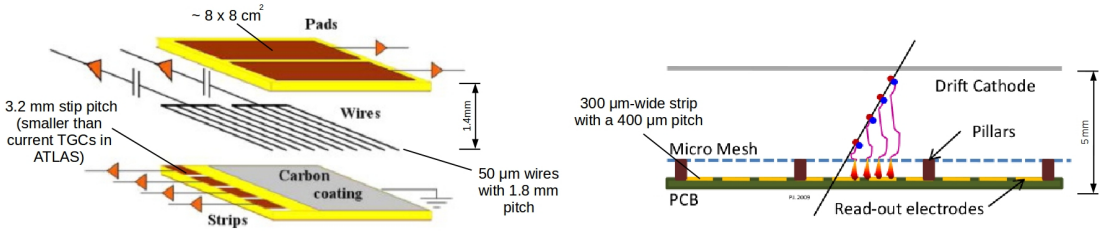


Figure 1.9: *Left:* Drawing of the sTGC detector. *Right:* Drawing of the MM detector.

Conclusions

This Chapter began with an overview of the LHC, and the concept of luminosity was defined. Luminosity is a quantification of how many events per second are produced in a collider experiment, and in order to be able to probe new physics processes, the LHC plans to boost its luminosity in the following years, by focusing its beams. This will increase the particle generation rate, affecting the performance of the LHC's detectors. The ATLAS experiment is one of these detectors, and because of the increase in particle flux, the inner end-cap region of the muon spectrometer is foreseen to experience a drop in its performance. For this reason, the inner end-cap will be replaced by the New Small Wheel, comprised of two detector technologies that will be able to overcome the

current system's limitations. The NSW will be installed at around the end of the second long shutdown in 2021 and at the time of writing of this dissertation, was in the phase of surface commissioning, as can be seen in Figure 1.10.

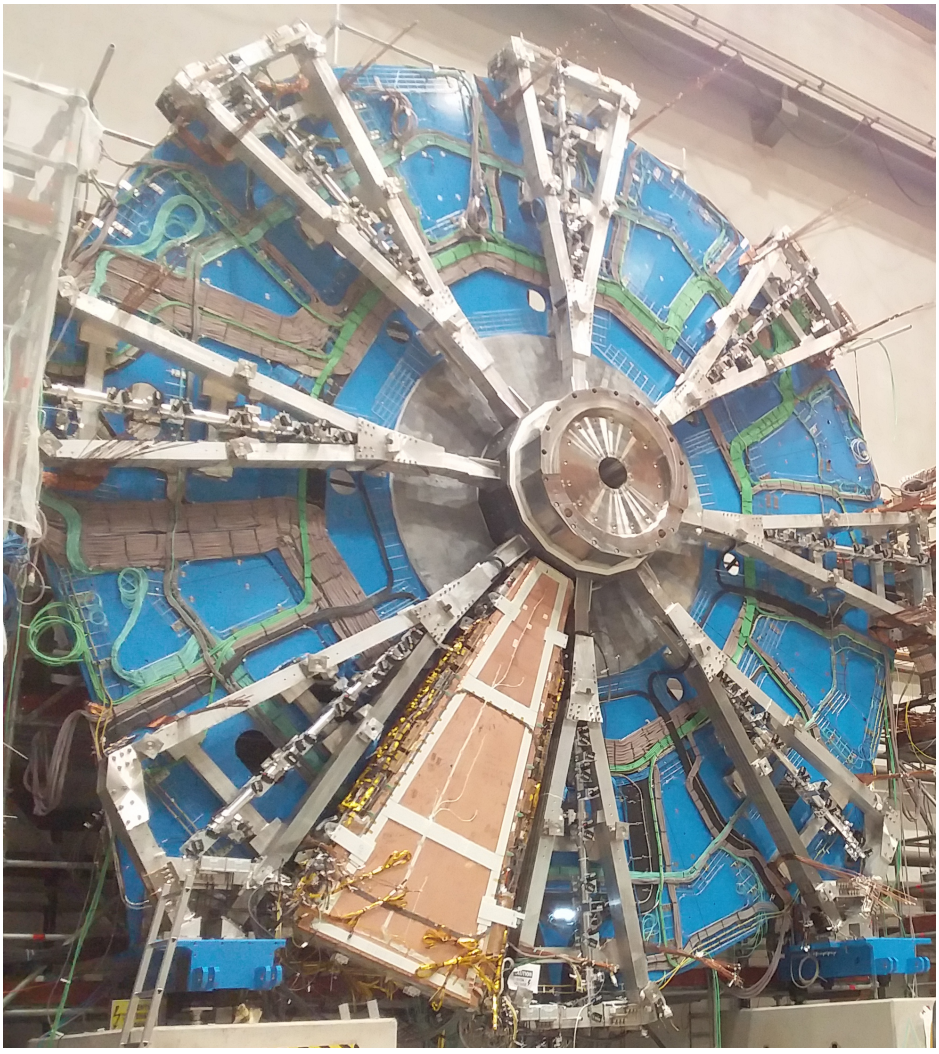


Figure 1.10: A NSW wheel during its commissioning phase in building 191 at the CERN Meyrin site. In this picture, one of the MM/sTGC sectors has been successfully installed and is awaiting the rest of the sectors to be placed in their designated positions prior to lowering the entire wheel into the ATLAS cavern.

Chapter 2

The Micromegas Detector and the VMM ASIC

In this Chapter, two of the main components that will be referred to continuously throughout this dissertation will be presented. As mentioned in Chapter 1, the NSW upgrade employs two detector technologies to provide trigger and tracking information to ATLAS. One of these two, is the *Micromegas Detector*, which is mainly used for precision muon track reconstruction, but also for triggering purposes. It has excellent tracking capabilities (better than $100\,\mu m$ for all particle impact angles in the New Small Wheel), and its very fine readout strip segmentation, alongside its reasonably good timing resolution, can also be exploited to complement the trigger scheme that is based on the sTGC, adding in the robustness and redundancy of the NSW system [7]. The NSW is the first project to make use of this detector technology in this scale, and numerous studies have been performed in the past, in order to validate its performance [3,4,24,25]. The NSW also makes use of a new generation of front-end electronics, in order to acquire the muon-related data and transfer them to the back-end DAQ system of ATLAS. The cornerstone of the NSW electronics scheme, is the VMM ASIC, a mixed-signal front-end chip that reads-out both detector technologies of the NSW [9,26,27]. The chip's main functionalities will be described here, while its relationship with the rest of the NSW electronics scheme will be studied in Chapter 7.

2.1 The Micromegas Detector

The Micromegas detector, an abbreviation for Micro MEsh GAseous Structure (MM), was introduced in the middle of the 1990's [28,29]. It is comprised of two regions. One

is the *drift* region, and the other is the *amplification* region. The *micromesh*[1] separates the two regions, and is usually tied to ground. It is woven, pre-stretched, and made of stainless steel. On the very top of the layer of the chamber, the *drift electrode* is tied to $-300\,\mathrm{V}$. The distance between the drift electrode and the mesh is $\sim 5\,\mathrm{mm}$, and defines the drift region. The *resistive strips* lie below the micromesh, in a distance of just $128\,\mu\mathrm{m}$. A set of cylindrical spacers (pillars), made of insulating material, are placed with a pitch of a few mm on the surface of the resistive plane, supporting the mesh. A voltage of over $500\,\mathrm{V}$ is applied to the strips. The region between the mesh and the resistive strips is the amplification region. Due to the application of the high voltage, an electric field is formed in both regions. The drift region is characterized by an electric field of $E_{drift} = 0.6\,\mathrm{kV/cm}$, while the amplification gap, because of the small distance between the grounded mesh and the under-voltage strips, has an intense electric field of $E_{ampl.} = 40\,\mathrm{kV/cm}$. The detector is filled with argon and carbon dioxide gas at $2\,\mathrm{mbar}$, with the nominal mix ratio between them being $93:7$ (in favor of the argon). Charged particles traversing the detector ionize the gas, and the electrons liberated by the ionization process drift towards the mesh, in tens of nanoseconds. the mesh is transparent to more than 95% of the electrons. Due to the intensity of the electric field in the amplification region, an electron avalanche takes place almost immediately after the readout electrode, within a few nanoseconds [4]. During the avalanche formation procedure described above, a number of electron-ion pairs is created. Electrons and positive ions, separated by the electric field, drift towards opposite directions. Their motion within the gas volume induce charge on the readout elements. Electrons drift fast and within a few nanoseconds reach the grounded mesh. Therefore, the current flowing on the readout strips because of them is on the order of nanoseconds and is usually ignored by the detector electronics. On the other hand, the positive ions drift with a velocity two to three orders of magnitude less than this of electrons, because of their mass. Hence, they induce negative charge on the electrodes with hundreds of nanoseconds duration, that is picked up by the front-end electronics that are attached to the MM readout strips [3]. These electronics are packaged in a mixed-signal ASIC called VMM that is described in Section 2.2. The MM chamber alongside a depiction of an ionization process is depicted in Figure 2.1.

The *Resistive Micromegas* design as it is called, did not occur without a significant research and development. To be precise, on 2007 a group called Muon ATLAS MicroMegas Activity (MAMMA) was formed to conduct research on the potential use of the original detector design for the ATLAS muon spectrometer. The first Micromegas prototypes that were produced did not follow the resistive strip concept described above. They were of the so-called *Bulk Micromegas* design, where the

[1] Also simply referred to as *mesh*.

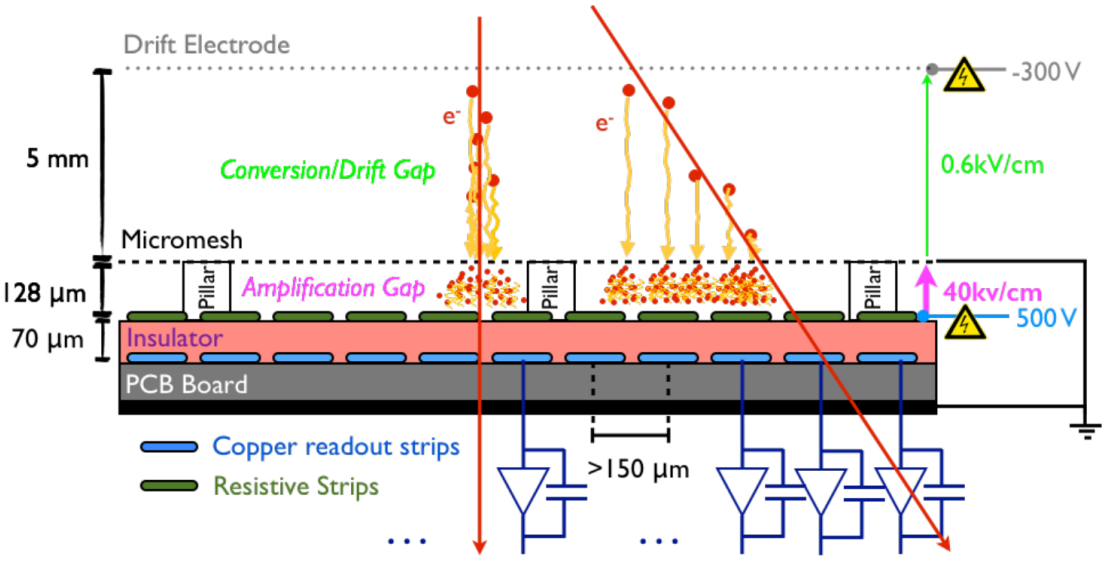


Figure 2.1: Graphical representation of a layer of the resistive Micromegas detector, depicting its various regions and elements, and the ionization process when two hypothetical charged particles traverse the chamber's active area [4].

resistive layer did not even exist. The MAMMA research project proved that these prototypes were hindered by sparks. Sparks are a major issue, as they lead to breakdowns of the supplied voltage with relatively long recovery times and thus inefficiencies [3]. Sparks may also damage the detector elements and the readout electronics, resulting in dead areas on the medium [4]. Moreover, a dangerously high particle flux is expected to be present in the ATLAS Muon system during the LHC Run-2 and beyond. To be precise, particle rates of the order of $15\,\mathrm{kHz/cm^2}$ are expected, with more than 70% coming from hadron (protons and neutrons) and photon interactions that lead to large energy depositions. This harsh environment increases the risk of sparking for the MM operated at high gas multiplication values. Hence the design of the resistive strips, which have a thickness of about $\sim 45\,\mu\mathrm{m}$, a resistivity of a few tens of $\mathrm{M\Omega/cm}$, and are placed on a $\sim 60\,\mu\mathrm{m}$-thick layer of insulator. These strips are arranged the same way as the readout strips below them [30, 31], and provide enough protection to the apparatus from the dangerous sparks.

2.1.1 The Micromegas Modules for the New Small Wheel Upgrade

Since the overall design and concept of operation of the Micromegas detector has been established, in this Subsection, some more details on the modules that will be deployed in the ATLAS cavern as part of the NSW upgrade will be given. The MM sectors, or

wedges, are divided into two large categories: the Small Module (SM) and the Large Module (LM). Each one of the two size families, is divided into two parts: SM1, SM2, and LM1, LM2. Each wedge is comprised of a double quadruplet, so eight discrete layers, like the one depicted in Figure 2.1. Sixteen SM and sixteen LM modules will be produced in total, with one of each wheels employing eight LM and eight SM wedges. Each sector is comprised of eight readout Printed Circuit Boards (PCBs), and every PCB has 1024 resistive/readout strips. The strip width is $300 \pm 20\,\mu\text{m}$ for all the modules, with a pitch of $425 \pm 20\,\mu\text{m}$ and $450 \pm 20\,\mu\text{m}$ for small and large modules respectively. The length of the strips vary from $0.5\,\text{m}$, up to $2.2\,\text{m}$. There are three different types of readout PCBs in each MM module. Two have their strips running perpendicularly to the radial direction, referred to as *precision strips* or *eta* (η), while the other two have their strips tilted by a small angle ($+1.5°$ and $-1.5°$), referred to as *stereo angle/strips*, to allow for the reconstruction of the second coordinate (ϕ) [4]. A diagram depicting all the aforementioned parameters, is displayed in Figure 2.2.

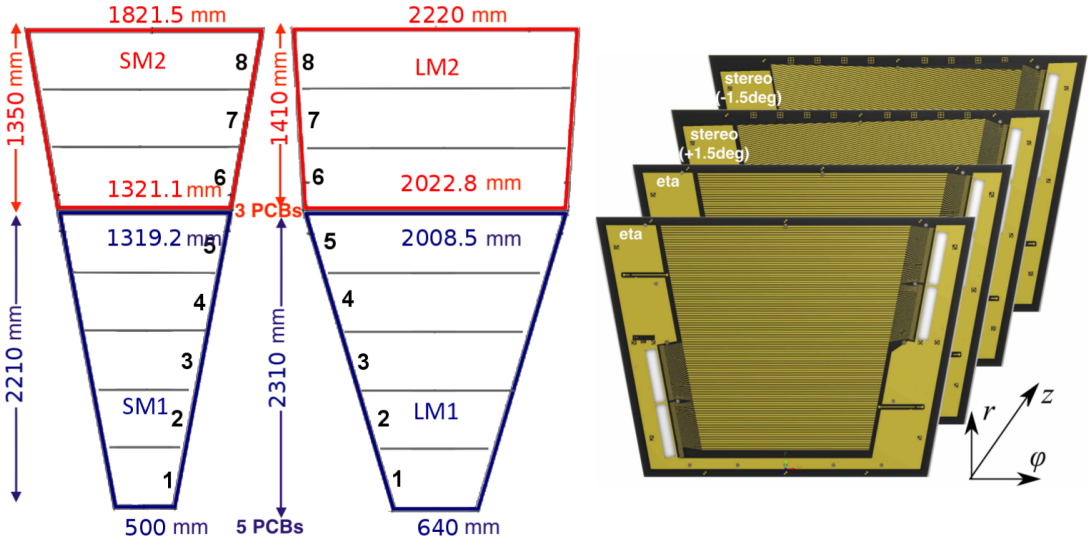


Figure 2.2: *Left:* Graphical representation of a the LM and SM wedges, with their eight readout PCBs. *Right:* Four successive readout PCBs of one MM quadruplet. The eta and stereo (angled) strip arrangement is shown [3, 32].

2.2 The VMM Application-Specific Integrated Circuit

The VMM is an Application-Specific Integrated Circuit (ASIC), intended to be used as the front-end readout chip of both the Micromegas and sTGC detectors of the NSW Phase-I upgrade project of the ATLAS experiment at CERN [7]. It has also been proposed for several other applications and smaller-scale experiments. Developed

at Brookhaven National Laboratory, the VMM is fabricated in the $130\,\text{nm}$ Global Foundries 8RF-DM process (formerly IBM 8RF-DM), and is composed of about ten million transistors. The device is packaged in a Ball Grid Array (BGA) with a footprint of $21 \times 21\,\text{mm}^2$. Radiation-wise, the VMM is expected to be exposed to a total ionization dose of $100\,\text{krad}$, but its Triple Modular Redundancy (TMR) architecture allows for it to operate smoothly, even after that dose. The chip has a plethora of configuration parameters that the users can tune according to their needs. This is done via the Serial Peripheral Interface (SPI) protocol, which makes it possible to shift 1728 bits into the ASIC's configuration registers.

The VMM is composed of 64 identical and independent input channels, each one of which connects to a readout element of the detector medium. For the Micromegas, this is the the chamber's resistive strip, while for the sTGC, the channel accepts input pulses from the wires, strips and the pads of the chamber. A block diagram of one of these channels is shown in Figure 2.3. Each channel integrates a low-noise charge amplifier (CA) with adaptive feedback, test capacitor, and adjustable polarity (in order to process either positively or negatively charged input signals). The VMM's input MOSFET is a p-channel with gate area of $L \times W = 180\,\text{nm} \times 10\,\text{mm}$ (200 fingers, $50\,\mu\text{m}$ each) biased at a drain current $ID = 2\,\text{mA}$; this corresponds to an inversion coefficient $IC \approx 0.22$, a transconductance $g_m \approx 50\,\text{mS}$, and a gate capacitance $C_g \approx 11\,\text{pF}$. A block diagram of the charge amplifier can be found in Figure 2.5. Finally, given the requirements of the NSW, the noise requirements of the analog circuitry are deemed to be within specifications (see Figure 2.4).

The integrated charge gets shaped by a third-order filter (shaper) with one real pole and two complex-conjugate poles, designed in Delayed-Dissipative Feedback (DDF) [26]. It has adjustable peaking time in four values (25, 50, 100 and $200\,\text{ns}$) and stabilized, band-gap referenced, baseline. The gain is adjustable in eight values (0.5, 1, 3, 4.5, 6, 9, 12 and $16\,\text{mV/fC}$). Next to the shapers are the sub-hysteresis discriminators [9] with neighbor-enabling logic and individual threshold trimming, the peak detector, and the time detector. The threshold is adjusted by a global 10-bit Digital-to-Analog Converter (DAC) and an individual-channel 5-bit trimming DAC. The neighbor-channel logic forces the neighboring channels to a triggered one to perform measurements, even if those channels did not exceed the set threshold. The neighbor logic extends also to the two neighboring chips through bidirectional I/Os. This scheme allows for setting the threshold level high enough to reject noise, without the fear of losing useful information, since most of the time a particle deposits a large amount of energy (i.e. pulse charge) to one or two detector strips, and much less to neighboring ones, thus producing pulses that do not cross the user-defined threshold. Therefore, the neighbor logic allows to retain all necessary information that aids to

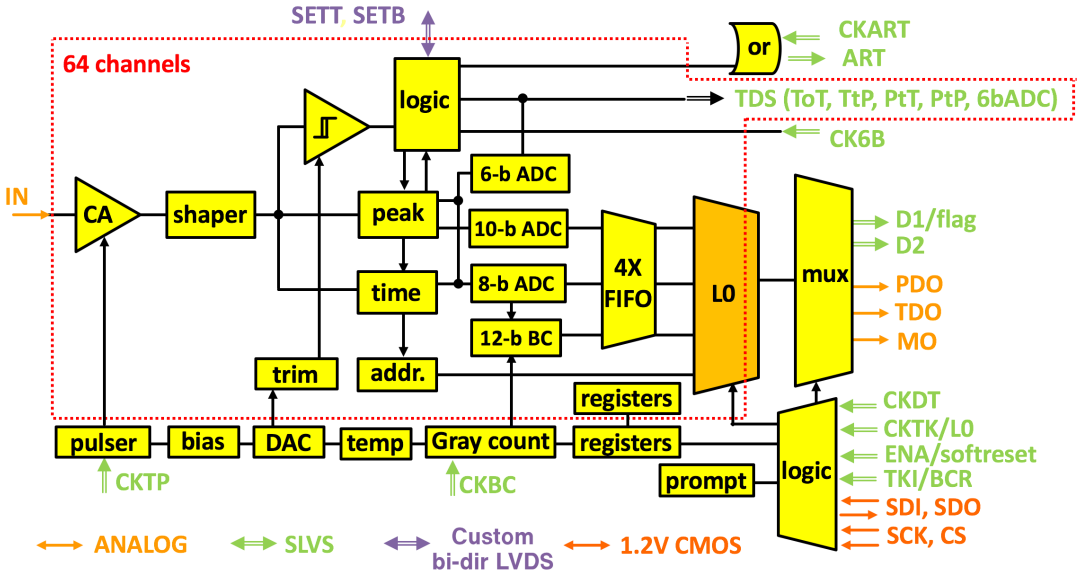


Figure 2.3: Architecture of the VMM. The majority of the block diagram describes one of the identical 64 channels of the chip. The Charge Amplifier (CA) and shaper process the input pulse, while the ADCs digitize the pulse height and timing information, before storing their results in the channel's FIFO. The L0 logic selects data based on the trigger input (L0) and timestamp information of the event. The data are sent to the external readout device via two lines, D1 and D2, synchronized with a clock provided to the CKDT input. Three outputs, useful for monitoring, drive the peak detector (PDO), time detector (TDO) and channel monitor (MO) voltages out of the chip. The test-pulse strobe (CKTP) input drives the channel's testing circuitry, while the reference clock is provided via the CKBC pin. The chip can be reset by issuing a pulse in the ENA or TKI pins. The configuration of the ASIC is conducted via the SDI, SDO, SCK and CS pins. Finally, the trigger primitives are sent out of the chip via the ART, ToT, TtP, PtT, PtP and 6bADC outputs, synchronized with the CKART and CK6B clock inputs.

track reconstruction [7].

If an input pulse crosses the user-defined threshold, or if a neighbor logic signal is received, the pulse is processed further as follows: The shaped pulse's peak amplitude is measured by the peak detector circuitry, which produces a voltage level proportional to the pulse's amplitude. In principle, the pulse's amplitude is proportional to the energy deposited to the detector medium by the ionizing particle, or proportional to the test-pulse amplitude DAC value, if the VMM is being test-pulsed. The pulse's timing is measured by the time detector, by employing a Time-to-Amplitude Converter (TAC); a voltage ramp that starts either at threshold crossing, or at the time of the peak and stops at the next falling edge of the reference clock. For the ATLAS implementation,

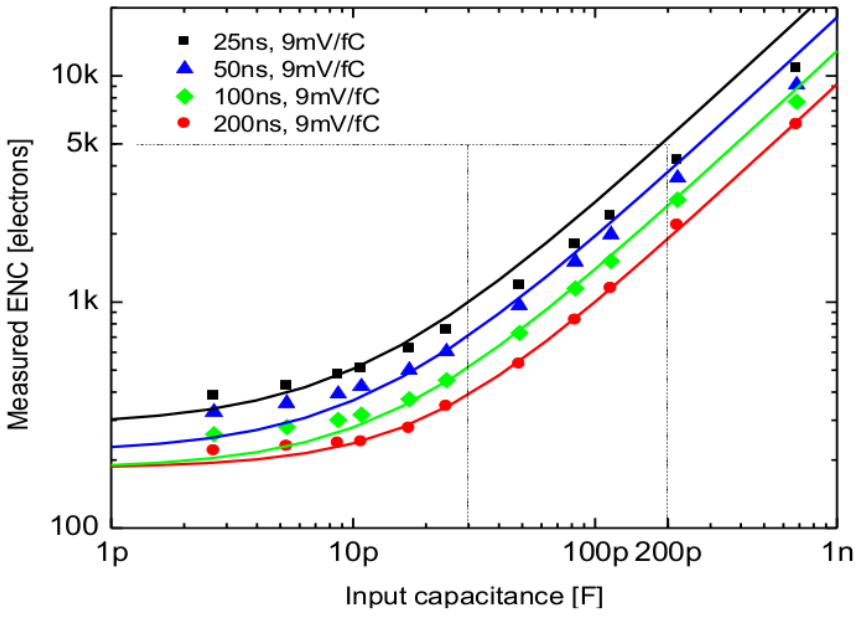


Figure 2.4: The VMM input channel noise as a function of all four peaking times at a gain of 9 mV/fC [11]. The solid lines represent the simulation results, while the markers are extracted from experimental data.

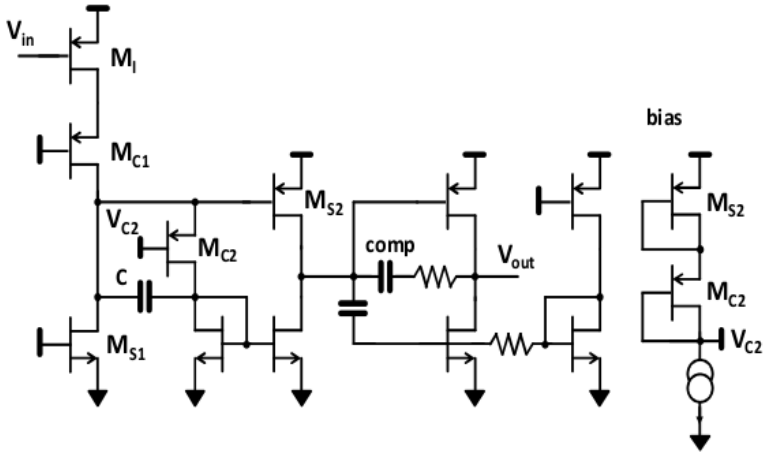


Figure 2.5: Block diagram of the VMM's charge amplifier at the transistor level [9].

this is the Bunch-Crossing (BC) clock[2] with a frequency of 40.079 MHz. The ramp duration is adjustable in four values (60, 100, 350 and 650 ns). The reader may refer to Figure 2.6 for a depiction of the TAC's operation. The resulting voltage from the TAC and the peak detector, are stored in associated analog buffers.

Three different Analog to Digital Converters (ADC) receive the results from the analog

[2]The rising edges of the BC clock coincide with the proton collisions in the core of ATLAS.

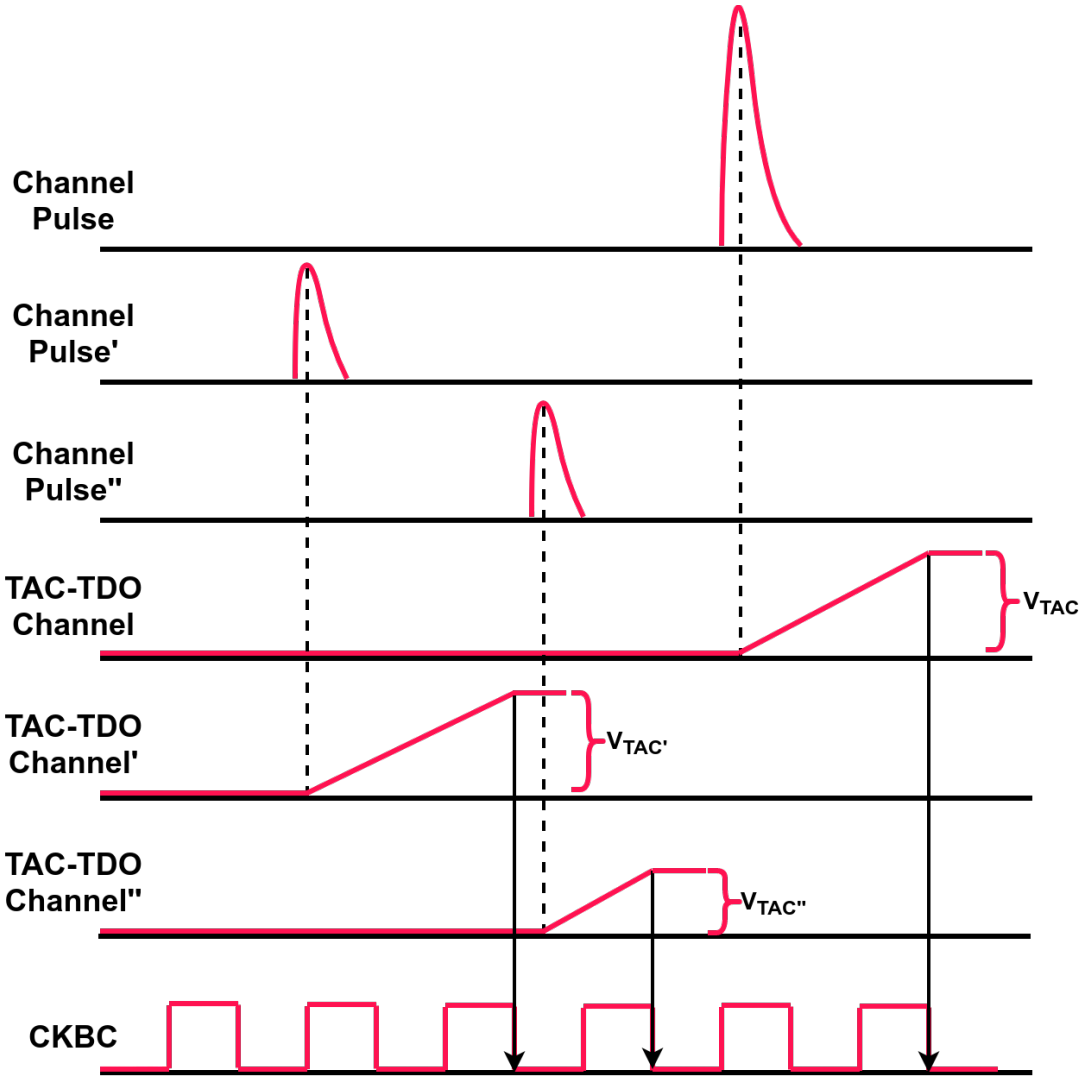


Figure 2.6: Depiction of an event in one VMM that stimulates three channels. Each input pulse gets shaped by the corresponding circuitry, and upon the assertion of the "peak found" signal, the Time-to-Amplitude Converter (TAC) starts ramping up. The next falling edge of the reference clock (CKBC), stops the TAC ramp. Depending on the duration of the ramp, a voltage level is created. The phase relationship between the pulse's peak and the reference clock therefore determines the amplitude of the said level. This voltage then is driven to the 8-bit TDO ADC, which digitizes it, and stores it into the VMM's readout buffers.

circuitry, stored in the buffers mentioned above, and digitize them. The measurement of the timing of the pulse is being converted by an $8-$bit ADC, that encodes the voltage level produced by the TAC into eight bits, resulting in a maximum fine timing accuracy of < 1 ns. This byte, or word, is called Time Detector Output (TDO). For the pulse's

amplitude digitization, a $10-$bit ADC is employed. This module receives the voltage level produced by the peak detector and encodes it into ten bits, resulting in a precise charge estimation with < 1 fC accuracy. This ten-bit word, is called Peak Detector Output (PDO). After the digitization results are ready, they are forwarded to a 4-event-deep FIFO at the next rising edge of the VMM's reference clock (CKBC). The entire procedure takes up about 250 ns to complete, which sets the upper limit to the maximum rate that a channel of the ASIC can handle, which is 4 MHz. However, if the user wishes so, the $10-$bit ADC may not be used, and a less precise $6-$bit ADC may be employed to perform the amplitude measurement instead. This results in a less precise amplitude measurement, but with a significantly reduced dead-time. Alongside the precise timing measurement, the VMM also performs a coarse timing measurement by timestamping the event with the so-called Bunch-Crossing Identification (BCID) counter value. As mentioned earlier, the VMM's reference clock is the Bunch-Crossing clock, or CKBC, operating at a frequency of 40.079 MHz. At each rising edge of the said clock, an internal 12-bit Gray Code counter increments by one. If a channel gets activated by a pulse that crossed the threshold, or by a neighbor logic signal, the value of the counter gets ”attached” to the event, thus constituting a coarse timing measurement, that in the final experiment correlates an event with a particular bunch-crossing that occurred in ATLAS. Hence, an event has its timing information encoded in 20 bits in total, one of which is the TDO (fine timing) and the rest is the value of the VMM's BCID counter (coarse timing).

The 20 bits of the timing information, plus the 10 bits of the PDO (charge information), and the 6 bits of the channel ID, alongside two flags, are stored in a FIFO with four positions that is 38 bits in width. This information is either forwarded to the Level-0 buffers, or retained in the shallow FIFO, depending on the chosen readout mode of the VMM.

2.2.1 Trigger and Readout Paths

The VMM, being an all-in-one front-end ASIC solution for the needs of the NSW, delivers both readout/tracking information for ATLAS, while it also produces trigger primitives. In principle, the VMM supports two distinct paths: the first is the *Readout* path, and the other one is the *Trigger* path. The first path has already been described above, up to the point of the buffering of the readout (or tracking) data. In parallel with these functionalities, the VMM also produces fast signals, that can be used by the ATLAS detector to make a trigger decision, and feed that trigger to the front-end electronics that in the meantime have been buffering data. In essence, the VMM continuously stores data into its buffers while it also produces trigger signals that are

immediately outputted, and get processed by the trigger electronics of ATLAS. The VMM is agnostic of the origin of the data that it processes. They can be related to electronic noise, to particles that are product of parasitic reactions occurring in the detector's materials, or, ideally, to particles originating from the interaction point. Only the trigger electronics (that have an overview of the entire system), decide which particular events resemble tracks originating from the interaction point and which don't. If a track is found, these modules fan-out a trigger signal, called *Level-0* or simply *L0* signal to the electronics. This signal should have a *well-defined latency* with respect to the event that is associated with. If this is true, then the front-end devices (VMM included) that receive this trigger signal "know" in which part of their buffers to look for the trigger-related data, that they then stream out to an external readout device. These data are eventually stored and are used for off-line data analysis. The whole scheme is roughly described in Figure 2.7.

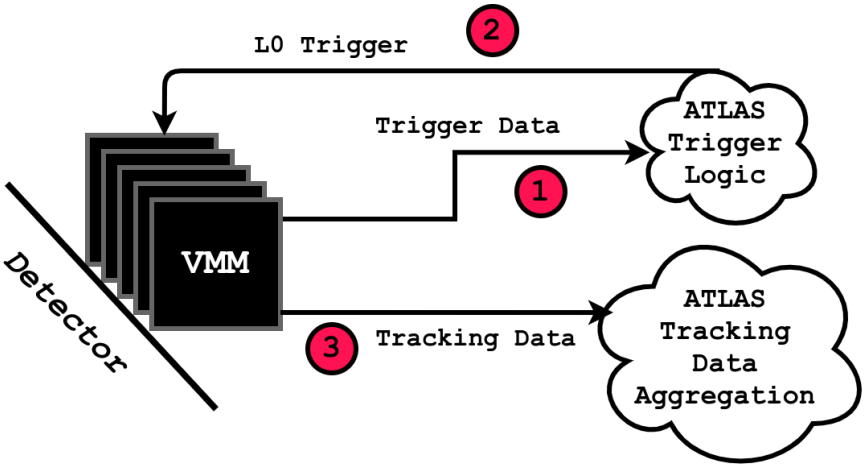


Figure 2.7: The VMM's data paths and its relationship with the ATLAS detector Trigger and Data Acquisition infrastructure. The ASIC first *(1)* outputs fast trigger signals associated with a particle that passed through the detector medium. Meanwhile the same event gets buffered in the VMM's memories. The ATLAS trigger electronics use the fast trigger outputs of the VMM, alongside other trigger sources, to make a decision. If a valid track is found, the electronics then issue the L0 trigger *(2)* , and the VMM uses that signal to select the related data from its buffers and forward them *(3)* to a tracking/readout data aggregation agent.

For the readout/tracking path, The VMM has three modes of operation: a Two-Phase Analog Mode, a Continuous read/write Mode, and the Level-0 Mode. All of these three modes will be briefly described below, and will be revisited in the actual readout implementation of Chapter 5.

Readout Path - Two-Phase Analog Mode

In the two-phase mode, data are registered while the VMM is in acquisition mode and then are read-out once the system is switched to the readout mode. Acquisition is re-enabled after the data extraction phase is completed. The readout advances to the next channel by injecting a token to the relevant input. The token is sparse, passed only among those channels with valid events. Once the procedure is complete, the token is routed to the output for reading-out the next chip, thus allowing for a daisy-chained readout with a single token input. In this mode, the internal ADCs are switched off and the analog signals are read-out through analog buffers with external ADCs.

Readout Path - Continuous Mode

In continuous mode, the simultaneous read/write of data assures dead-timeless operation that can handle rates up to the maximum of 4 MHz per channel[3]. Higher rates can be achieved by interrupting the 10-bit ADC once the 6-bit ADC has finished, since the 6-bit ADC conversion takes up $\sim$ 40 ns including the channel reset. The peak and time detectors convert the voltages into currents that are routed into the $10-$ and $8-$bit ADC, respectively. The channel is reset once both conversions are completed and the digital values are latched in digital memories. The self-reset of each channel provides continuous and independent operation of all 64 channels. In this mode, each channel of the ASIC stores the $38-$bit event-related data into a four-positions deep de-randomizing FIFO. The data are read-out via two serial data lines (*D1* and *D2* in Figure 2.3).

Readout Path - Level-0 Mode

The Level-0 (L0) readout mode of the VMM was designed to be compatible with the readout scheme of the ATLAS experiment. It provides the ability to buffer the channel hit data in a deeper memory architecture and allows the VMM to select the data to be read-out after the reception of an external trigger input, namely the L0 signal. The L0 selection logic applies a configurable latency to the trigger and forwards out only hit data for which their timestamp is within a window of the trigger's time-of-arrival plus its latency offset. A block diagram of the logic is depicted in Figure 2.8, where data from the VMM channel(s) are forwarded from left to right through various buffers. The first FIFO is a 64-deep latency buffer (LAT. FIFO in Figure 2.8) where all hits can be buffered. Each channel has a selector circuit based on the arrival of the external trigger signal which finds events within the configurable window[4] (*L0*

[3] This is a result of the 250 ns conversion time of the internal ADCs.

[4] Maximum size of eight reference clock cycles, which is equal to 200 ns.

SEL in Figure 2.8). Once there is a match, the data are copied to the channel FIFO (*L0 CH FIFO* in Figure 2.8) and are ready to be read-out in a round-robin manner, starting with the information of the BC timestamp. The data are forwarded out of the ASIC via two serial lines, synchronous to both edges of an externally-provided (via the *CKDT* pin) 160 MHz data clock, resulting in a total bandwidth of 640 Mb/s. The data are encoded using the 8b10b protocol [33], leading to an effective user bandwidth of 512 Mb/s.

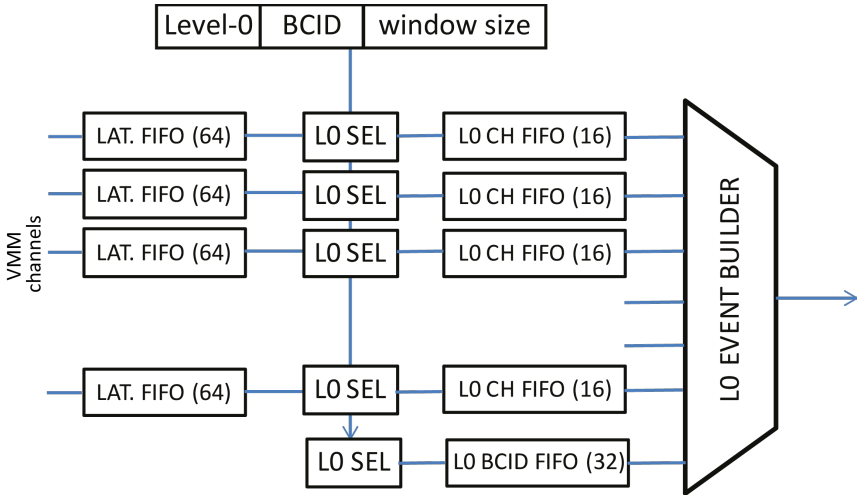


Figure 2.8: L0 buffer and event selection logic block diagram. The trigger signal, or L0, gets timestamped with a Bunch-Crossing ID (BCID) value, then a configurable offset is applied to it. The hits stored in each 64-event-deep latency FIFO (LAT. FIFO), have a timestamp themselves as well. The L0 selection logic (L0 SEL), check the hit timestamps against the ones of the trigger. Hits within a window of configurable width, enter the L0 CH FIFO, before being sent out of the ASIC by the L0 Event Builder.

The VMM's Trigger Path

For the MM trigger scheme, the Address-In-Real Time (ART) trigger output mode is used. Upon the detection of a pulse peak, or upon a signal crossing of the user-defined threshold, the chip streams out the *six-bit channel address*[5] that this peak was detected on *first*. Depending on the chosen configuration parameters on the VMM, the total latency of the streamed ART address can be within ~ 15 ns, or ~ 20 ns plus the peaking time. This latency is the sum of several delays, some of which are the 5 ns delay between the peak and the peak-found signal, the 5 ns digital latency from the comparator firing to the leading edge of the ART, and the 5 ns digital latency from the peak-found signal to the leading edge of the ART. For the NSW implementation, the

[5] Accompanied by a flag.

$6-$bit channel address plus a flag are sent to an external device that aggregates trigger primitives at a rate of $320\,\mathrm{Mb/s}$.

For the sTGC detector technology, the VMM provides trigger information via its 64 *direct timing* outputs, one for each of its channels. Each channel implements a fast digitization 6-bit ADC, that provides a coarse, but rapid (the conversion takes up about $40\,\mathrm{ns}$, including the channel reset), amplitude measurement of the pulse. This is the same ADC that was mentioned earlier in this section, when describing the ADCs of the VMM's channels. The conversion starts immediately at the peak time and the data are streamed at each output. Another way to operate the direct timing outputs is in the Time-over-Threshold (ToT) mode. As long as a pulse is above the threshold, the VMM outputs a digital ToT pulse, in one of its direct-timing output pins with a latency of about $14\,\mathrm{ns}$. There are two types of sTGC trigger-data-aggregating devices, each dedicated to handling one of the two types of trigger datastreams.

Conclusions

In this Chapter, the Micromegas detector was introduced in more detail. It is a type of micropattern gaseous detector, a well-established technology [1]. The Micromegas detector, innovative in its design, was introduced in 1997. It is able to perform precise charge and timing measurements to incident radiation, while it is also resilient to high particle flux. These characteristics make it ideal for deployment in the NSW upgrade. The MM's readout strips will be instrumented by the VMM ASIC. It is a mixed-signal custom chip, with a long R&D history beginning from 2012. The VMM is fully capable of supporting all the needs of the MM detector, processing the muon-induced pulses fast and accurately enough to provide precision tracking information with minimal dead-time, and also output muon trigger primitives, to-be-used by the NSW trigger logic, within a few nanoseconds. The ASIC is not only capable of reading-out the MM, but has also been designed to provide trigger and tracking data originating from the sTGC chambers. The MM detector will be revisited in Chapter 6, where it will be deployed and read-out in testbeam scenarios, and in Chapter 7, where its integration with the ATLAS DAQ system will be studied. The VMM on the other hand, is a prominent figure in the second part of this dissertation, where a readout system that has been developed around it will be described in Chapters 4, 5 and 6, before finalizing with mentioning the important role of the VMM in the NSW integration and commissioning part, in Chapter 7.

For more information, the reader is also referred to a VMM-related proceeding that was published by the writer on behalf of the ATLAS collaboration [10].

Chapter 3

Field-Programmable Gate Arrays

Field-Programmable Gate Arrays, or simply FPGAs, is a family of integrated circuits that can implement any set of logical functions, depending on the needs of its user (also called *designer*, or *firmware engineer*), that programs it accordingly. In any FPGA-based application, whichever this might be, the end result will be a digital circuit, implemented inside the FPGA device. The main advantage of these chips is their flexibility, since they can be re-programmed/re-designed by their user over a practically infinite amount of iterations. This dynamic reconfigurability feature, alongside its high-performance characteristics, which allow it to process data with minimal (and more importantly, *deterministic*) latency, compared to off-the-shelf personal computer processors and embedded microprocessors[1], deem it ideal for demanding applications. FPGAs may be used as processing units in satellites, aircraft, oscilloscopes, servers, routers, PET scanners, high-end monitors and many more. As it will become evident in the second part of this dissertation, another important application of these devices is the ATLAS experiment at CERN. Since the FPGAs used in ATLAS are produced by Xilinx®, the current Chapter will be devoted into describing chips of the said vendor. However, the differences between FPGA architectures are not that substantial, when comparing various FPGA-fabricating companies.

[1] These devices are burdened by the overhead of a variety of other processes that must be executed by their logic units. Therefore, the time it takes to execute a desired operation, cannot be estimated at the nanosecond level, which is desired in many applications, including these of high energy physics. On top of that, if designed optimally, FPGAs take full advantage of parallelism features, which greatly reduces processing time.

3.1 Fundamental Architectural Elements

The first FPGA was fabricated by Xilinx® in 1985 [34]. The general philosophy of implementing logical functions in the FPGA die is based on utilizing two types resources, implemented as of 2020 in 16 nm technology [13]. These types are: Configurable Logic Blocks (CLBs) and Configurable Interconnect [13, 35–37]. The CLBs are the components performing most of the logical computations inside the device. The signals that the logic blocks handle, are mediated throughout the FPGA via a complex routing network, which is responsible for connecting the CLBs with each other, or with the chip's Input/Outputs (I/Os). This is the configurable interconnect. As their name suggests, both groups of resources are fully programmable. Depending on the syntax of the Hardware Description Language (HDL)[2] developed by its user, the FPGA will activate different CLBs and interconnect in various ways, in order to implement the functions indicated by the source code. This complex procedure is being performed by each FPGA vendor's Electronic Design Automation (EDA) tool. The designer inputs the firmware into the EDA tool, which translates the human-readable code into CLB configurations and in-between connections. The tool then creates the so-called *bitstream*, which downloads into the FPGA. The device uses the received dataset to activate the logic blocks and routing that have been designated by the electronic design software through the bitstream. The chip is then operating under the user's firmware design. This procedure may be repeated as many times needed.

In the Sections that follow, a more detailed description of the logic blocks and the configurable interconnect will be given, alongside a deeper insight on how the user's source code infers a logic circuit inside the FPGA.

3.1.1 Look-Up Tables, Flip-Flops and Configurable Logic Blocks

Configurable Logic Blocks, or CLBs, are deployed throughout the FPGA die, with their main components being two types of circuit: Look-Up Tables (LUTs), and Registers (usually in the form of flip-flop gates), that will now be described in more detail.

Combinatorial computations inside the FPGA are not actually being performed by conventional logic gates[3], but via Look-Up Tables (LUTs), which are one of the basic elements of the CLBs. In essence, a LUT is a Read-Only Memory (ROM), that is loaded with an initialization value upon the FPGA's configuration by the user. For example, if a part of the designer's source code is translated into a three-input `OR` gate, the EDA tool

[2] The two most common languages are: Very High Speed Integrated Circuit Hardware Description Language (VHDL), and Verilog.

[3] Meaning e.g. `OR`, `AND`, `XOR` gates - the reader is referred to the extensive literature for more information on their operating principles [34, 38, 39].

will choose which LUT of the FPGA's fabric will implement it, and load its memory with the truth table of a three-input boolean `OR` function. The most common type of LUT implemented by the Xilinx® 7-Series FPGAs that have been utilized extensively for the purposes of this dissertation, has six inputs and one output. The five of the six inputs are driven into two five-input LUTs, that are loaded with the initialization memory vector. The sixth input is used to select between the single-bit output from the two sub-components, extending the LUT's functionality. The architecture of the block is shown in Figure 3.1.

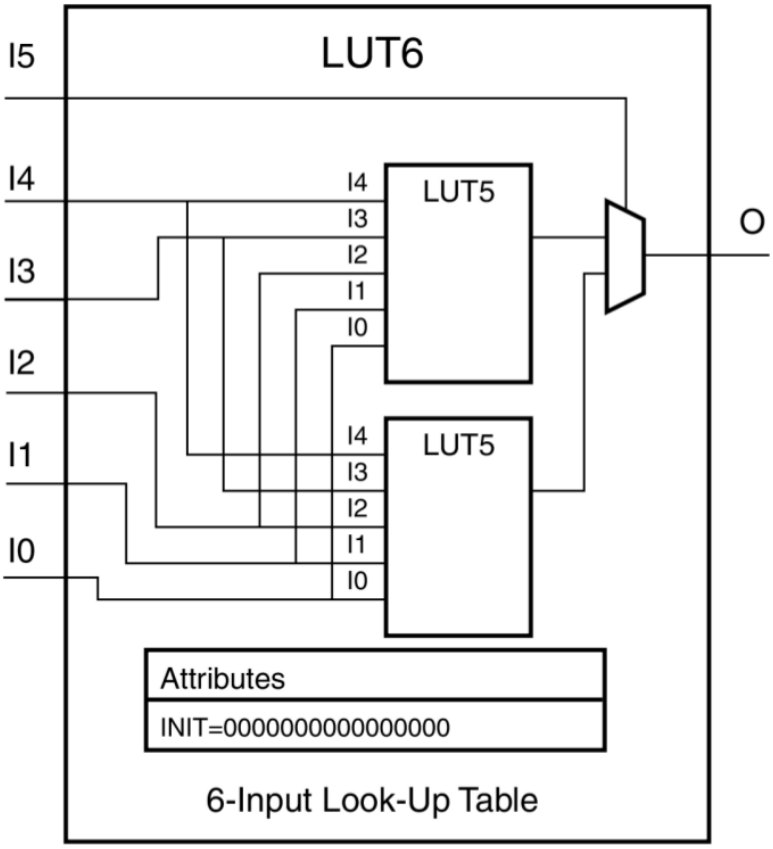


Figure 3.1: Xilinx® 7-Series FPGA six-input Look-Up Table (`LUT6`). The inputs are driven into two sub-LUTs with five inputs, that forward their output into a multiplexer, which in turn drives the LUT6's output.

A large FPGA design however, does not only make use of a single (or more) LUT(s) to implement all of its combinatorial functions. In digital design, logic elements are either *combinatorial*, or *sequential*. For combinatorial circuits (e.g. LUTs), the state of the output only depends on the state of the input signals. In sequential logic circuits though, the state of the output also depends on the previous states that the system was in. Therefore, designs of sequential logic employ memory elements, or *Registers*. One

of the most common types of register, is the *Flip-Flop Gate*[4], which alongside the LUT, is one of the major components of the FPGA's CLBs. In order for a flip-flop gate to operate, the existence of a *clock* signal is essential. The flip-flop's output may change state only at the moment the clock changes state (usually from low to high - which is a positive-edge flip-flop). If for example a register uses a clock of $f = 40\,\text{MHz}$ as a clock input (usually denoted as `C` or `CK`), then its output may be changing every $25\,\text{ns}$. At the moment the clock changes state, the flip-flop forwards the value of its input (usually denoted as `D`) to its output (usually denoted as `Q`). If any change in the logical input signal occurs in-between the edges of the clock, then it will never be propagated to the output (i.e. the flip-flop samples its input only when its clock changes state). The functionality of the flip-flop is depicted in Figure 3.2.

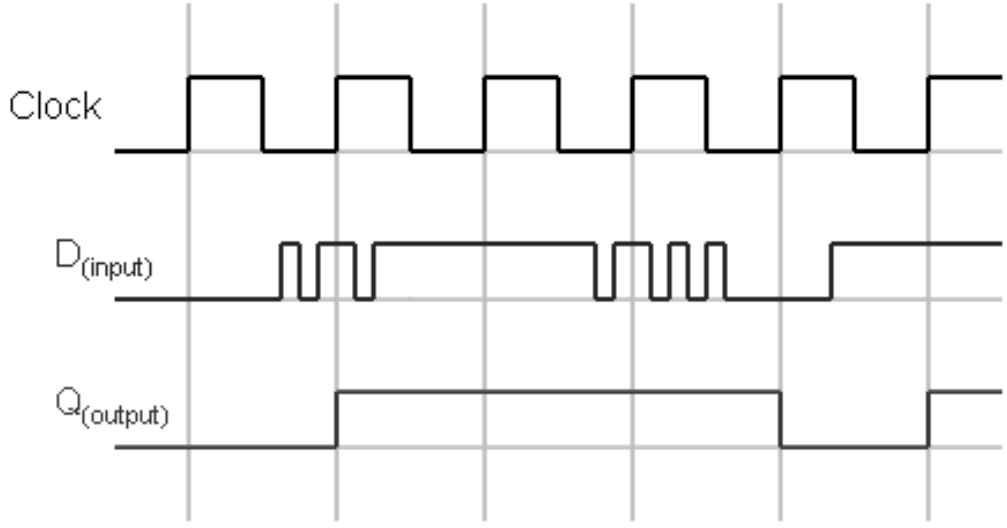


Figure 3.2: Timing diagram of a positive-edge DFF, which is a common primitive of the Xilinx® 7-Series FPGA family. Note that the output of the gate changes only when the clock transitions from low to high.

FPGAs utilize flip-flop gates in their CLBs in order to store the LUT outputs. FPGAs combine these two basic elements, alongside multiplexers which are also present in the CLBs, to implement most logical functions [13, 35, 36] as designated by the user's HDL input. For the Xilinx® 7-Series FPGAs, CLBs are separated in two regions, called *slices*. Each slice implements four LUT6s (see Figure 3.1) with two outputs instead of one, one originating from each sub-LUT with five inputs. Four DFFs (see Figure 3.2) are also present, each one of which may be connected to one of the LUT6

[4]The Xilinx® 7-Series FPGAs utilize D Flip-Flops (or *DFFs*) in abundance, and throughout this text the functionality of this type of primitive is described.

outputs. Finally, the basic elements of each slice is complete by 2-to-1 multiplexers. Additionally, an adder logic is deployed in each slice. It is a dedicated primitive (called Arithmetic Logic Unit (ALU)) that performs summations, and can provide its output to four dedicated flip-flops, which may feed the product back to the summation logic's input, thus creating a clocked adder, a counter, or a comparator [13]. This block also features two pins that can drive (receive) a carry bit to (from) a neighboring slice/CLB via the `CIN/COUT` pins, thus cascading many adding primitives. This feature is called *fast carry chain* and greatly improves the device's abilities to perform simple arithmetic operations over wide buses. The CLB's architecture is depicted in Figure 3.6 at the end of this Chapter.

Slices are flexible blocks that may not only be configured to perform a wide series of logical operations, but they can also be utilized as a piece of Random Access Memory (RAM). Also called *Distributed RAM*, this type of configuration allows the slice to use its clock input to execute synchronous data writing (i.e. storing) and reading, and can transform each of the two slices of a CLB into a $256-$bit RAM [13]. In addition, each slice can be programmed to delay an input signal by a maximum amount of 32 clock cycles, thus creating a $32-$bit *Shift Register* from cascaded DFFs, connected together via the slice's multiplexers and LUTs to create the shift register logic.

3.1.2 Programmable Interconnect

Apart from the generic blocks that perform logical operations (i.e. CLBs and their slices), FPGAs also utilize an abundance of other primitives that are dedicated in more specific tasks. These will be covered later on in this Chapter. What will be described now is the dynamic network of interconnections, that allow all aforementioned elements to communicate with each other, thus creating a high-performance array of logic that can process data with minimal latencies.

For a start, it is worth mentioning that the FPGA's area is mostly covered by its configurable network of interconnections, rather than the logic blocks themselves. To be precise, $80-90\%$ of the FPGA's die is covered by routing, whereas only $10-20\%$ is reserved by generic CLBs or dedicated blocks [37]. Today, most FPGAs follow the so-called *Island-Style Architecture* [35], where the FPGA's fabric is mostly covered by a network or vertical and horizontal routing, which carry the signals all around the device. These interconnections are multiplexed in specific nodes, which are programmed upon the FPGA's configuration, in order to connect the different routing segments together and relay a signal from its source to its final destination. These programmable nodes are the Connection Blocks (CB) and the Switch Boxes (SB)[5]. CBs are strongly affiliated

[5] An SB is also called a *Switch Matrix* for the Xilinx® FPGAs.

with CLBs. All of the CLB's inputs and outputs are multiplexed inside its dedicated CB, which in turn routes its outputs, and receives its inputs to the nearest central node, which utilizes a SB. Each SB is connected to several CBs and inputs/outputs from dedicated hard blocks (described in Subsection 3.1.4). Finally, it is worth mentioning that apart from the routing that connects CBs and SBs together, modern FPGAs also utilize the so-called *Long Lines*, that are dedicated buses managed by the SBs, which can drive a signal fast from one side of the device's fabric to another, without having to pass from many SB nodes, thus easing timing closure for signals that have to be propagated from one side of the chip to another. The general interconnection scheme (excluding long lines and dedicated blocks), is depicted in Figure 3.3.

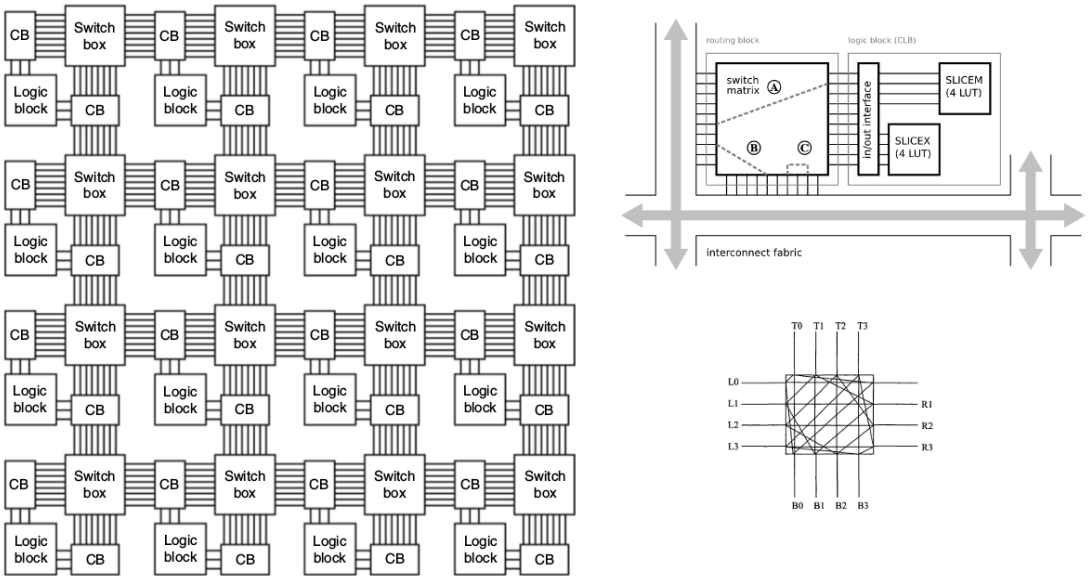


Figure 3.3: *Left:* Graphical representation of the island-style architecture which FPGAs utilize. Every CLB is connected with a CB, which in turn is interfacing with the global interconnect via two SBs [35]. *Top Right:* The CLB's interfacing with the switch matrix (the SB of Xilinx®) [13]. *Bottom Right:* Drawing of an SB.

Apart from the programmable internal routing inside the device, the FPGA also employs a series of blocks that implement the interfacing with whatever is connected to its Ball Grid Array (BGA) pins. Depending on its size and capabilities, an FPGA can receive (send) signals from (to) many different parts of the PCB that hosts it. The voltage levels and frequencies of these signals cover a wide range of standards and protocols. These blocks are the so-called *I/O Blocks*, and can mediate differential (e.g. LVDS, differential SSTL) and single-ended signals (e.g. LVCMOS, TTL). Each pin that the FPGA uses to interface with the PCB that is on, is connected to one I/O block[6],

[6]Which in turn is connected to an SB inside the FPGA fabric, thus allowing the FPGA's logic to

and belongs to the so-called *bank* [13]. Each bank has fifty I/O pins, with a common voltage supply, that sets the upper limit on the maximum voltage the pins on that bank can drive, or receive. Banks are divided into two groups: High-Range (HR) and High-Performance (HP) banks. HR banks support a wider range of voltages (up to 3.3 V), with respect to an HP bank that can reach up to 1.8 V. High-Performance banks though, are designed in such a way as to support a wider spectrum of frequencies, and are thus being used in high-speed interfaces (e.g. Gigabit Ethernet), where signal integrity is of utmost importance.

3.1.3 Clock Management Tiles - Phase Locked Loop/Mixed Mode Clock Manager

One of the other important parts of the FPGA fabric, are dedicated sub-components that are responsible for generating *clock* signals for the device's logic. These are the so-called Clock Management Tiles (CMT) [13]. Every CMT utilizes a set of circuitry that deliver clocks of configurable frequency, with acceptable jitter, and with minimal phase error. CMTs are organized in tiles, and are often closely tied with specific FPGA I/O blocks. This is because CMT elements operate under a reference clock, which is usually provided from an external source, hence the need to use an I/O pin to drive the reference clock to the CMT. These clocking-dedicated I/O ports are able to deliver the external reference to the CMT with minimal delay, jitter and noise. Inside the FPGA's backbone, the so-called *Global Clock Trees* are connected to the CMTs, and distribute the clocking to all of the FPGA's elements with negligible latencies. There are also more localized clock distribution networks, that receive the clocks from the global routing, and drive the nearby clocked elements (e.g. a slice's DFFs), with the signal. For Xilinx® FPGAs, access to these resources are provided to the FPGA designer via dedicated buffer primitives (`BUFG, IBUFG, BUFGMUX, BUFGCTRL`) that receive CMT-generated clocks to their input, and fan-out the reference frequency to the desired blocks via the global or local routing.

Every CMT is comprised of two main components that are responsible for synthesizing the clocks. These are the so-called Phase Locked Loop (PLL) and the Xilinx®-specific Mixed-Mode Clock Manager (MMCM). These elements, receive a reference clock at their input, and use that to generate a number of other related clocks at their outputs, and distribute them via global or local buffers to the logic. FPGAs are usually organized in the so-called *Clock Regions*, which define the maximum range of local clock buffers, after which the clocks that are driven by them start having non-negligible delays that may have a negative impact on the circuit's timing. An overview of the clocking

interface with the outside world.

resources and regions is shown in Figure 3.4.

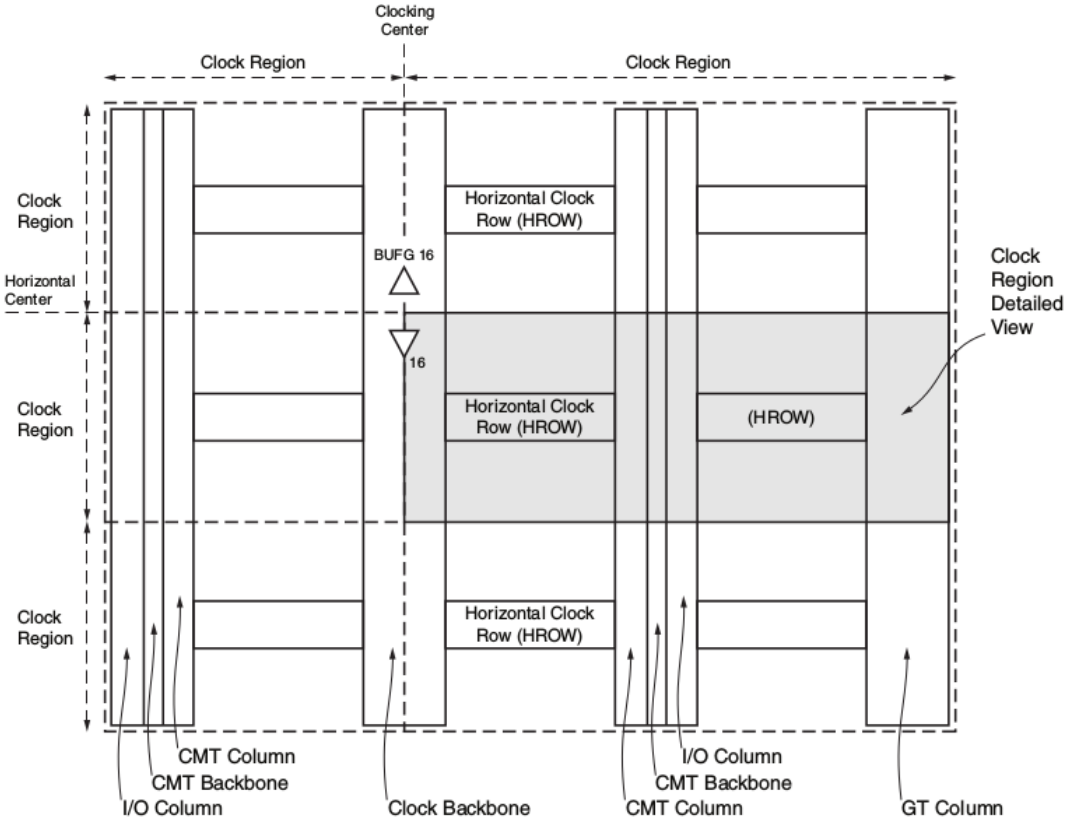


Figure 3.4: Graphical representation of a Xilinx® 7-Series FPGA CMTs, clock regions and clock trees.

The MMCM is a Xilinx®-dedicated version of the PLL, so the functional description that follows applies to both. PLLs belong to a well-established family of circuits that are used as clock synthesizers and can create a wide range of frequencies, given a reference signal at their input [13, 34]. They also work as jitter filters, and as the basic elements for Clock and Data Recovery (CDR) procedures[7]. PLLs have four basic elements: a phase detector, an amplifier, a Voltage-Controlled Oscillator (VCO), and a Multiplier. The PLL receives a reference clock in its input, which drives into the VCO. The VCO is a circuit that in turn produces a clock itself, that the PLL controls, in terms of phase and frequency. This clock is fed-back to the PLL, and the module uses its phase detector to probe the phase difference between the input reference clock and the VCO's feedback. The phase detector is able to identify the amount of difference of the time-of-arrival

[7]Mostly used for high-speed serial communication, CDR PLLs are deployed in transceivers; they receive an input datastream pattern, and generate a clock out of it, that is used by the receiver to sample and deserialize the inbound bits.

between the rising edges of both signals, and the bigger the difference, the longer the logical pulse it produces stays high. In essence, this means that as long as the two clocks are in phase, the phase detector keeps its output grounded. The detector's output is driven to the VCO, which uses it to fine-tune its output frequency, until it receives no high signal from the phase detector's output, which means that the VCO's clock matches the input in terms of phase and frequency, and the PLL gets *locked*. The VCO's output also drives multipliers that perform frequency multiplications, and to other modules that perform divisions, to produce a variety of clocks. Each Xilinx® 7-Series FPGA PLL/MMCM unit may produce up to six clocks related with the input reference. The various sub-components of a PLL are shown in Figure 3.5.

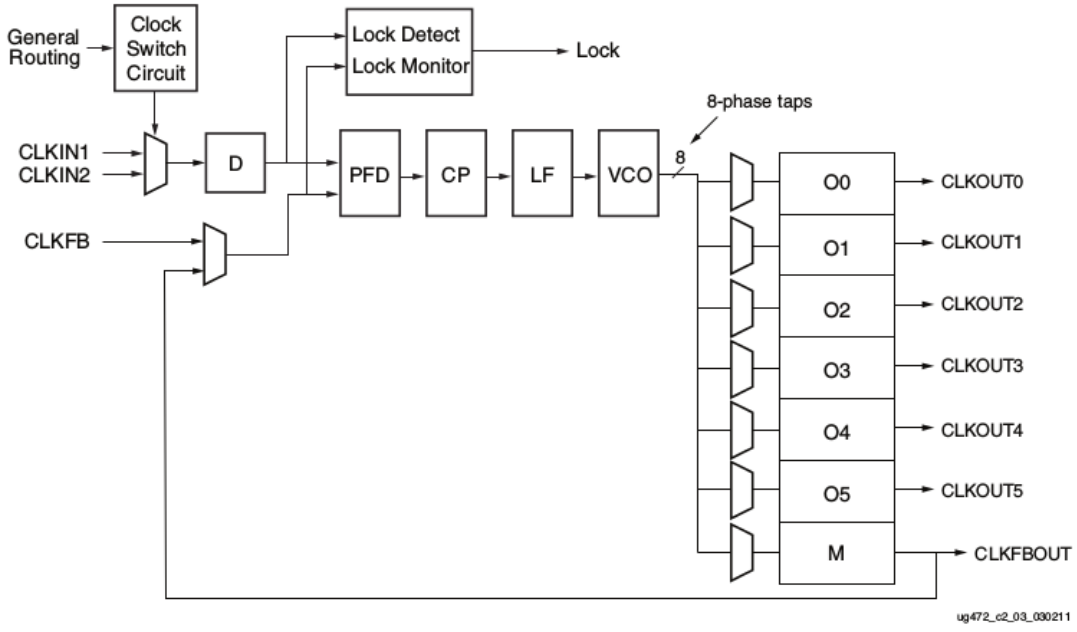


Figure 3.5: Drawing of a PLL. The VCO drives multiple outputs, that may change the phase of its generated clock, or a multiplier. The VCO receives a logical signal from the phase detector via low-pass filter that allows only more-or-less constant voltage levels to pass to the VCO. The phase detector compares the input reference with the feedback to determine if the PLL is locked or not.

3.1.4 Other Hard Blocks

Apart from general-purpose logic blocks that implement the vast majority of generic logic functions, modern FPGA architectures also make use of dedicated elements, or *Hard Blocks*, which are inferred by the FPGA vendor's EDA tool either through generic HDL code, or because the designer explicitly did so. For instance, when

a multiplication calculation has to be performed, the FPGA may use its *multipliers* for that purpose [35, 36]. Instead of reserving many CLBs to execute the said task, the device makes use of this dedicated piece of logic to save space, power, and ease timing closure on the device. Another hard block that is used extensively is that of built-in RAMs (or *blockRAMs*), which are situated in many places around the chip's die [13, 35]. As mentioned in 3.1.1, a generic CLB may be configured to store data into an address space of specific depth. Dedicated RAM modules though, are used extensively to implement synchronous read/write operations, and can actually reach up to several hundreds of Mb in memory capacity in some Xilinx® devices. Dedicated blockRAMs are used even in the simplest designs, to temporarily store large amounts of data. These primitives are usually utilized when a First-In First-Out (FIFO) memory is invoked by the user.

FIFOs are one of the most commonly used components in digital design. For Xilinx® FPGAs a FIFO is generated by creating a set of auxiliary logic around a group of blockRAM primitives. FIFOs are deemed very useful when one is trying to buffer asynchronous data [34]. FPGAs use clocks extensively, and in many cases, a design may feature several frequencies that originate from different sources, which leads to the definition of many *clock domains* inside the design. When one wants to pass data buses from one domain to the other, the only safe way to do so, is via a FIFO. A Clock-Domain-Crossing (CDC) FIFO as it is called, acts like a bridge between the two domains. Its write ports operate under the source frequency space, while its read ports are clocked by the destination clock domain. An internal gray code counter is what keeps the interface in synchronization, by keeping track of the data that are written and read, managing the read and write pointers, and asserting the status signals. These are the *empty* and *full* signals, that are synchronized with the read and write domains respectively. When the *empty* signal is de-asserted, the reading logic is aware that there are data-to-be-read, and extracts the data that were written by the asynchronous write domain. The writing logic, usually probes the *full* signal's status, and if that is raised, it knows that an overflow is imminent, which should prompt it to stop writing into the buffer and wait for it to be read by the other domain. These are the two handshaking signals that are implemented by the FIFO's logic, and keep the transactions error-free.

Finally, another type of hard block that is used extensively, is that of *Transceivers*. Due to the industry's trend to transition from older parallel communication schemes to rapid serial interfacing systems, high-end transceivers are one of the most valuable features of modern FPGAs. Due to the increasing demand in throughput, parallel schemes of interfacing between chips have now become redundant. Their limitations, which come in the form of difficulty of link synchronization at high speeds, increased board

complexity, and noise related to simultaneous switching output effects, have pushed serial schemes quickly up to prominence. Xilinx® 7-Series FPGA transceivers for instance, can serialize and deserialize bits at datarates that can reach up to $28.05\,\mathrm{Gb/s}$, and implement a variety of protocols, such as Ethernet via copper or fiber, Interlaken, SATA, or PCI Express [13]. In principle, modern transceivers use advanced analog front-end circuitry that perform signal conditioning, which is absolutely necessary given the minimal rise and fall times of the serial datastreams that are exchanged between different nodes. These analog components are often configurable, to make them adaptable for different types of applications. Fast interfacing schemes usually demand a DC-balancing line code, to avoid baseline wander effects, that may induce bit-flips. One of the most commonly used type of line code is the so-called *8b10b encoding* [33], that guarantees a balance between zeros and ones over a transmission line, thus keeping the overall DC level at manageable levels (see also Appendix B). Hence, transceivers usually implement built-in encoders and decoders of these types of protocols, to transform the data from one domain to the other. Also, these sophisticated blocks utilize dedicated, high-performance PLLs, that either generate a low-jitter clock to serialize the data that are being transmitted, or receive the inbound data as a reference, and generate a clock that is being used by the receiver to deserialize the RX data. The interested reader is referred to [40] for a detailed overview of serial communication schemes, and for a more in-depth description of the transceiver functionality.

3.2 Design Flow

The procedure which translates a human-readable HDL into active logic elements and interconnect inside the FPGA, has been an active domain of research for several years [35–37, 41]. The general scheme that is followed by any EDA tool is:

- `Logic Synthesis`: This is the process that the EDA's *Synthesizer* follows, in order to convert the HDL source code into a set of conventional combinatorial (e.g. `OR`, `XOR`, or `NAND` gates) and sequential (i.e. flip-flops, and sometimes latches) elements. After choosing the primitives, the tool also performs the necessary connections between them.

- `Technology Mapping`: After creating the generic logic gates, the synthesizer converts them to their physical form, which actually exist in the FPGA. In Xilinx® 7-Series FPGAs for instance, the combinatorial elements are mapped into LUTs, while the sequential pieces of logic are converted into slice DFFs, or less frequently, to latches. This is also the step where the various hard blocks that will be used are chosen.

- `Packing`: This is the step where LUTs and DFFs that are tightly affiliated in terms of logic, are packed (or grouped) inside the same, or neighboring, CLBs.
- `Placement`: In this stage, the EDA's *Placer* allocates the CLBs, I/O blocks, and hard blocks into specific physical elements inside the FPGA die.
- `Routing`: The most complex and time-consuming of all stages. After placing all primitives into the device, the SBs/CBs are programmed in order to connect all elements together. This is performed by the *Router*, which employs timing estimation and optimization algorithms to determine the necessary length of each interconnection, in order to avoid timing violations, which would result in system failure.
- `Bitstream Generation`: When the final FPGA architecture has been determined by the EDA tool, the information is downloaded to the FPGA via the so-called *bitstream*. This serial dataset is parsed into all elements of the device that must be activated in a certain way, in order to implement the final circuit.

Conclusions

In this Chapter, the architecture and functionality of a Field-Programmable Gate Array was described. FPGAs are used in many applications that demand high performance today, and ATLAS is no stranger to this, as it utilizes Xilinx® FPGAs in several parts of its experimental setup. The FPGA uses its designer's HDL source code as an input, in order to activate its logical elements and programmable interconnect, which are implemented in its fabric in abundance. The EDA tool that each FPGA vendor distributes, converts the high-level code into logic gates that are mapped inside the silicon, and programs the chip accordingly. The ability to perform this for practically an infinite amount of times, make FPGAs a very attractive solution to many applications, since it reduces the cost of fixing bugs, and permits on-the-fly changes of the device's features. These types of devices were used and programmed extensively for the studies related to this work, which will become evident in the second part of this dissertation that follows.

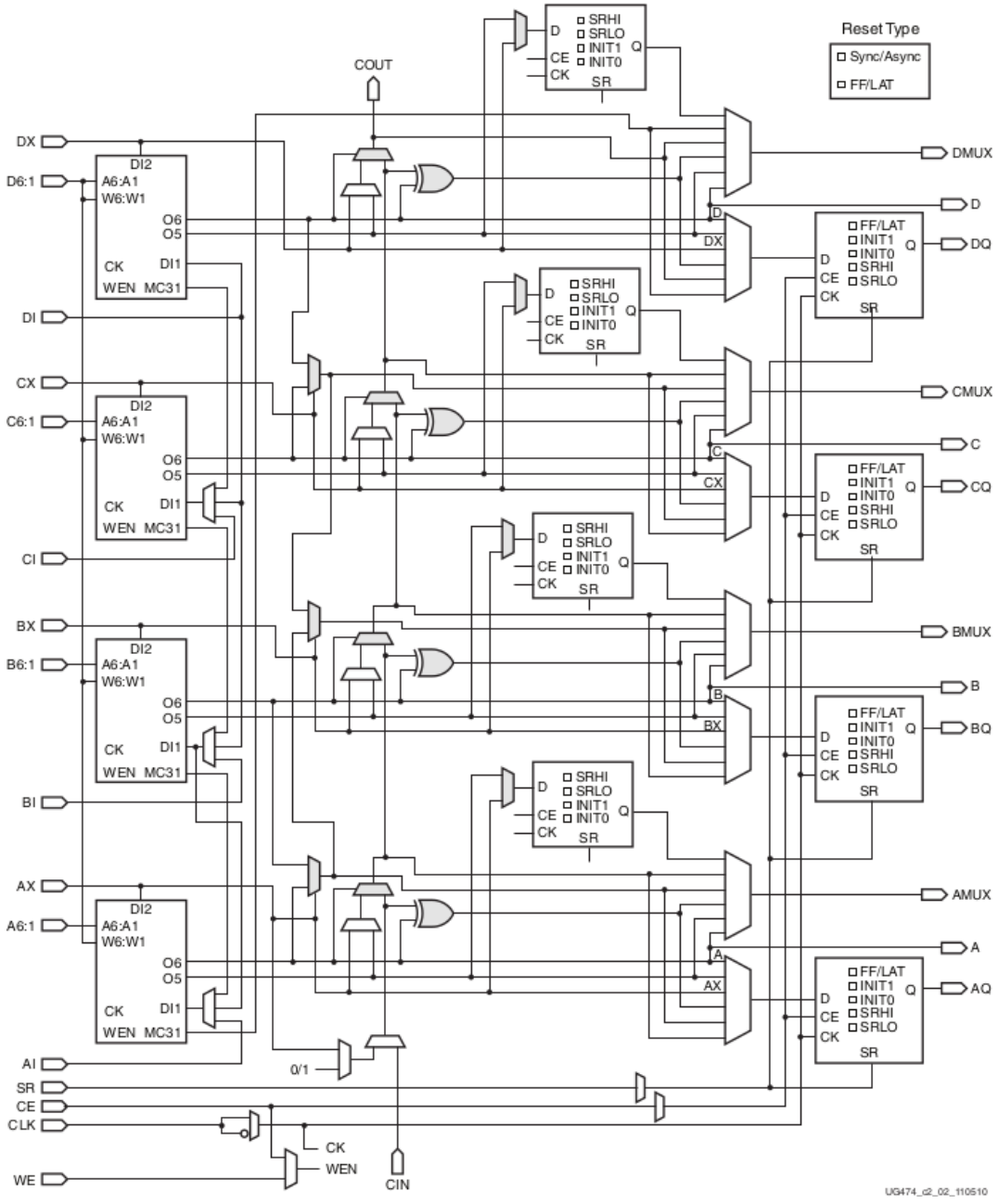


Figure 3.6: A Xilinx® 7-Series FPGA slice. On the left column one can see the four LUTs, and one the right the DFFs (which can also be programmed as a *Latch Register*, though this is not so common). The ALU is in the center of the block, alongside its registers. Several multiplexers select different outputs from the slice's primitives. The clock (denoted as `CLK`) path is also visible, that drives the related flip-flop inputs. On the very left and right, the general inputs/outputs of the slice are depicted, while on the top and bottom one can see the dedicated carry bit input/output pin. This is the basic primitive of the said FPGA technology (which does not differ a lot from other FPGA instances), and it performs the vast majority of logical operations, as inferred by the designer's HDL code [13].

Part II

Contribution to the Research and Development of the New Small Wheel Electronics and Data Acquisition System

Chapter 4

The VMM Front-End FPGA Firmware

As mentioned in the previous part, one of the key elements of the NSW electronics scheme is the VMM ASIC [9,26,27] (see Section 2.2), a mixed-signal front-end device. In the final NSW electronics system, it is read-out by the Read Out Controller (ROC) [42], and configured by the Slow Control Adapter (SCA) [43] ASICs (see Chapter 7). The need for developing and deploying other ASICs to interface with the VMM, mainly stems from the fact that the number of front-end boards that host these chips and will be deployed in the final system is very large. For instance, for Micromegas alone, 4096 boards will be produced to instrument all 32 Micromegas sectors. These are the *Micromegas Front-End Board 8 (MMFE8)*, that will bear eight instances of the VMM and one instance of each of the two aforementioned chips, that configure and read-out the VMM.

The VMM ASIC however, has been in development since 2012, long before the readout and configuration chips were made available. The VMM eventually went through three major revisions and a minor one, all of which could not had been conducted, if a reliable testing and characterization platform did not exist. This gap was filled by the VMM Readout System (VRS) [44], which employs FPGA devices that reside on the same board that the VMM is on. This FPGA is able to configure, read-out and calibrate the ASIC, and adapt its functionalities to different implementation scenarios (i.e. testbench and testbeam use-cases), and various boards or FPGA packages. The VMM's flexibility allows it to read-out effectively a variety of particle detectors that present some common characteristics. For that reason, a significant number of different front-end boards that bear VMMs and FPGAs were designed and fabricated, in order to test the response of

the ASIC under various operating conditions. Some of these boards can be viewed in Figure 4.1.

Figure 4.1: Three boards that bear the VMM and the FPGA, which hosts the firmware that will be described in this Chapter. On the top-right, one can see the MDT_446, on the top-left, the MMFE1 (see Appendix C), and on the bottom, the FPGA-based MMFE8. The most prominent components of these boards are the FPGA, the VMM, and the Ethernet controller/connector.

The FPGA's flexibility due to its reconfigurability makes it ideal for the application described. Its functionalities can be altered dynamically, in order to adapt itself into the various scenarios and testing schemes, which are subject to change, depending on the needs of the project. In this Chapter, the firmware that has been developed for the FPGA-based VMM Front-End boards will be described. The said design was created to test the VMM's functionalities, in order to assist the VMM's designers into the development process. The other requirement set to the firmware, was to be able to read-out any detector medium that is instrumented with the said boards, in order to validate the VMM's performance in real conditions. The FPGA design emphasizes on flexibility; that is, using the same source files, the firmware can be ported to different boards and FPGA packages, with some rudimentary changes in global implementation

parameters. This approach eases the development effort.

The VRS Front-End FPGA Firmware [45] has two main interfaces: The first one is its front-end interface with the VMM, via which it configures, reads-out and calibrates the ASIC, and the other one is the back-end 1 Gb/s Ethernet interface with a software host (see Appendix D), from which it receives commands, and to which the FPGA forwards the data it extracts from the VMM. If the setup is being used to evaluate the performance of the VMM on the bench, then no other component is needed to perform the measurements. If however, the user wishes to use the VMM to read-out a detector medium, then the front-end board that bears the VMM should be attached to the said chamber to complete the scheme. For the purposes of this dissertation, it is assumed that the VMM is being used to read-out the Micormegas detector, which is the detector that the ASIC was initially designed for. A rough block diagram of the scheme can be viewed in Figure 4.2

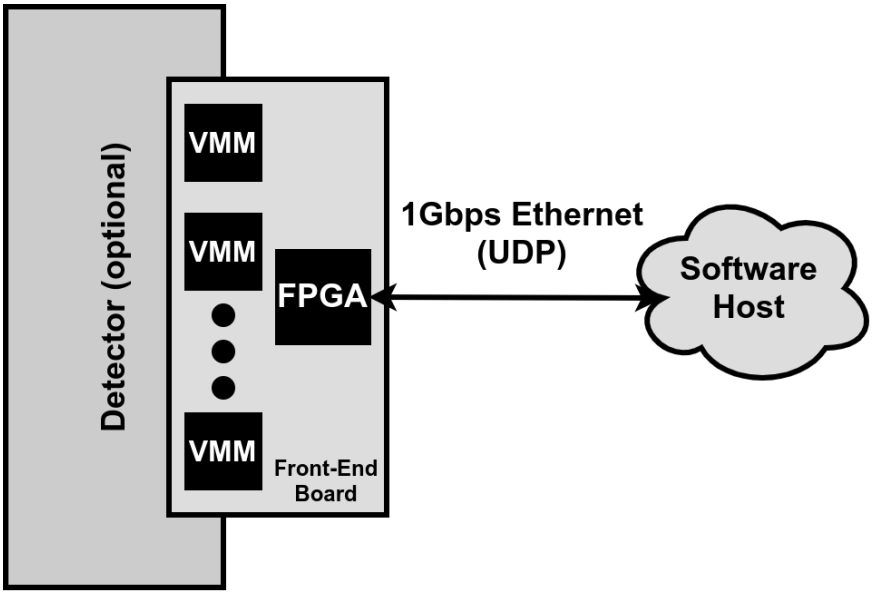


Figure 4.2: General scheme of the setup that the VRS Front-End FPGA Firmware is being used. The FPGA interfaces both with the VMM, and with a computer that hosts the VRS Ethernet Readout Software (VERSO).

To sum up, the firmware implements two distinct *paths*:

- `Configuration Path`: Via this path, the user forwards *Slow Control* commands to the FPGA, that are used to dynamically switch the FPGA logic's states and functionalities, and alter the VMM's configuration via the SPI protocol. These data are mediated via 1 Gb/s Ethernet over the User Datagram Protocol (UDP), where the one node is the VMM Readout System Ethernet Software (VERSO),

and the other one is the FPGA itself. VERSO sends the commands to the FPGA, which in turn decapsulates the Ethernet frames, before evaluating the data in the user payload.

- `DAQ/Readout Path`: In most use-cases, the VMM is configured in digital readout mode, in which the input pulses from the detector or from the test-pulser circuitry are digitized and stored in internal buffers. The FPGA then issues a trigger associated with an event-to-be-read-out, and extracts the relevant data from the VMM's buffers before forwarding them to VERSO. There are two ways to read-out the VMM digitally, one is the *Continuous Readout Mode*, and the other one is the *Level-0 (L0) Readout Mode*. The latter will be used in the final implementation and will be discussed more thoroughly in this study. The other way to read-out the VMM is the *Two-Phase Analog Readout Mode*, where the VMM does not digitize the input pulses, but presents the analog measurements to an external Analog-to-Digital Converter (ADC) that digitizes them instead. This mode is slower to operate than the L0 mode, but since the external digitizing module that is used is the FPGA's 12-bit XADC, it allows for more precise measurements, compared to the VMM that employs internal $10-$ and $8-$bit ADCs. This readout method, alongside the aforementioned digital data extraction techniques, will be described in greater detailed in Chapter 5.

4.1 Overview of the Firmware Logic Blocks

The main modules of the FPGA design are shown in Figure 4.3, and are listed and briefly described below.

- `Flow FSM`: This component is the supervisory state machine of the FPGA's logic. It interfaces with all other building blocks of the design and defines the overall state in which the system is, depending on the user's directive as it is forwarded by the *UDPdin_handler*.

- `Ethernet Wrapper`: Partially based on the *1Gb/s UDP/IP Stack* design from OpenCores [46], this sub-module provides interfacing capabilities with the software host. Modifications have been made to the original design to accommodate some Internet Control Message Protocol (ICMP) functionalities. The block also deploys the Xilinx® GTP Transceiver and Tri-mode Ethernet Media Access Controller (TEMAC) IP cores that interface with the on-board PHY chip to provide $1\,\mathrm{Gb/s}$ Ethernet-over-copper connection with the software host.

- `UDPdin_handler`: This module registers the incoming UDP payload data from the software and applies the values in internal configuration registers accordingly. It also forwards any commands for switching the system's state to the Flow FSM. Furthermore, it receives the VMM configuration data, and transmits them as they are to the ASIC via the SPI protocol.

- `FIFO2UDP`: This block accepts the user-data-to-be-sent to the software host via the UDP/Ethernet Interface. The main feature of the component is a Clock-Domain-Crossing (CDC) FIFO that receives the data that are synchronized with the main clock domain of the design, and transfers them to the Ethernet clock domain, that is in sync with the Transceiver's clocking.

- `Clocking and VMM Test-Pulse Block`: This module generates the BC clock (CKBC) and the test-pulse signal (CKTP). The CKBC is used as the VMM's reference clock, while the CKTP is driving the VMM internal test-pulser. The test-pulse strobe is generated by the FPGA synchronously to the CKBC. The reference clock source may be an on-board oscillator or an external clock source. The latter is used in multiple-board readout schemes to ensure synchronization between independent boards. This module is also responsible for the timing calibration of the ASIC.

- `VMM Readout Wrapper`: This part of the logic deploys different sub-blocks (readout cores) that extract data from the VMM. Supporting all possible readout modes of the ASIC, it receives a trigger signal that initiates a readout cycle from the *Packet Formation* module and forwards this trigger to the VMM accordingly. Depending on the board implementation, the wrapper can instantiate more than one readout cores that can extract data from multiple on-board VMMs.

- `Packet Formation`: This module lies one hierarchical level above the *VMM Readout Wrapper*. It forwards trigger signals to it, as received by the *Trigger Module*. It implements an internal trigger counter that the software uses to group events from multiple boards, when applicable. This module also aggregates the VMM data from the readout core(s). After accumulating all VMM data, it builds the event packet and finally forwards the data to the software via the *FIFO2UDP* component.

- `Trigger Module`: This is the module that receives a trigger signal from either an external source or from the *CKBC/CKTP Generator*[1]. This trigger signal is then forwarded to the *Packet Formation* module, which eventually drives it to

[1]The CKTP is also used as a trigger signal in the case of internal-generated-trigger mode, primarily applicable when acquiring data using the VMM's test-pulser.

the VMMs via the Readout Wrapper logic blocks.

- `XADC Module`: This module implements the Xilinx® IP core of the same name. It essentially consists of a wrapper of the integrated 12-bit ADC that is driven by the VMM analog outputs. The only part of the logic written in Verilog HDL, the XADC is used when calibrating the threshold and pulser DACs of the VMM, or to sample the voltage variation of the VMM's channel inputs, a measurement indicative of the noise each channel is subjected to. Finally, the module is responsible of digitizing the PDO and TDO voltage levels when acquiring data from the VMM in the Two-Phase Analog Readout mode. Whichever the case may be, the samples are forwarded upstream to the software which handles them as typical data.

- `IP Configuration Modules` (not depicted in Figure 4.3): When using multiple readout boards within the same network domain, the ability to easily modify each FPGA's host IP and MAC address becomes a necessity. The desired IP address, which is configured by the software, is registered via the SPI or the I^2C protocol. Upon a power start, the same module extracts the previously written address from the memory and stores the local IP and MAC addresses.

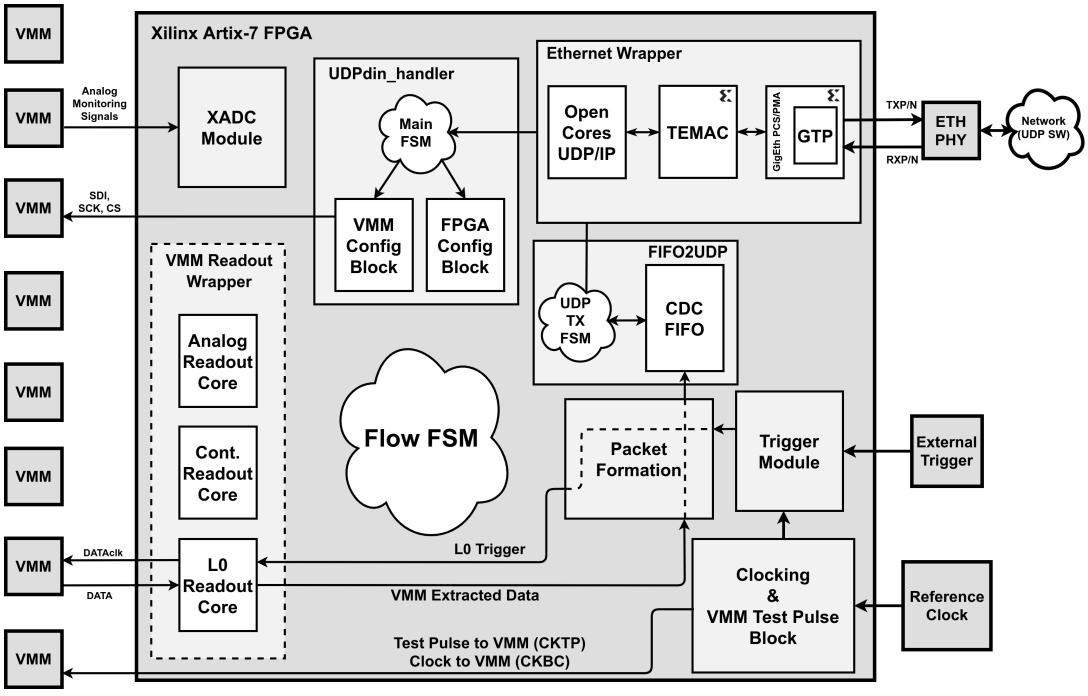


Figure 4.3: Block Diagram of the VMM Front-End Readout Firmware. Some of its components have been omitted to simplify the diagram, but they are mentioned in the main body of the text.

In the sections that follow, a more in-depth description of the most important logic blocks of the VRS Front-End FPGA firmware will be provided. The main focus will be on the architecture of the modules that implement the interfaces - both for the Ethernet communication, and for the extraction of the VMM data, as well as on the components that calibrate the VMM ASIC.

4.2 Ethernet Communication - Configuration and Read-out Paths

In this section, a more detailed description of the parts of the firmware that mediate the Ethernet data to and from the FPGA will be given. These include the modules that implement the actual connection with the network (i.e. the PCS/PMA, the TEMAC, and the OpenCores UDP/IP Stack - all part of the *Ethernet Wrapper*), and the surrounding user logic that handles the inbound and outbound data packets (i.e. the UDPdin_handler and the FIFO2UDP).

In order to connect the FPGA with other nodes in an Ethernet network, it is mandatory to implement some of the so-called Open Systems Interconnect (OSI) Layers (see Appendix B.1) inside the FPGA logic. The OSI model describes the different functions that a computational system must apply in order for it to be part of a communications scheme with other nodes that comply with the same set of rules. The OSI model dictates a certain hierarchy, thus dividing the different functionalities into distinct *layers* [12]. The layers that are implemented in the FPGA logic are: *Transport, Network, Data Link* and *Physical*. Some of this logic, comes in the form of hard blocks inside the FPGA (i.e. Xilinx® Transceivers), or components deployed in the general-purpose FPGA fabric.

As it must be evident by now, the data that are being extracted from the VMM(s), are aggregated by the FPGA, and subsequently formatted in such as a way as to be transmitted via Ethernet to the software host that is connected to the same network the FPGA is in. For this reason, the data are first being encapsulated inside a UDP datagram, according to the associated standard, which is part of the OSI's Transport Layer. Then, the UDP payload (i.e. the VMM data) alongside its header are forwarded to the Network Layer, where the Internet Protocol version 4 (IPv4) standard is implemented in the firmware. Both of these functionalities are being carried out by the *UDP/IP Stack* module, from OpenCores.org [46]. In order to implement the Data Link and Physical OSI Layers, two Xilinx® IP cores are deployed. The Tri-mode Ethernet Media Access Controller (TEMAC), and the Physical Coding Sublayer/Physical Medium Attachment (PCS/PMA) which also infers a Xilinx® GTP

Transceiver that is configured to communicate with an external Ethernet PHY device at a bandwidth of 1 Gb/s [47]. The deployment of these three cores, inside the *Ethernet Wrapper* of the design (see Figure 4.3), allows the user to build a full networking schema inside the FPGA, that implements 1 Gb/s Ethernet-over-copper.

4.2.1 PCS/PMA & TEMAC

The three blocks that implement the Physical and the Data Link OSI Layers will be discussed briefly here. They are all IP cores from Xilinx®, and their connectivity is as follows: TEMAC ↔ PCS/PMA ↔ GTP Transceiver. The scheme can also be inspected in Figure 4.4. The Transceiver is a dedicated hard block inside the FPGA that can mediate serial data via differential lines at a very high speed. Most FPGAs today present a heterogeneous architecture, largely consisting of configurable logic[2], and of dedicated electronic sub-modules, or hard blocks, that most of the time are instantiated explicitly by the designer. One of the key features of FPGAs is high-speed serial I/O blocks, or Transceivers. For instance, the Xilinx® Artix-7 FPGA that hosts the design under discussion, has sixteen available Transceivers that can be deployed in specific places inside the fabric, in order to establish several types of high-speed serial communication schemes, such as Ethernet-over-copper, Ethernet-over-fiber, or Peripheral Component Interconnect Express (PCIe). For this application, the GTP Transceiver is inferred by the PCS/PMA IP core, to implement 1 Gb/s Ethernet-over-copper, and communicates via two differential pairs (one for sending and one for receiving the serial stream), with another off-chip Transceiver, called Ethernet PHY. This is the Marvell® 88E1111 Integrated Ultra Gigabit Ethernet Transceiver. This connection allows the Ethernet frames of the network that the FPGA is connected to, to enter the FPGA, and also provides the device the ability to forward Ethernet frames to that same network. The way that the data are being mediated between the Transceiver and the external Marvell® PHY chip, is via Serial Gigabit Media Independent Interface (SGMII), which is a subset of the Gigabit Media Independent Interface (GMII), which is parallel and is operated internally in the FPGA logic (see Figure 4.4) [13]. The Marvell® chip implements the SGMII-1000BASE-T protocol to communicate with the FPGA, and the PCS/PMA module is configured to comply with that scheme[3].

As presented also in Appendix B.1, most high-speed serial communication schemes, usually require the implementation of some type of *DC-balancing* line code that transforms the serial datastream into something that is manageable, both in terms of

[2] Usually inferred by HDL code.

[3] 1000 denotes a speed of 1 Gb/s, while the -T implies the chip operates an Ethernet-over-copper interface.

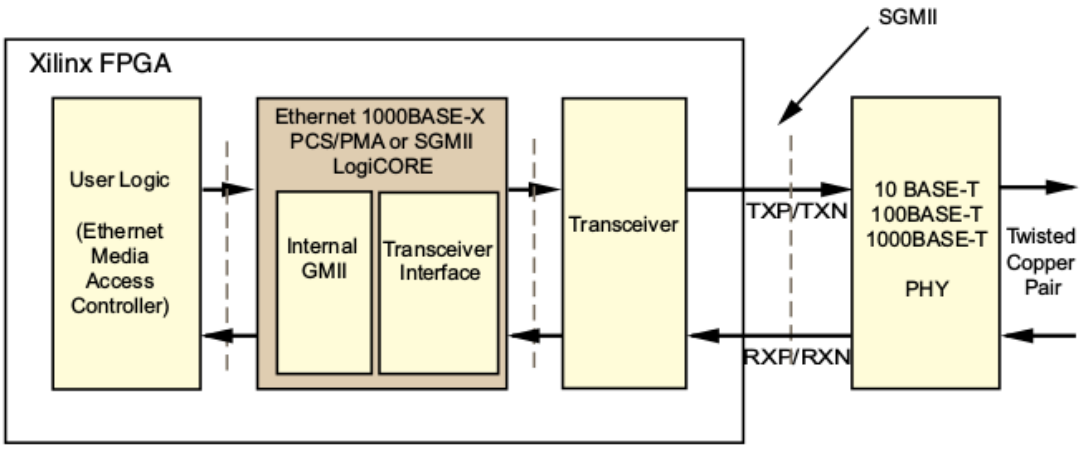


Figure 4.4: Figure depicting the connectivity between the two Xilinx® IP cores that implement the OSI Physical Layer, as well as their parallel (GMII) interfacing with the User Logic (for this implementation it comes in the form of the Xilinx® TEMAC IP core), that implement higher levels of the OSI model, and their serial (SGMII) interfacing with the external Ethernet PHY chip [13].

baseline voltage[4] [48], and in terms of clock recovery by the receiver. For this reason, one of the most common line codes that provide DC-balancing is used to encode the data to and from the FPGA Transceiver: this is the 8b10b encoding. There is a constant presence of comma or idle characters in the datastream, that allows the Transceiver to recognize the word boundaries, thus ensuring correct phase alignment with the inbound data - this is how the device ”knows” when it is best to shift-in 10 deserialized bits from the line into its 8b10b decoder. Also, the 8b10b line code ensures that there are enough transitions in the datastream, in order for the FPGA Transceiver to recover a clock based on the data coming from the external Ethernet PHY Transceiver chip.

For the design at hand, the timing of all the modules that implement the Ethernet interfacing inside the FPGA, is being done by a clock that the Transceiver recovers from the inbound datastream originating from the Marvell® Ethernet PHY chip. The Transceiver receives a reference clock, with a frequency of $f_{ref} = 125\,\text{MHz}$ into its dedicated input port *(MGTREFCLK)*, originating from a high quality on-board crystal oscillator[5]. The Transceiver's PLL then uses that frequency to synthesize two internal clocks, in-phase with the inbound datastream. These are called *userclk* and *userclk2*

[4]The term *Baseline Wander* as it is called, is the slow variation in the average of a signal's waveform, and can result in bit errors during the deserialization of the digital datastream.

[5]The clock actually originates from a crystal (Abracon® ABM8-25.000MHZ-B2-T) that drives the input of a Crystal-to-LVDS frequency synthesizer (Renesas® 844021BG-01LF), with a $f_{ref} = 25\,\text{MHz}$ clock. Using that, the synthesizer creates the high precision, low-jitter $f_{MGTREF} = 125\,\text{MHz}$ reference for the FPGA's Transceiver.

with frequencies of 62.5 MHz and 125 MHz respectively. These are the so-called *recovered clocks*, and are used to deserialize the inbound datastream, and clock the Transceivers' RX-side components.

The 125 MHz recovered clock is used also by the rest of the components of the *Ethernet Wrapper* to handle the inbound and outbound data-flow. The PCS/PMA receives the decoded data from the Transceiver, and hands the Ethernet frame to the TEMAC block. The TEMAC is a Xilinx® IP module that implements the Data Link Layer functionalities, as per the IEEE 802.3™Ethernet standard [49]. The module performs some processing on the inbound Ethernet frame, and if the prerequisites are met (e.g. if the destination MAC Address matches that of the FPGA's, and if all error-checking procedures using the frame's CRC/FCS fields are satisfied), the frame's contents are presented one layer up, to the UDP/IP block, that implements the Transport and Network Layers of the OSI model. For the outbound direction, the TEMAC accepts the IPv4 datagram from the UDP/IP block, and builds an Ethernet frame that is then subsequently forwarded to the Physical Layer, implemented by the PCS/PMA.

4.2.2 OpenCores UDP/IP Stack and Modifications

Downloaded from Opencores.org [46], the *1G Ethernet UPD/IP Stack* is a VHDL-based solution that implements the Network (IPv4) and Transport (UDP) Layers of the OSI model. It implements two interfaces, one for receiving and forwarding the Ethernet frame contents (TEMAC), and one for receiving and forwarding the UDP data/payload (User Logic). The latter comes in the form of two distinct components, each handling either data-flow direction. One of them evaluates inbound UDP traffic, usually originating from the software host (VERSO), named *UDPdin_handler*, while the other is named *FIFO2UDP*, and is responsible for forwarding any data from the FPGA (usually data extracted from the VMM) to the network, in order for the software host to receive and process them.

All of the aforementioned components, operate on the Ethernet clock domain, synchronized by the Transceiver-recovered-clock, *userclk2*, at $f = 125$ MHz. The UDP/IP Stack's interface with the User Logic side, for both TX/RX, follows the same scheme as its own TX/RX interface with the TEMAC. The latter module, forwards the inbound Ethernet frame contents to the stack in the form of *bytes*, alongside a *valid* signal, that denotes which bytes should be registered by the stack, and a *last* signal that is raised for a single clock cycle on the last byte of the forwarded Ethernet payload. In a similar manner, the stack forwards the Ethernet frame-to-be-transmitted to the TEMAC by issuing *valid* and *last* signals alongside the bytes themselves. As mentioned above, the FPGA User Logic that handles the TX/RX UDP payload datastream of the stack,

follow the same communication scheme, a timing diagram of which can be inspected in Figure 4.5.

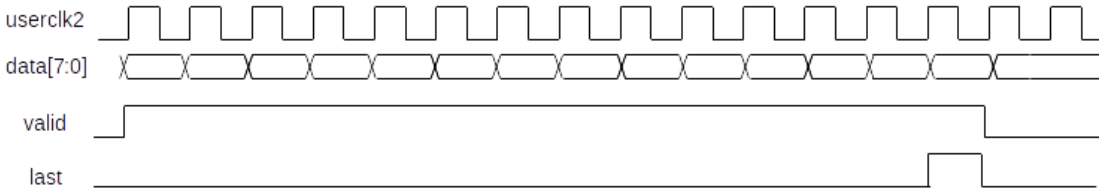


Figure 4.5: Timing Diagram of the Ethernet data as the TEMAC forwards them to the UDP/IP Stack. The data are being sent on a per-byte basis, on every cycle of the *userclk2*. The *valid* signal goes high on the first byte of the data, and goes low after the last byte. The *last* signal denotes which is the last byte by going high for one clock cycle. The same scheme is followed when the stack forwards data to the TEMAC, as well as on the interface of the stack with the User Logic, for both the TX/RX.

The UDP/IP Stack's IPv4 RX block receives the Ethernet frame contents from the TEMAC, and checks the *EtherType* field of the frame (see Appendix B.1) to make sure it is equal to 0x0800, which corresponds to the IPv4 protocol. If a frame of that type is indeed received, its data are forwarded to the UDP RX block. These data carry the IPv4 datagram, header and payload. Inside the header, the *Protocol* field that occupies a byte of the IPv4 datagram's header, must be equal to 0x11 (indicates that the datagram's data correspond to the UDP protocol), in order for the IPv4 datagram's payload to be processed by the UDP RX block. If this is indeed the case, then the UDP RX block will check the UDP data using the protocol's checksum, which is in the header, and if no error is found, the payload data are presented on the RX User Logic side in the manner of Figure 4.5 alongside the *Source Port* and the *Destination Port*, which are part of the header. These signals will be processed by the *UDPdin_handler*, which decides whether the packet was sent by the software host and not some other node in the network.

Internet Control Message Protocol Add-On

One of the Ethernet protocol's most useful connectivity diagnostic functionality is the so-called *Echo Request* or *Ping Request* or just *ping*. In this scheme, the transmitter, which is a node in a network, relays the ping command to the network, and some other node that is connected, may respond with an *Echo Reply* or *Ping Reply*. The node that performed the ping in the first place will then evaluate the reply and determine if the connectivity is sound between itself and other parts of the networking scheme. This functionality is part of the Internet Control Message Protocol (ICMP), which is generally used by network devices, including routers, to send error messages

and operational information indicating success or failure when communicating with another IP address. ICMP differs from transport protocols such as TCP and UDP, as it is not typically used to exchange data between systems. The ICMP frame is encapsulated within the IPv4 datagram, and both of their structures can be inspected in Figure 4.6.

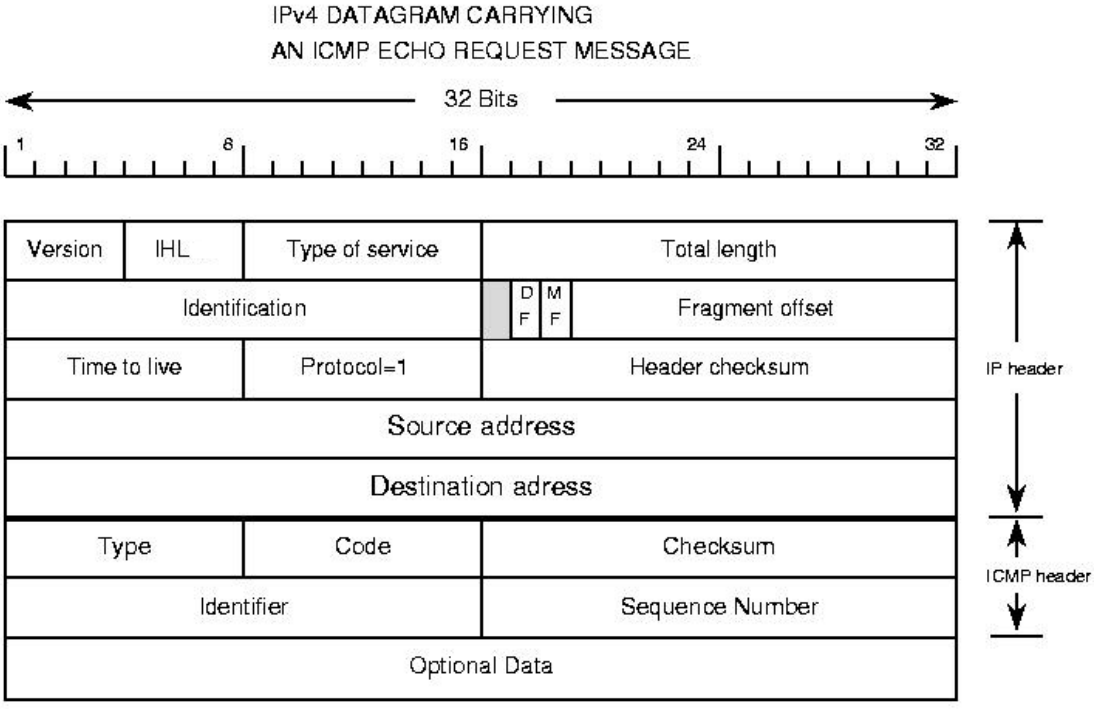


Figure 4.6: Structure of the IPv4 datagram, when that is carrying an ICMP packet. Note that the *Protocol* field of the IPv4 header equals to 0x01 when there is an ICMP packet in its data. The reader may also refer to Appendix B.1 for more information on the IPv4 datagram.

For the current application, the ping functionality can be used by pinging any FPGA in the network via the software host, to easily determine if the connectivity between an FPGA and the network is well-established. All modern operational systems implement ping services within their framework, hence, it would make sense to make the FPGA compatible with the ICMP protocol, in order for it to be able to process the packets of the said scheme. The user can then easily ping the FPGAs before runtime, and expect to see the associated replies from the chip.

Therefore, an ICMP add-on was designed, that would perform the said functions. Three components were designed, and were placed in the UDP/IP Stack. Roughly, these components first recognize if an inbound IPv4 datagram is part of the ICMP protocol, then if the ICMP packet is an echo request, its data are buffered and a ping reply is being

generated internally and forwarded to the IPv4 TX component of the stack. The block diagram of the scheme can be inspected in Figure 4.7. A more detailed description of the modules is now presented.

- `ICMP_RX:` Originally, the UDP/IP stack activates its *UDP_RX* sub-module (see Figure 4.7), only when the protocol code of the inbound IPv4 header equals to 0x11. The ICMP_RX module extends the functionality by evaluating the protocol field and if it equals to 0x01 (ICMP), it forwards the ICMP packet, which is the data of the IPv4 payload as-they-are, to the *ping_reply_processor*.

- `Ping Reply Processor:` Reads the inbound ICMP packet originating from *ICMP_RX* and scans the *Type* and *Code* fields of the packet, and if they are equal to 0x08 and 0x00 respectively, it means that the inbound packet is a ping request. The ICMP data are then buffered into a FIFO, and a reply is generated. The ping reply packet differs slightly with the ping request packet it is associated with. To be precise, the *Type* field of the reply equals to 0x00, and the *Checksum* field will subsequently be different. The data of the packet are the same, and this is why they are buffered upon reception. After putting the packet together, the processor switches a multiplexer that selects between the *UDP_TX* and the *ICMP_TX* sub-modules, and forwards the reply to the latter component.

- `ICMP_TX:` The processor builds the ICMP packet and forwards it to *ICMP_TX*, which in turn sends it to the part of the logic that builds an IPv4 datagram, to-be-encapsulated in an Ethernet frame at a later stage, before being sent via the Transceiver to the network. The software host that performed the echo request will then receive that reply, and print a message to the user that verifies the FPGA's connectivity. After sending the entire ICMP packet, the module in question signals the *ping_reply_processor* accordingly via handshaking signals, which in turn switches the multiplexer back to its default state, that allows only the *UDP_TX* module to forward data to the network.

4.2.3 UDP User Logic

The logic block that receives the inbound UDP data from the Ethernet Wrapper is the *UDPdin_handler* and its architecture can be viewed in Figure 4.8. The predefined communication protocol between the DAQ software (VERSO) and the VRS Front-End Firmware dictates that there are two configuration paths available: one for configuring the functionalities of the FPGA, and one for the VMM. As mentioned above, the UDP/IP Stack decapsulates the UDP datagram (header and payload) from the IPv4 data and presents the raw data bytes to the UDPdin_handler (following the scheme of

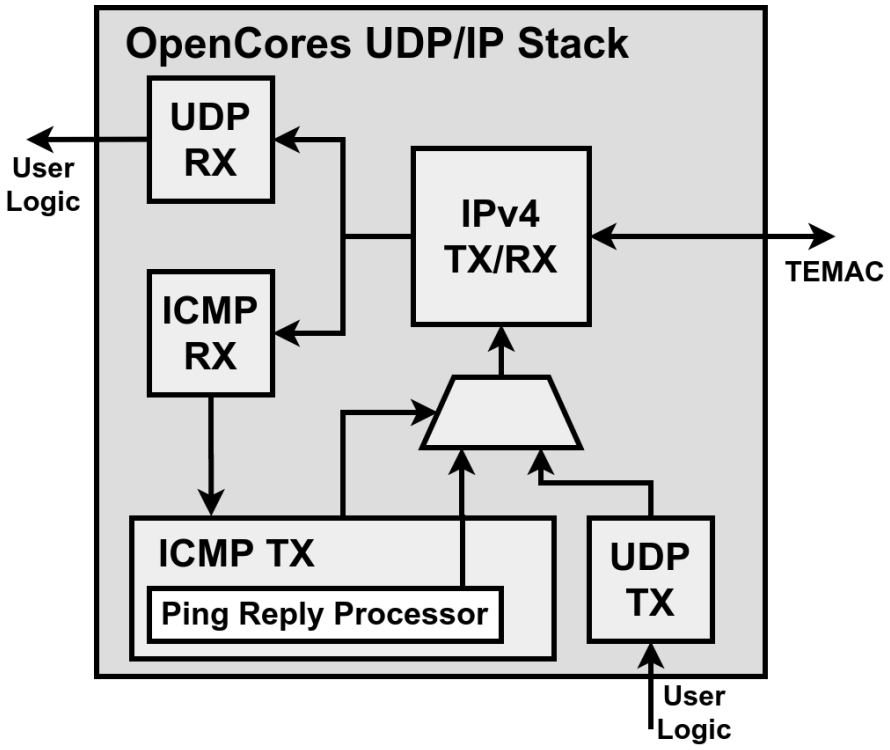


Figure 4.7: Structure of the ICMP add-on in the UDP/IP Stack. The new components are the *ICMP_RX*, *ICMP_TX* and the *Ping Reply Processor*.

Figure 4.5), alongside the UDP packet's header information. The *Main FSM* of the UDPdin_handler is activated upon the assertion of the *valid* signal from the inbound UDP datastream, and on the next clock cycle of the *userclk2*, it starts counting the valid bytes and inspects the packet's *Destination Port*, which is part of the header. The value of the port parameter first of all indicates if the packet was indeed relayed by the software[6], while it also dictates whether the inbound packet is supposed to alter the FPGA's or the VMM's address space. If the port address is equal to 0x1777, 0x19cc, or 0x19d0, then the packet is directed towards the FPGA's registers, and if it equals with 0x1778, then the packet contains VMM configuration data. After determining the value of the port, the main FSM which acts as a demultiplexer, forwards the datastream to the corresponding FIFO (one for each configuration path). The main FSM also samples the 8-byte configuration packet header which contain some fields useful for the VMM configuration, as will become evident later in this subsection[7]. After the assertion of the *last* signal by the stack, the main FSM activates the associated secondary FSM, that

[6] A small set of valid port addresses was chosen, after making sure that none of these are already used.

[7] Only one byte is being used out of eight, and the rest of the data are reserved for other functionalities, or are padded with zeroes, serving as placeholders for any future extensions.

reads its buffer until it is empty of configuration data.

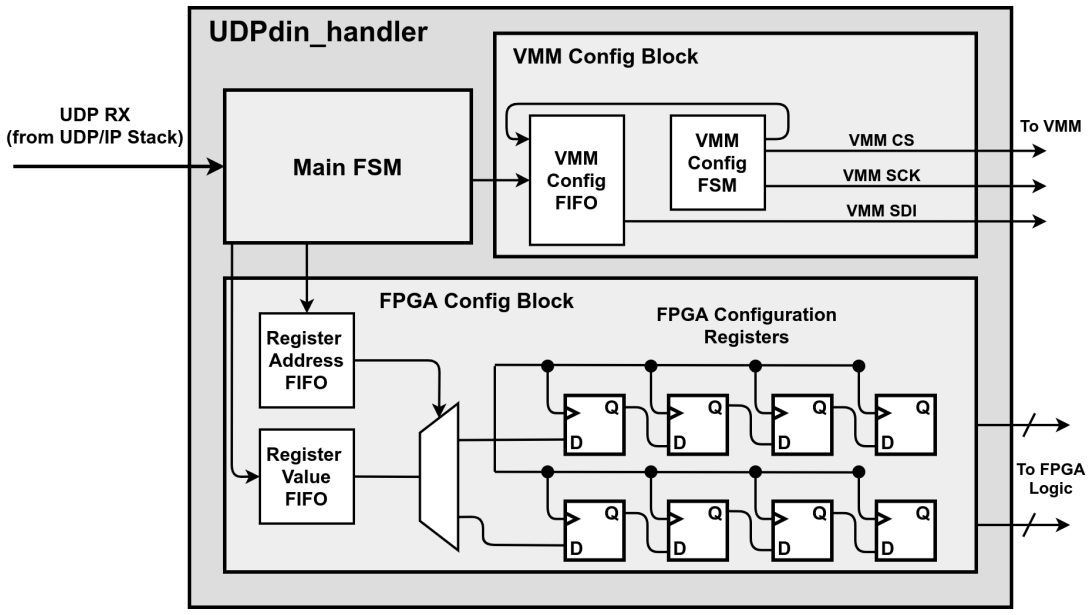


Figure 4.8: Architecture of the module that handles the inbound UDP traffic. The *Main FSM* receives the inbound traffic from the UDP/IP Stack and activates one the sub-FSMs accordingly. Each sub-FSM implements buffers that store the configuration data (the FPGA configuration sub-FSM is not depicted here).

Finally, it is worth mentioning that the said component operates in *two* clock domains. The main FSM is under the *userclk2* clock domain, that clocks the Ethernet blocks, while both sub-FSMs operate at the main clock domain of the design, in sync with the VMM's reference clock of 40.079 MHz. For this reason, the FIFOs of the secondary FSMs are implemented in CDC mode, using dual-clock blockRAMs, in order to transfer the inbound Ethernet data from one domain to the other safely. Also, the handshaking signals between each secondary FSM and the main FSM are passed through synchronization flip-flops, that ensure timing closure and operational stability, since the two domains are completely unrelated.

FPGA Configuration

The FPGA implements a variety of configuration registers which are connected with different parts of the design, in order to allow the user to dynamically alter the behavior of the FPGA's logic during runtime via the DAQ software. A timing diagram which depicts the structure of a UDP packet sent by the software that is directed towards the FPGA configuration registers can be inspected in Figure 4.9.

There are two sets of parameters present in the FPGA configuration packet: the *Register*

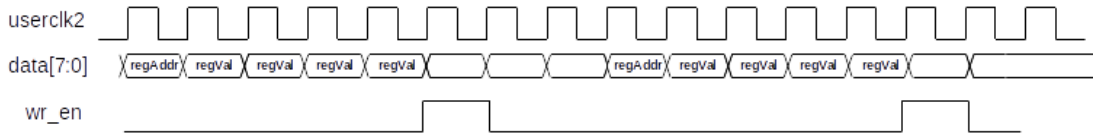


Figure 4.9: Timing Diagram of a packet that configures the FPGA address space. The address of the register is being sent (one byte) first, and the value of the corresponding register (four bytes) follows. The FSM registers the two parameters, and writes them in two FIFOs. One stores the addresses, and the other one buffers the values (see Figure 4.8).

Addresses and the *Register Values*. The communication protocol between the software and the firmware, defines the valid positions of these parameters within the UDP payload itself. As mentioned above, after the assertion of the *last* signal, the main FSM knows that all possible addresses and values have been written into the two FIFOs, and activates the FPGA configuration sub-FSM. The latter piece of logic progressively reads the two FIFOs, and uses the address to switch a demultiplexer that receives the read register value, and depending on the address, directs the value to the appropriate FPGA configuration register. The data are parsed into the registers bit-by-bit (the configuration registers have been arranged in the form of a *Shift Register*). The reason behind this architecture is because some registers tend to be wide (32 bits maximum), and if they are addressed in parallel, the demultiplexer's combinatorial logic, alongside the FPGA routing occupancy and complexity, increases. The chosen shift register arrangement however, ensures that the data path between the output register of the *Register Value FIFO* and the target register is a single bit, and is not a function of the register's width. This decreases the routing complexity and the combinatorial logic utilization of the demultiplexer. After shifting-in all bits of a register, the data are passed from the local registers to the FPGA logic, thus completing the configuration process for that register. The whole procedure is deemed done when both FIFOs are empty, after which the local secondary FSM signals the main FSM via a handshaking signal.

VMM Configuration

If the user wishes to configure the VMM's functionalities, they manipulate VERSO's GUI accordingly (see Appendix D of the current Chapter) and send the VMM configuration packet via UDP to the FPGA, using the destination port 0x1778. Upon reception of the said packet and registration of the port, the main FSM routes the UDP payload data into the associated FIFO, and after the transmission ends, the VMM configuration secondary FSM is activated. The latter piece of logic will read the FIFO, which has an 8-bit input and a single-bit output, one bit at a time. This process

essentially serializes the data-to-be-directed to the associated VMM configuration input pin, named *SDI*. The said FSM also takes care of keeping the VMM *ENA* signal grounded throughout the configuration process. The FSM implements the SPI protocol, as per the VMM's requirements, and drives the configuration bits into the ASIC directly from the FIFO's output register, while it also asserts the *SCK* and *CS* signals that drive the chip's accordingly. The FIFO of the module contains all 1728 bits of the VMM's configuration address space, which originate from the software, and are progressively forwarded as-they-are to the ASIC. In the case of a board that has multiple VMMs, the user may wish to configure different chips with specific configuration parameters. For this reason, the UDP configuration packet's header is used to dictate the VMM that the data should be relayed to. An 8-bit bitmask is present in the 6th byte of the 64-bit header, that the main FSM registers, and applies to the eight possible *ENA, SDA, SCK* and *CS* signals that drive each of the eight possible VMMs of the front-end board, in the form of a *clock-enable* signal to the associated output registers. Therefore, the sub-FSM is agnostic of which VMMs will be configured, and essentially fans-out the SPI signals to all (maximum) eight VMMs. The pre-applied bitmask only allows the configuration bits, clock, and strobe, to be propagated to the VMMs that the user desires to configure.

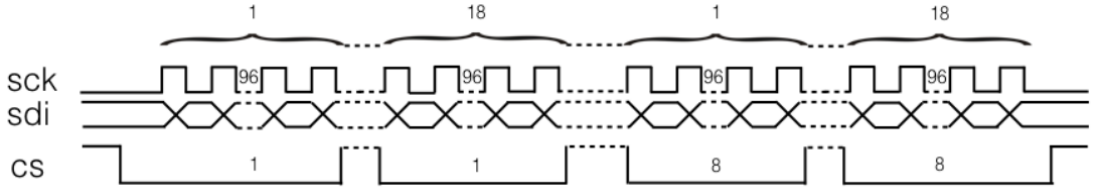


Figure 4.10: Timing diagram depicting the protocol between a device that configures the VMM and the VMM itself. For the current implementation, this is the FPGA itself, but for the final NSW electronics scheme, it will be the SCA (see Chapter 7). The procedure follows the SPI protocol. The *CS* signal is being asserted after 96 *SCK* clock ticks, and in total, 1728 configuration bits are being propagated to the ASIC [50].

FIFO2UDP

In the readout path, which follows the other direction of the Ethernet interface (this is the *TX* path, from the FPGA towards the network and the software host), the most important component is the so-called *FIFO2UDP* module. Its architecture can be inspected in Figure 4.11. This piece of logic implements one FSM operating at the Ethernet *userclk2* domain, and two CDC FIFOs, accepting data into their write ports that are synchronized by the main clock domain of the FPGA's logic (i.e. the domain related with the 40.079 MHz VMM reference clock), and outputting the data into the

Ethernet domain. The FSM is activated once a UDP packet size word is written to the length FIFO. The defined scheme dictates that the FPGA user logic writes the raw UDP payload data first, while it increments a UDP packet length counter for each written byte. After finalizing the data forwarding, the counter value is written to the packet length FIFO, thus activating the UDP packet propagation. Using the size information, the FSM reads the indicated amount of bytes from the data FIFO, and issues the control and header signals to the UDP/IP Stack accordingly. This scheme allows for *simultaneous* write and read operation of the FIFOs, thus minimizing the dead-time of the design. To elaborate more, the following hypothetical scenario should be taken into consideration: If a packet of N bytes is written, the module will start sending it to the Ethernet modules accordingly. If *more* data need to be forwarded to the software, but the FIFO2UDP FSM is busy handling the data of the *previous* event (i.e. if the bytes that have been sent is smaller than N), then the implemented architecture allows for immediate writing of the UDP data and length to the associated buffers without waiting for the previous transmission process to finalize.

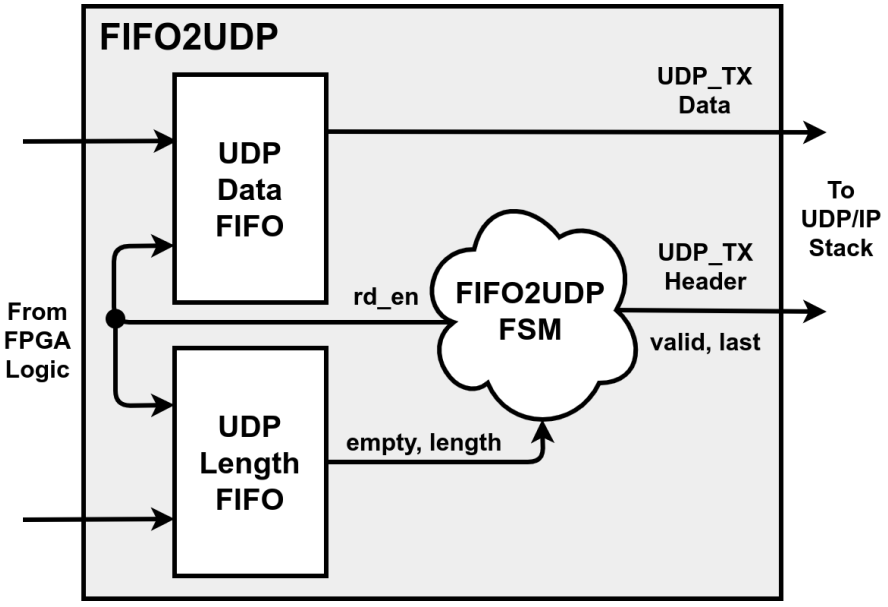


Figure 4.11: Block diagram of the *FIFO2UDP* module. Since the UDP payload length needs to be pre-defined before starting to forward the VMM data to the UDP/IP Stack, a dedicated buffer is instantiated, that stores this information alongside the data themselves. The FSM reads the packet length FIFO when it is not empty, reads the defined amount of words from the data FIFO, and issues the control (i.e. the *valid* and *last* signals) and UDP header signals accordingly.

4.3 Reference Clock and Test-Pulse Generator

In order to study the behavior of the VMM under a variety of conditions, the ASIC implements an internal *test-pulsing* circuit that injects artificial test-pulses into each of its channel inputs, independent of the implementation setup. This circuit mainly involves a 300 fF capacitor that is activated by an external stimulus. This is the *Test-Pulse Clock*, or simply *CKTP* signal, that can be issued by an external device that drives the VMM's associated pin. The signal is being fanned-out to all 64 channels, each one of which deploys the test-pulsing circuitry. For the VRS Front-End scheme, the CKTP signal is being driven by the FPGA logic, alongside other fundamental VMM signals, such as the 40.079 MHz reference clock (*Bunch-Crossing Clock*, or simply *CKBC*). To be precise, these signals are generated by the FPGA's *CKBC/CKTP Generator* (denoted as *Clocking & VMM Test-Pulse Block* in Figure 4.3). The reader may inspect a rough block diagram of the scheme in Figure 4.12.

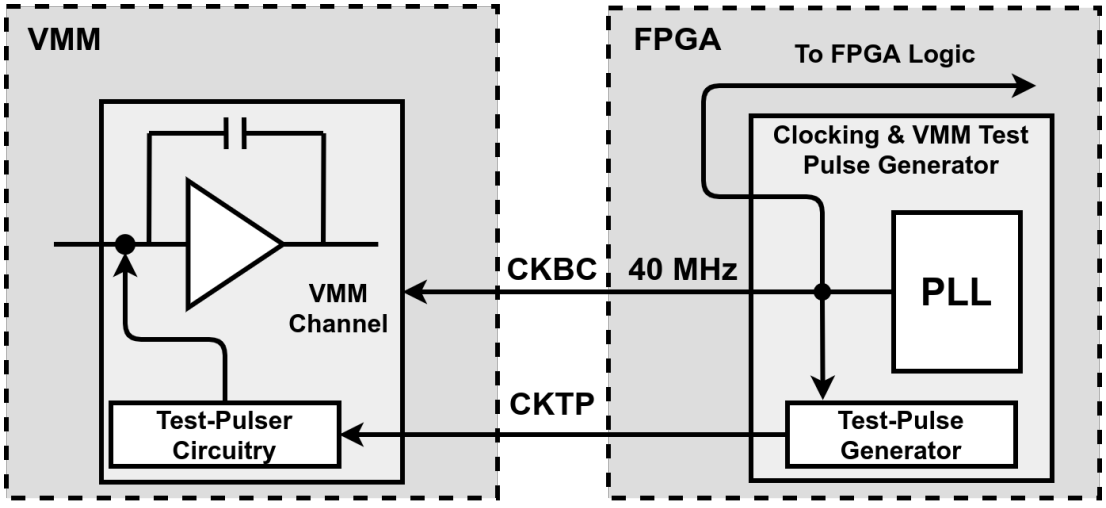


Figure 4.12: Block diagram of the test capacitor circuitry of each VMM channel and its relationship with the FPGA logic. It is reminded that the *CKTP* and *CKBC* signals are common for each of the 64 channels of the chip, but each test-pulse circuit is independent and every channel has its own associated test capacitor that is stimulated by the CKTP signal. The FPGA logic is clocked by the same ~ 40 MHz clock, alongside another, phase-locked 160 MHz clock. The Test-Pulse Generator logic finally, receives two clocks, the ~ 40 MHz one, and a phase-locked 500 MHz clock.

There are two reasons behind the need for having a mechanism to inject pulses upon the user's command into the channel: First of all, it allows for performing fundamental testing of the VMM on a per-channel basis without having to connect the chip to a detector medium, which can lead to a plethora of other issues that are not related to the chip's functionalities. Thus, after or prior to deploying an instance of the ASIC to the

field application, the test-pulsing scheme allows for defining any functional problems that may arise during the process of applying the chip in an experimental setup, from as early as its production, up to its deployment. Reading-out the chip under test-pulsing conditions can indicate if there are any problems in the analog circuitry of the channel, in its ADCs, or in its digital logic that derandomizes events before forwarding them to the external readout device.

The other reason that makes test-pulsing essential, is that it is the only way that allows for *calibrating* each individual channel of the chip. As it is known from previous Chapters of this dissertation, the VMM digitizes the information related to the pulse's amplitude and timing by employing dedicated ADCs in each individual channel. These ADCs however, do not have an identical response with respect to each other across different channels of a chip, let alone between different instances of the ASIC itself. Looking back at Section 2.2 for example, one can find information on how the VMM measures the phase relationship between an input pulse and its reference clock. The reference clock of the VMM in the final NSW experiment, will be the ATLAS system clock, called *Bunch-Crossing Clock* or *CKBC*[8], that has a frequency of 40.079 MHz. The timing measurement is being performed by a purely analog circuit that calculates the phase relationship between the peak of an input pulse and the next falling edge of the CKBC. This component produces a voltage level, that has an amplitude proportional to the "distance" between the pulse's peak and the falling edge of the CKBC (see also Figure 2.6). The *Time Detector Output* (TDO) ADC encodes this into 8 bits. The other functionality that must be calibrated is the pulse height measuring. The pulse's height is proportional to the charge deposited into the channel, and when the VMM is attached on a chamber, is correlated with the energy that was deposited by an incident particle into the detector medium. The pulse height analog voltage level is encoded by the *Peak Detector Output* (PDO) ADC into 10 bits. Thus, the only way to associate these ADC codes with actual physical quantities (i.e. *time* and *charge*[9]) is by performing a calibration read-out routine, that necessitates knowing a priori the charge that is being injected into the channel, alongside its phase relationship with the reference clock. An off-line analysis routine would then perform the necessary calculations from the calibration procedure and correlate each channel's ADC codes with the corresponding value. The only way to do this is, as the reader may have imagined by now, is by *test-pulsing* the VMM ASIC.

[8] As also mentioned earlier, the main feature of that clock is that its rising edges coincide with the protons colliding in the core of the detector. In the VRS Front-End Firmware, the FPGA emulates this scenario by generating a clock of that frequency and driving it into the ASIC.

[9] It is reminded that the VMM's specifications indicate that the accuracy of these measurements are ~ 250 ps and ~ 1 fC respectively.

Configuration-wise, a 64-bit VMM configuration bitmask, called *ST Register*, defines whether a channel's input is connected to its capacitor or not. The DAQ software has full control of the state of the 64-bit configuration string, as with the entire VMM's address space. Upon the user's directive, the combination of the channels that can be test-pulsed are parsed into the VMM via the FPGA and the SPI protocol, that previously received the associated UDP configuration packet generated by the software (see previous Section of the current Chapter). The next step that must be followed in order to retrieve test-pulse related data, is to stimulate the channel, and this is being done by issuing the *CKTP* signal via the FPGA, which implements the related circuitry (see Figure 4.12), alongside a set of related registers that control the CKBC frequency that is outputted to the VMM (40, 20 or 10 MHz)[10], the CKTP frequency (from several hundreds ns up to a few ms) and duty cycle, and its phase relationship with the CKBC (for the timing/TDO calibration). These registers belong to the subset of the *FPGA Configuration Registers*, controlled by the related sub-module of the *UDPdin_handler* (see Figure 4.8).

Upon the arrival of the CKTP's positive edge, and if the said channel's *ST Register* is high, a pulse is injected into the channel, which then processes the pulse and stores its digital data into its buffers, to-be-read-out by the FPGA at a later stage. As mentioned above, an FPGA configuration register controls the phase relationship between the CKTP and the CKBC, thus constituting a way to calibrate a channel's pulse timing response (or TDO). In order to calibrate the pulse amplitude response (or PDO) of a given channel, the user must also configure the VMM's Test-Pulse DAC, which controls the amplitude of the pulse by using the value of a 10-bit Digital-to-Analog Converter (DAC), that defines the capacitor's voltage[11]. In principle, the largest the *test-pulse DAC* value, the largest the *PDO ADC* value/response of all channels, with differences arising from variations in the analog[12] and ADC components of each channel. An example resulting plot of the PDO calibration procedure can be inspected in Figure 4.13.

Pulse Timing/TDO Calibration - The CKTP Skewing Logic

In order to perform the TDO calibration, more functionalities are required in the FPGA logic, since the VMM does not feature some kind of configurable mechanism that changes the phase relationship between the CKTP and CKBC signals. For this

[10] For the vast majority of implementations, the 40 MHz clocking option is used.

[11] Naturally, this DAC, which is common for every VMM channel, has to be calibrated as well. This is being done by the FPGA's XADC. More information on the said procedure can be found in the related Section of the current Chapter.

[12] For example, the baseline voltage varies from channel to channel, and this is calibrated by the FPGA's XADC as well (see Section 4.4).

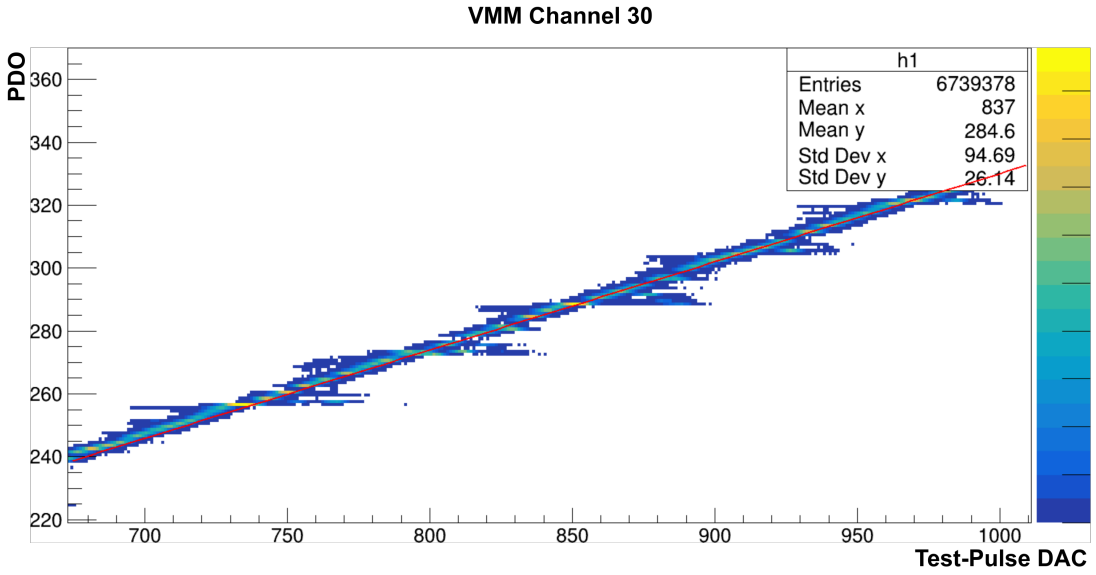


Figure 4.13: Plot depicting the relationship between the test-pulse DAC (x axis) codes, and the corresponding PDO ADC (y-axis) codes, for a given channel, under a PDO calibration test-pulse run/loop, where successive DAC values are being applied to the VMM's configuration address space. The data were extracted using the Level-0 Readout Mode (see Section 4.5). The test-pulse DAC must be calibrated separately by the FPGA's XADC to correlate the DAC's codes with actual charge.

reason, a circuit that performs the said functionality was designed in the FPGA, named *CKTP Skewing Logic (or Module)*, instantiated as a sub-component of the *Test-Pulse Generator*. This module accepts an FPGA configuration parameter, controllable by the software, named *CKTP Skew*, and according to the value of the said register, determines how much to shift the rising edge of the CKTP with respect to the rising edge of the CKBC. The general scheme can be viewed in Figure 4.14.

To delve more into the details of the procedure, the architecture of the FPGA's CKTP skewing module is depicted in Figure 4.15. The initial CKTP generator will produce a CKTP signal that its positive edge is aligned with the CKBC's. The CKTP output of the generator enters a *skewing register chain*, while it also drives one of the two inputs of a *BUFGMUX* FPGA primitive[13]. The output of this primitive drives *ODDR* instances, one for each VMM (eight in maximum, for multiple-VMM implementations), that forwards the signal to its associated VMM. This approach allows for a constant delay relationship between the CKBC and the CKTP, that does not vary between different design versions. The "unskewed" CKTP is selected from the *BUFGMUX* if the *CKTP_skew* signal equals to zero. However, if the latter signal yields a different value,

[13]This is a global clock buffer multiplexer, that allows for dynamically switching a clock.

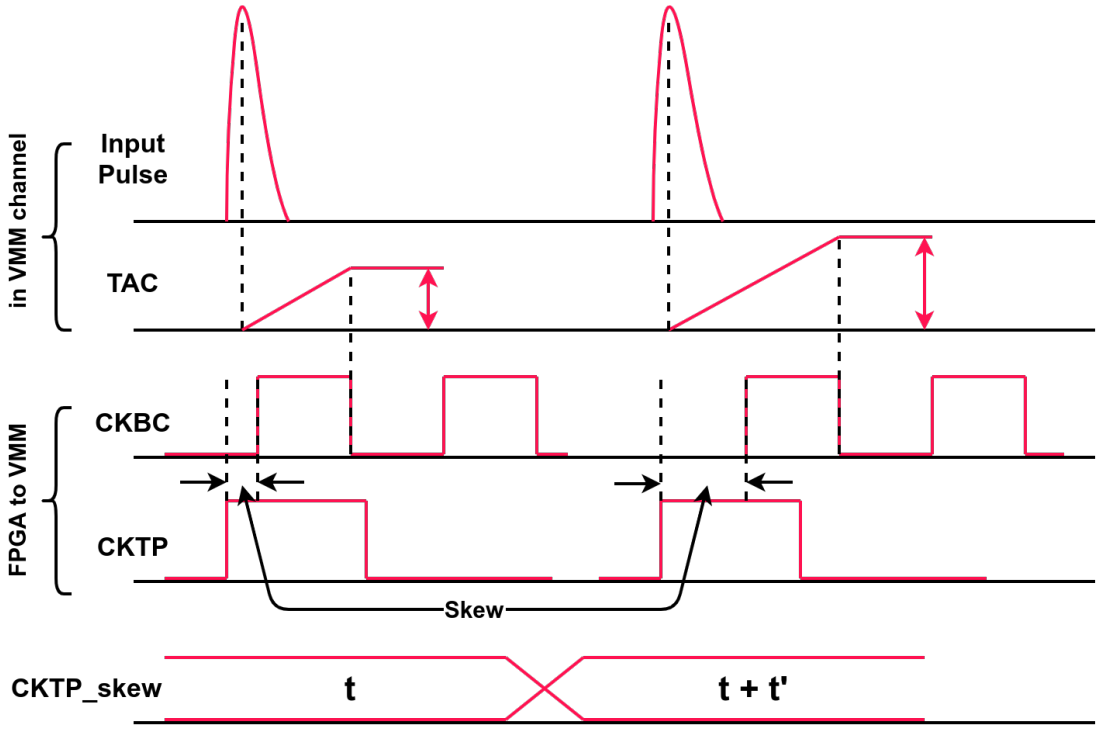


Figure 4.14: Timing diagram depicting the relationship between the VMM channel's response, the CKBC, and the CKTP and its skew are shown. At the top, the pulses generated in the channel are shown, alongside their associated Time-to-Amplitude (TAC) ramps. Each channel pulse is being triggered by the positive edge of a CKTP signal, issued by the FPGA. The CKBC dictates when the TAC ramp will stop, and essentially, the phase relationship (or *Skew*) between the CKBC and the CKTP defines the amplitude of the TAC ramp, and the subsequent TDO ADC code that will be stored in the VMM's buffers before being read-out by the FPGA. Finally, the *CKTP_skew* parameter is shown, which is a configurable FPGA register, defined by the readout software. Different values of that register will dynamically shift the CKTP, which consequently affects the VMM channel's timing response.

then the CKTP signal that has passed through the skewing register chain inside the *CKTP Skewing Logic*, is selected.

The skewing logic progressively adds more and more delay stages into the signal, thus changing its phase relationship with the CKBC. This is being achieved by implementing a *shift register with dynamic depth*, which is comprised by the aforementioned register chain and a multiplexer, that is being driven by different delay stages. Which of the stage is selected, is defined by the value of the multi-bit CKTP_skew signal. The register chain is being clocked by a 500 MHz clock, phase-related with the CKBC. Therefore, after each register, the CKTP will be delayed by 2 ns equal to the said clock's period.

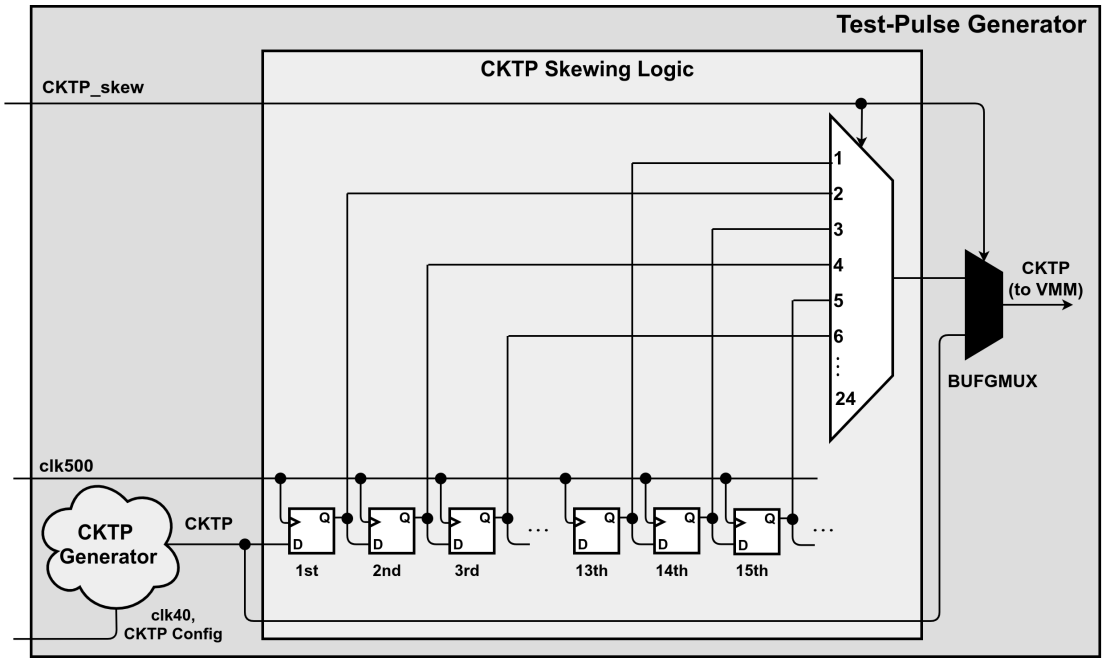


Figure 4.15: Block diagram of the CKTP Skewing Logic. For a description of the functionality, it is suggested to go through the main text of the current Section. The CKTP Generator issues the associated signal with configurable period and duty cycle, always edge-aligned with the 40 MHz reference clock that also drives the VMM's CKBC pin. Note also that there is a final register stage after the multiplexer and before the clock buffer that is not depicted here. This register is also clocked by the 500 MHz clock, and is deployed to alleviate any differences in routing delays between the multiplexer's inputs.

Given the fact that the period of the clock defining the skewing granularity is 2 ns, it would be easy to come to the conclusion that there can be *twelve* delay stages (or steps) in total. That is, if the skewing is equal to zero, the "raw" CKTP would be selected, a skewing of one count would produce a 2 ns-delayed CKTP, a skewing of two counts would produce a 4 ns-delayed CKTP etc. The twelfth skewing step would delay the CKTP by 24 ns, and that would be the end of the procedure, since the period of the reference clock performing the measurement in the VMM, has a period of 25 ns. However, the observant reader may have noticed by now that in Figure 4.15, there are *more* than only twelve registers, and that the multiplexer has twenty-four inputs instead of just twelve. This is because in reality, the signal is not shifted with steps of two nanoseconds, but in steps of *one*.

This is achieved by adding more registers into the chain. If one takes into account another register stage after the initially-thought final 12^{th} stage, then that 13^{th} register would skew the CKTP by 26 ns. This of course, is a useful result, since $T_{CKBC} = 25$ ns. Essentially, that last register has skewed the CKTP by *one* nanosecond, but rolled-over

to the next period of the clock. Therefore, a 14th register shifts the CKTP by *three* nanoseconds, the next by *five*, and so on. The final design includes twenty-four registers that are connected to the multiplexer's inputs in such a way (note that in Figure 4.15 they are not connected "sequentially" to the multiplexer) as to allow for skewing of the CKTP with 1 ns granularity. An example resulting plot of the TDO calibration procedure can be inspected in Figure 4.16.

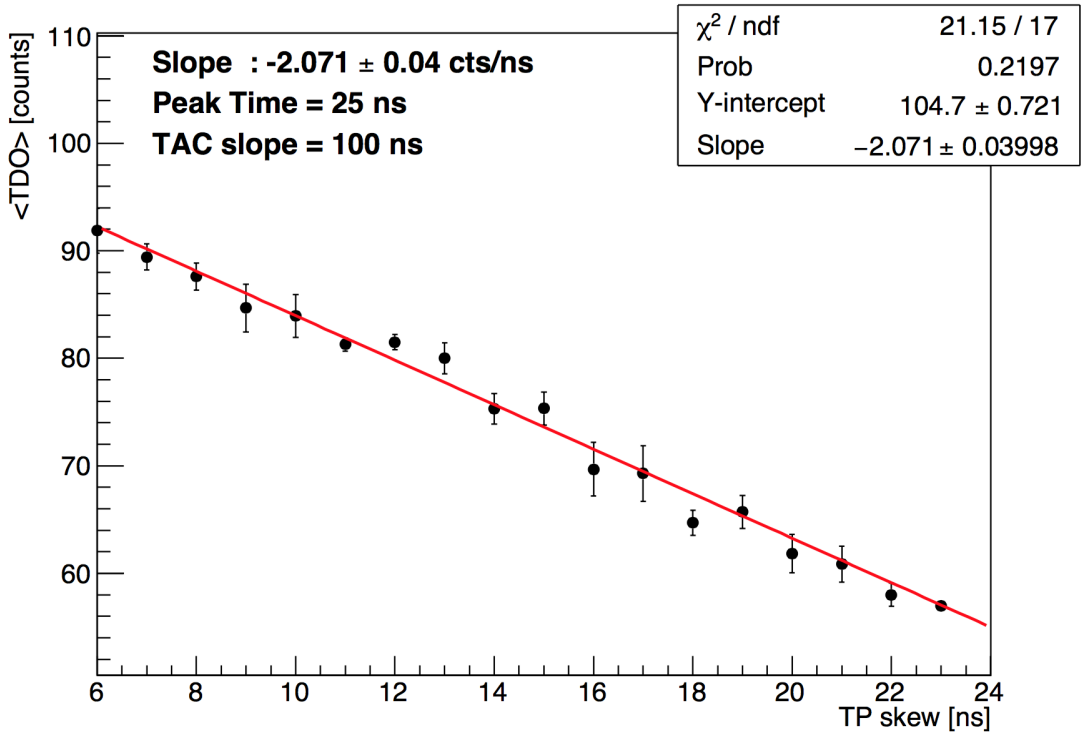


Figure 4.16: Plot depicting the relationship between the CKTP's phase difference with the CKBC (x axis) in nanoseconds, and the corresponding TDO ADC (y-axis) codes, for a given channel, under a TDO calibration test-pulse run/loop, where successive CKTP skew values were applied to the FPGA's configuration address space. The data were extracted using the Level-0 Readout Mode (see Section 4.5).

4.4 The XADC Module

Xilinx® 7-Series FPGAs deploy an internal 12-bit ADC, named *XADC*, that allows for sampling of external-to-the-FPGA analog inputs [13]. This module can be controlled and read-out using regular user logic and FPGA fabric interconnect. Its analog voltage input ranges from $+0$ to $+1$ V in unipolar mode, and it can sample at a rate of one Mega Sample Per Second (1 MSPS). For the current application, the sampling mode

(or *Timing Mode*), is configured to be *Event Driven*; that is, the user logic defines when an analog input must be digitized, and the XADC provides the measurement result to the supporting logic after 1 μs.

The *XADC Module* that is deployed in the VRS Front-End FPGA firmware, is essentially a wrapper[14] for the XADC primitive, that interfaces with the rest of the logic via a simple interface. To be specific, an FSM implemented in the wrapper, handles commands and configuration parameters from other hierarchy levels, and mediates any necessary data to the XADC primitive via the so-called *Dynamic Reconfiguration Port (DRP)*. The primitive itself, interfaces with the VMM's analog outputs via two differential pairs, called PDO and TDO. The default output of these ports are the voltage levels of the Peak Detector and Time-to-Amplitude Converter, that also drive the inputs of the VMM's internal ADCs. These voltages are used in the *Two-Phase Analog Readout Mode*, that is described in Chapter 5. The VMM however, deploys an internal multiplexer that routes other types of analog voltages to its PDO output as well. The state of the multiplexer is controlled by the VMM's configurable registers, and depending on the user's needs, may switch the PDO output to the VMM's internal temperature sensor, to the baseline voltage of one of its 64 channels, or to the test-pulse amplitude DAC, to name a few. Hence, the user can control the voltage input source to the XADC via the VMM configuration, and can also configure the XADC to perform a specific amount of samples, and with a certain delay between them via the FPGA configuration. These last two parameters affect the supporting FSM's functionality that is deployed inside the XADC module. Upon the user's directive, the conversion loop may start, and the FSM initiates that by issuing the *CONVST* signal to the XADC, which then proceeds to sample the input voltage level driven by the VMM. ~ 1 μs after the command that starts the sampling, the XADC issues the *EOC* (End Of Conversion) signal. The FSM uses that signal to know when to extract the 12-bit sample that is stored inside the XADC's address space, by using the DRP port. The read data are then stored by the FSM, and if another sample must be taken, the procedure is run again. The registered samples are forwarded to the UDP interface via the *FIFO2UDP* module, which buffers the data. After all samples have been taken, the XADC module signals the FIFO2UDP accordingly, by writing the UDP packet length word into the associated buffer, which initiates the UDP transmission. The general connectivity can be inspected in Figure 4.17.

The implementation of the described logic is imperative, in order to debug, calibrate and read-out (via the Two-Phase Analog Readout Mode - see Section 5.4) the VMM. After connecting the ASIC to a detector medium, one of the first steps to evaluate the noise

[14] The only component written in Verilog HDL.

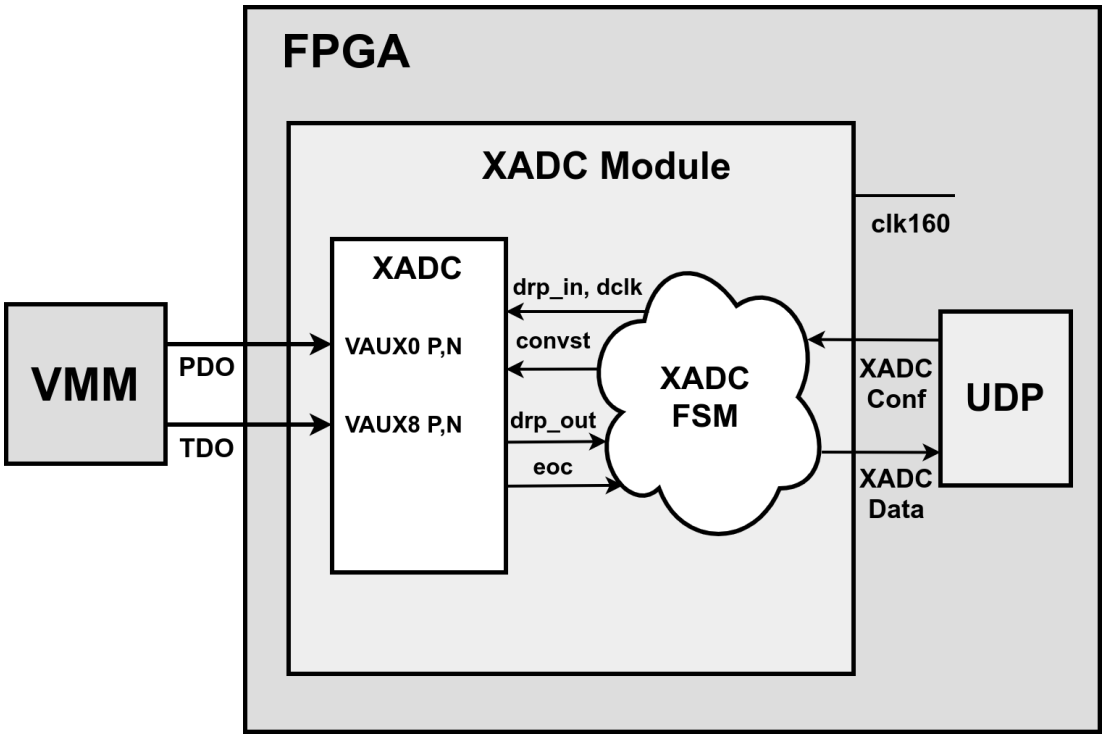


Figure 4.17: Representation of the XADC's connectivity with the VMM, its supporting FSM inside the XADC Module, and its relationship with the FPGA's UDP interface. The VMM drives the two auxiliary analog inputs of the XADC via its *PDO* and *TDO* output ports. These ports can be configured to output a variety of analog voltages. The XADC's DRP interface is controlled by its supporting FSM, that receives configuration parameters by the UDP interface. A conversion may start at any time by the FSM upon the assertion of the *CONVST* signal, and the XADC primitive informs the FSM that the conversion is done by issuing the *EOC* signal. The FSM temporarily buffers the samples, and forwards them to the UDP interface, so that the DAQ software can evaluate them at a later stage.

that a channel is subjected to, is to measure the *Root Mean Square (RMS)* of its baseline voltage. In order to do that, the user configures the VMM to successively output all of its channel's baselines to the XADC, which performs sampling on all voltages, before forwarding them to the DAQ software. The user may then derive the *Equivalent Noise Charge (ENC)* of a channel, and its *Signal to Noise Ratio (SNR)*. The reader may refer to Figure 4.18 for a plot depicting the baseline voltage variation across all channels of a VMM.

In addition, the user can use the XADC to calibrate the two Digital to Analog Converters of the ASIC. One of them is associated with the test-pulser amplitude. As mentioned above, the first step towards calibrating the PDO ADC response of a channel, is to first calibrate the test-pulse DAC. Through the simple formula $Q = CV$, where $C =$

300 fF, the charge that is injected into the channel as a function of the voltage is given. This voltage, is the voltage that is being set by the DAC, and the same voltage can be outputted to the XADC if the VMM is configured accordingly. By applying successive DAC values to the VMM, the corresponding voltage can be sampled by the FPGA, thus deriving the valuable function that associates test-pulse DAC values with charge, and converting the scale of the x-axis of Figure 4.13 from counts, to meaningful fC. The other DAC that must be calibrated to ensure proper operation of the ASIC, is the threshold DAC, that sets the channel thresholds in mV. If the user knows the ENC that a channel is subjected to, together with the actual threshold value that is applied for each threshold DAC count, a precise threshold can be set on a per-channel basis, essentially eliminating noise from the readout process.

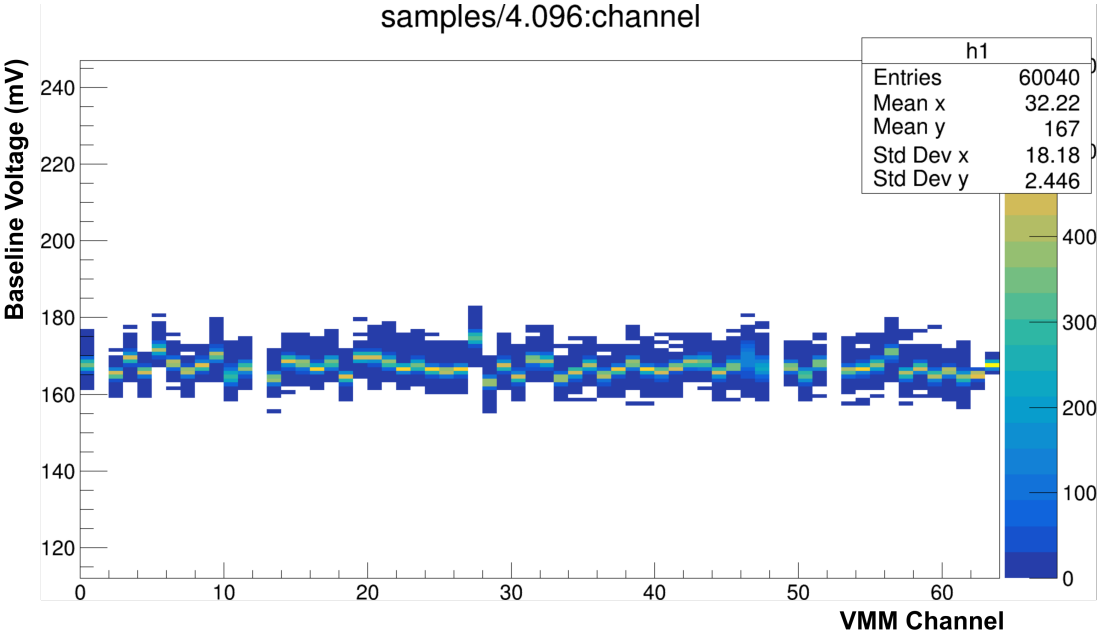


Figure 4.18: Baseline voltage variation across all 64 channels of a VMM. Measurements taken using the FPGA's XADC module.

4.5 VMM Readout Path

As mentioned earlier in this Chapter, the VRS Front-End Firmware, supports all three readout modes of the VMM. The main module implementing the related logic is the *VMM Readout Wrapper*, that contains three distinct modules, each dedicated to one readout method. For more information regarding the components of the FPGA logic that take part in the readout of the VMM, the reader is referred to Chapter 5 which is dedicated in describing the aforementioned modules.

When reading-out the ASIC, the firmware's logic first expects a *Trigger*, either generated internally or driven into the FPGA by an external source. Upon the reception of that signal, the logic forwards it to the ASIC, extracts the digital or analog data from the VMM, forms a packet with a specific format, and forwards that packet via the UDP interface to the DAQ software. The packet that is formed and forwarded to the software has three different formats, that depend on the readout mode that has been implemented. These different packet formats though, have one thing in common; they are all constituted by: a *Header*, the *Payload* that contains the VMM data, and a *Trailer*. After receiving the said UDP packet, the software analyzes the data, and creates histograms that contain all necessary information (e.g. pulse charge (PDO) or pulse timing (TDO) information of a given event for a particular channel), in order for the user to analyze them so that they can characterize the VMM's performance, in conjunction with the detector medium, if present. A plot depicting the PDO distribution across all 64 channels of a VMM, for one test-pulser DAC value, is provided in Figure 4.19.

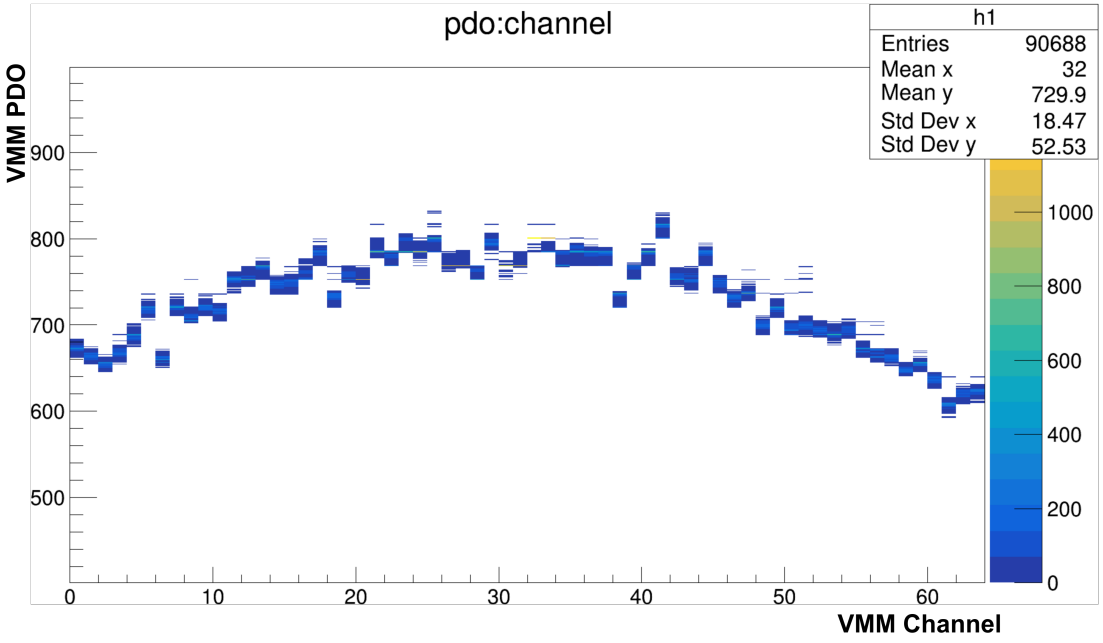


Figure 4.19: Plot depicting the PDO distribution across all 64 channels of a VMM, for one test-pulser DAC value after a test-pulser data-taking session.

Conclusions

In this Chapter, the architecture and functionalities of the VMM Readout System Front-End FPGA Firmware was described. Deployed in an FPGA that is on the same board that the VMM is on, the firmware is responsible for configuring, calibrating, and

reading-out the ASIC. Emphasizing on modularity and adaptability, the firmware can be implemented in several board versions and FPGA packages. It uses UDP-over-1 Gb/s Ethernet to communicate with a DAQ software host that collects the VMM readout data, as extracted by the FPGA. The said software employs a GUI that allows the user to configure both the VMM and the FPGA. The VRS Front-End firmware and its accompanying software (named VERSO) can be used together to extract detector data using the VMM as a front-end ASIC, or to calibrate that very same chip on the bench via the VMM's internal test-pulser. The flexibility and reliability of the scheme resulted in it being used to perform the mass-testing of the VMM, while rudimentary modifications to the design led to the development of the larger-scale VMM Readout System, comprised of many front-ends, which will be described later in this dissertation.

Chapter 5

Reading-Out the VMM

This Chapter focuses on the way data are extracted from the VMM, using the associated front-end FPGA firmware described in Chapter 4. As mentioned earlier in this dissertation, the VMM features three readout modes, and the firmware supports them all adequately. The reader is suggested to refer to Section 2.2 for more information on how the VMM processes input pulses before storing them into its buffers. What follows in this Chapter, is a brief overview of the FPGA readout scheme; that is, a description of the FPGA logic that applies the VMM data extraction mechanisms will be given, before closing with a depiction of a readout system that concentrates on minimizing digital noise that couples to the ASIC's analog circuitry, when operating the VMM in the analog readout mode.

5.1 The VMM Readout Wrapper and its Supporting Modules

The readout process and the logic blocks involved for the two digital data extraction methods (i.e. *Continuous* and *Level-0 Readout*) will now be analyzed. In Figure 4.3, the general block diagram of the VMM Readout Front-End FPGA firmware was given. The module that implements the readout logic of the VMM, is the *VMM Readout Wrapper*. Three distinct cores, each dedicated in reading-out the VMM in a specific mode, are contained inside the wrapper. Three *'generic'* VHDL variables control which of these cores is deployed (therefore, a firmware version can only support one readout option at a time), prior to generating the design's netlist. In principle, each of these cores handle the low-level signals that need to be driven to the VMM's inputs, in order to extract the data from it. The VMM sends out the pulse-related information via two data lines

(*DATA0* and *DATA1*, denoted as *D1* and *D2* in Figure 2.3), and the FPGA deserializes the bits from these two VMM outputs. The FPGA then buffers them temporarily, in FIFOs that are implemented within the readout core that operates. These data are eventually driven to the UDP interface via the *FIFO2UDP* module. Alongside the readout wrapper, there are two other modules that play an important role in the readout scheme. These are the *Trigger Module*, and the *Packet Formation* component. The Trigger Module is responsible for prompting a readout cycle, by receiving a *trigger* input signal, and converting it to the *newCycle* signal, that is sent to Packet Formation. The latter module waits for the *newCycle* signal to go high in order to initiate the readout procedure by signaling the readout cores in the readout wrapper accordingly, which lie one hierarchical level below the Packet Formation component. As data are extracted by the ASIC, Packet Formation inhibits any inbound trigger signals by issuing a *busy* signal to the triggering logic. Their architectural relationship is illustrated in Figure 5.1, and their logic is briefly described below.

The Trigger Module can accept a signal from two sources: *Internally*, from the CKTP generator (see Chapter 4), or *externally* from another trigger signal source. The FPGA can only be triggered one way or the other for a given run, and this is configurable via the software through the UDP-accessible FPGA's configuration registers. When operating under test-pulse conditions, the CKTP issues the corresponding signal to the VMM, and subsequently signals the FPGA's triggering mechanism, so that it can extract the test-pulse related data from the VMM. When the VMM is used to read-out a detector medium, an independent triggering apparatus is used, and usually this comes in the form of a set of scintillators and photomultipliers. An incident particle stimulates them, and they produce a logical pulse with minimal latency, essentially signaling that a particle has gone through the detector, and consequently, that an associated set of pulses have been processed by the VMM's channels, thus prompting the FPGA to read it out.

The Packet Formation resides one hierarchical level above the readout wrapper, and supervises its functionality. Upon reception of the *newCycle* signal from the triggering logic of the FPGA, Packet Formation creates an *Event Header*, which is prepended to the VMM data packet that will eventually be sent to the software via UDP. The structure of this header and the data packet comes in different forms, depending on the readout mode that the FPGA is operating in. The main feature of the event header though, is that it contains the so-called *Trigger/Event Counter* value. Upon reception of the logical signal that initiates a readout round, Packet Formation increments this counter by one, and attaches it is as metadata (i.e. in a header) in the VMM information that are extracted by the readout cores. As the readout component extracts data from the VMM, Packet Formation awaits for it to finalize the procedure. Once done, Packet Formation takes care of relaying the data from the readout component's local buffer,

to the FIFO2UDP data buffer, which are connected by a 16-bit bus. For every word propagated from one FIFO to the other, Packet Formation increments the event's UDP packet length counter. After the readout component's FIFO has been emptied, the byte-size counter word is written to the associated buffer of FIFO2UDP, which initiates the UDP/Ethernet transmission. If instantiated on a board with multiple VMMs, Packet Formation acts as a demultiplexer, that grants access to the numerous readout core buffers, and calculates the packet size for each core. For each VMM on the board, a separate UDP packet is being sent to the software. In order to group these packets together in a single event (see Appendix D), the software uses the event header trigger counter information to know the trigger that the events are associated with.

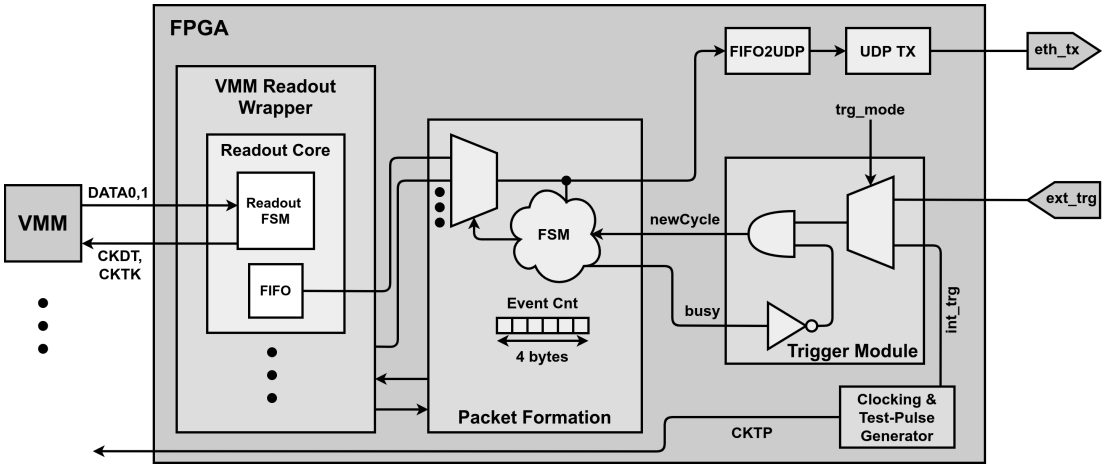


Figure 5.1: A more detailed description of the firmware modules that focus on the VMM readout. The *VMM Readout Wrapper* instantiates a *Readout Core* that is compatible with one of the two digital readout modes of the VMM. An FSM handles the low-level signals, and a FIFO buffers the data extracted from the VMM. The *Packet Formation* FSM waits for the *newCycle* signal to initiate a readout cycle. Upon the assertion of this signal, the *Event Counter* gets incremented by one, and a header containing status flags plus the value of the counter is sent to the *FIFO2UDP* module. Then, the Packet Formation FSM waits for a handshaking signal from the readout FSM which indicates the data extraction is complete. The Packet Formation will then read the corresponding FIFO, and forward its contents to the FIFO2UDP component. If more than one readout logic blocks have been implemented (one for each VMM on the board), then the Packet Formation switches its data bus demultiplexer to the next FIFO that contains data from another VMM. Throughout this process, a *busy* signal is high, that inhibits any triggers originating from the *Trigger Module*. The latter component relays trigger signals, either originating from an external source, or internally from the *CKTP* assertion logic, in the case of test-pulse runs.

5.2 Continuous Readout Mode

In this section, the FPGA's *Continuous Readout Mode* implementation will be described. This readout method was first implemented in VMM2, and remained as an option in all future versions of the chip (that is, VMM3 and VMM3a). As mentioned in Sections 2.2 and 4.5, after processing an input pulse via its analog circuitry, the VMM channel digitizes the information and timestamps the event with the coarse timing measurement, or BCID, and buffers all data in a 4-event-deep FIFO. This FIFO is 38 bits in width. At each of the four positions, it stores the full event data, which constitute of two flags, the coarse timing measurement (12 bits) or BCID, the fine timing measurement (8 bits) or TDO, the fine pulse amplitude measurement (10 bits) or PDO, and the channel address (6 bits). In order for the VMM to present these data to an external readout device for deserialization, the *CKTK* signal (or simply, *token*) must be issued to one of its pins. This is being carried out by the continuous readout mode FPGA logic core, which is triggered to do so after being activated by its supervising module.

Upon issuing of the *newCycle* signal by the triggering logic to the *Packet Formation* component, the latter activates the continuous readout core after a fixed latency. The main reason behind the application of this latency, is to give the necessary amount of time for the VMM's analog and digitizing logic to fully process the input pulse. This is especially critical for the case that the VMM is being used to read-out an actual detector. In this type of situation, the trigger signal may arrive to the FPGA before the VMM has managed to process all of its events, because of detector-related physical phenomena that affect the time the particle-related charge is formed (i.e. *drift time*, see Section 2.1). After activating the FSM of the readout logic, Packet Formation builds the event header, writes it into the FIFO2UDP's data buffer, and waits to receive the *VmmWordReady* signal from the readout FSM.

Upon its activation, the readout FSM issues the *CKTK* signal, and at the next clock cycle, probes the *DATA0* line for a flag indicating that the VMM yields data in one of its channels. If that signal is high, the FSM will proceed to issue 19 *CKDT* strobes, and shift-in the 38 bits of the event word out of the two data lines into a local shift register, before padding it up to 64 bits and buffering it into its local FIFO. The FSM will then issue the token signal again, and probe the *DATA0* line for the flag. In essence, each time the token is issued, the VMM cycles through the channels that yield data, and makes the channel event word available through its two data lines. The FPGA will keep issuing strobes to the VMM until it does not detect a flag in the data line, which means that the 4-event-deep FIFOs across all 64 channels of the chip are empty. A detailed timing diagram of the procedure is shown in Figure 5.2.

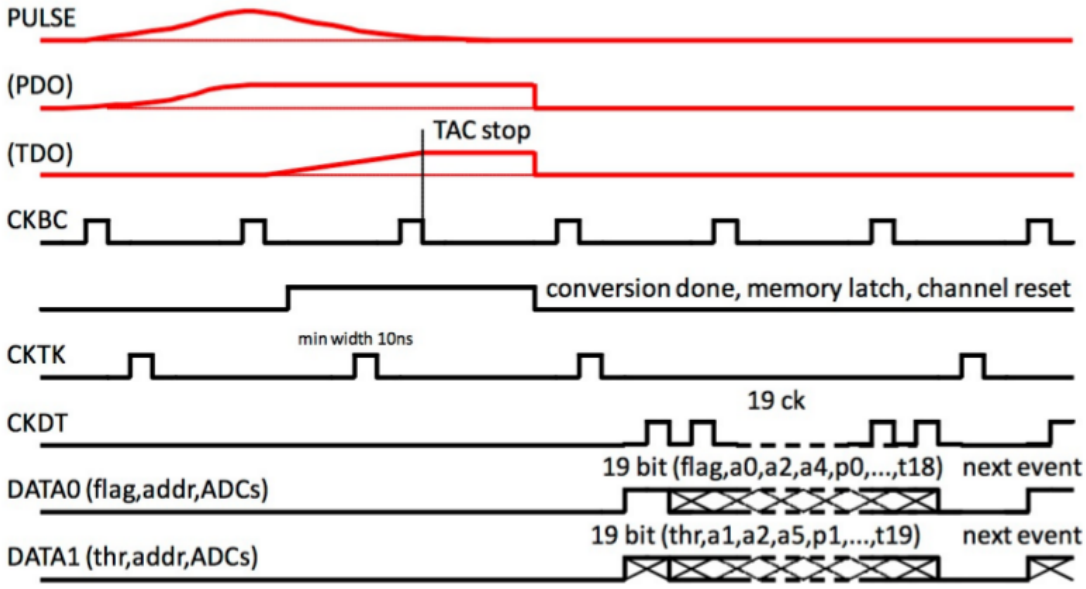


Figure 5.2: Timing diagram of the Continuous Readout mode [50].

After the readout FSM stops encountering a flag in the *DATA0* line, the *VmmWordReady* signal is issued, and the Packet Formation module will progressively read all the data from the readout core's FIFO, and forward each 64-bit word to the FIFO2UDP data buffer. These data are eventually transmitted to the software via the Ethernet interface, as UDP payload. The packet structure is depicted in Figure 5.3.

5.2.1 Eliminating Noise - CKBC Strobing Method

The readout method depicted is a simple way to extract data from the VMM. Its downside though is the fact that the VMM is continuously streaming data into its buffers, and the FPGA will extract *all* of these data upon the reception of a trigger. To be precise, the caveat is that given the implementation that was described, there is no way to distinct which of the data that are buffered in the channels are correlated with the trigger and which are not. On one hand, this is not an issue when operating the VMM on the bench under test-pulses, where noise interference is negligible, given the setup and the fact that the user has control of the injected pulse amplitude. However, when reading-out the VMM while it is connected to a detector, problems may arise. Consider a hypothetical scenario where a noise burst prompts some of the ASIC's channels to process and buffer the noise-related pulses. These may occupy some positions in the channel FIFOs. Then, a particle traverses the chamber, thus creating pulses that get processed by the channels. This particle will also generate a trigger, and initiate a readout cycle in the FPGA, that will "blindly" extract a series of events from the VMM,

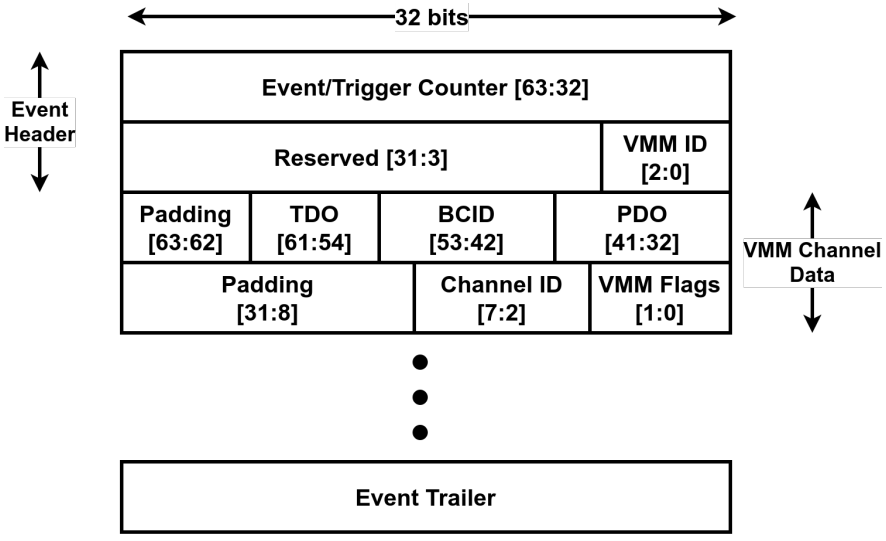


Figure 5.3: Structure of the data packet that the FPGA sends to the DAQ software when operating in Continuous Readout mode. The *Event Header* is being generated in the FPGA (by the *Packet Formation* component), while the rest of the data are being sent from the VMM to the FPGA as they are (apart from the padding).

some useful, and some not.

One way to work around this, is by using a *local* BCID counter in the FPGA's logic, that is in sync with the associated counter in the VMM. This approach would have to be implemented as follows: when the data acquisition process starts, and before the issuing of any trigger, the FPGA will have to assert a so-called *soft-reset* to the VMM[1], which will force its BCID counter to go to zero, and at the same time, reset its local BCID counter as well. Hence, the two counters will become synchronized. Upon the reception of a trigger, the FPGA would have to timestamp that trigger with its local BCID counter value, and attach it to the event's header (see Figure 5.3). Then, the software would be able to correlate the channel hit BCID values with the equivalent trigger timestamp. A similar scheme is actually implemented inside the VMM's digital logic, when operating in the Level-0 Readout Mode (see Chapter 2) as will become evident in the following Section.

Another approach, somewhat more complicated in its implementation, but with an extra benefit that will become evident in Chapter 6, is the so-called *CKBC Strobing Method*. This readout scheme is a subset of the continuous readout mode (that is, the related readout core is implemented in the FPGA, and the VMM is configured accordingly), but with one major difference to the regular readout described in the Section above: the *CKBC* reference clock driven to the VMM is for the most part held *low*. Hence, it is not

[1] This is achieved by toggling the *ENA* signal that is being driven to the VMM by the FPGA.

a *free-running* clock, but rather a *strobe* signal, that is activated only upon the reception of a *Trigger* signal by the FPGA. This methodology takes advantage of the *self-reset* feature of the VMM. In absence of a toggling CKBC signal, the Time-to-Amplitude Converter (TAC) never halts its ramping-up (see Figure 2.6 for a depiction of the TAC's functionality), and reaches a *saturation threshold* which resets the channel to its original state. This threshold equals to the voltage that corresponds to the maximum ramp duration. For example, if a given VMM is configured with a TAC ramp duration of 100 ns, and a pulse is processed by one of its channels and its peak is found, but no *CKBC* is fed to the VMM, then the TAC will ramp-up for 100 ns, saturate, and reset the channel, deeming it ready to accept another pulse. This use-case can be examined in Figure 5.4.

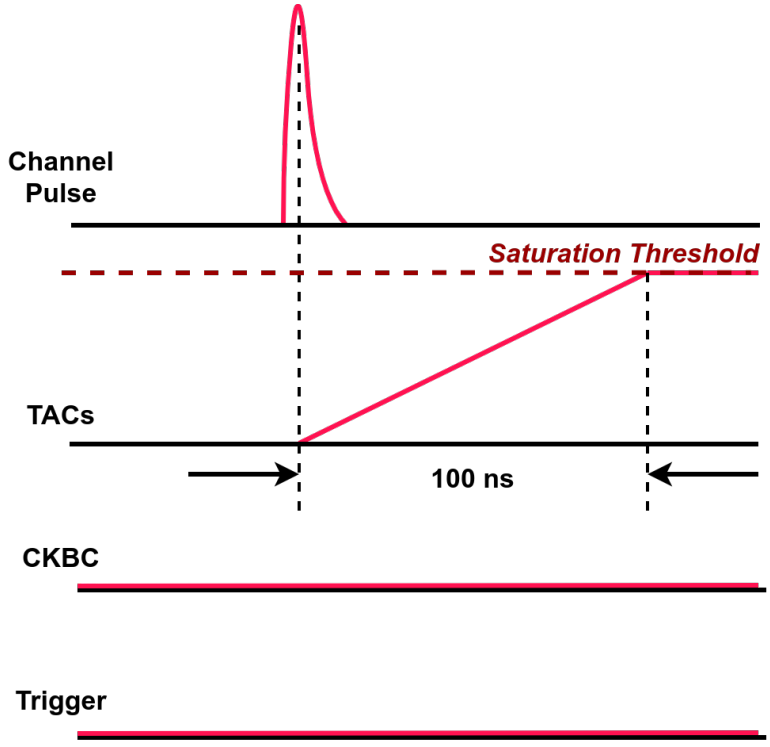


Figure 5.4: Depiction of the TAC's saturation, given the absence of a *CKBC* signal. If 100 ns (the hypothetical TAC ramp) pass after the activation of the ramp, but no *CKBC* pulse is driven to the VMM, then the channel self-resets because of the saturation of the TAC. The *Trigger* signal is kept low in this Figure as a contrast to Figure 5.5.

Consider again the hypothetical scenario depicted in the beginning of the current Subsection. If a noise burst stimulates the VMM's channels, then no *Trigger* signal would be driven to the FPGA. Hence, no *CKBC* strobes will be sent to the VMM so that it can process the pulses. Its FIFOs will therefore never store irrelevant data. If however an actual particle traverses the detector and the external triggering apparatus,

pulses will be generated in the VMM's channels, alongside a related *Trigger* that will initiate a readout cycle. In the strobing readout methodology though, a trigger does not only activate the readout core, but it also releases a configurable amount of *CKBC strobes* after a configurable *latency*. These strobes will halt the TAC ramps across the channels, while the last strobe will serve as a write-enable signal, that registers the ADC words (PDO and TDO) into the VMM's FIFO. After sending that last pulse, the continuous readout core logic will extract the data from the FIFOs under the exact same procedure depicted in this Section. This method allows for a "clean" readout, free of noisy events that are not correlated with a trigger. The scheme can be examined in Figure 5.5.

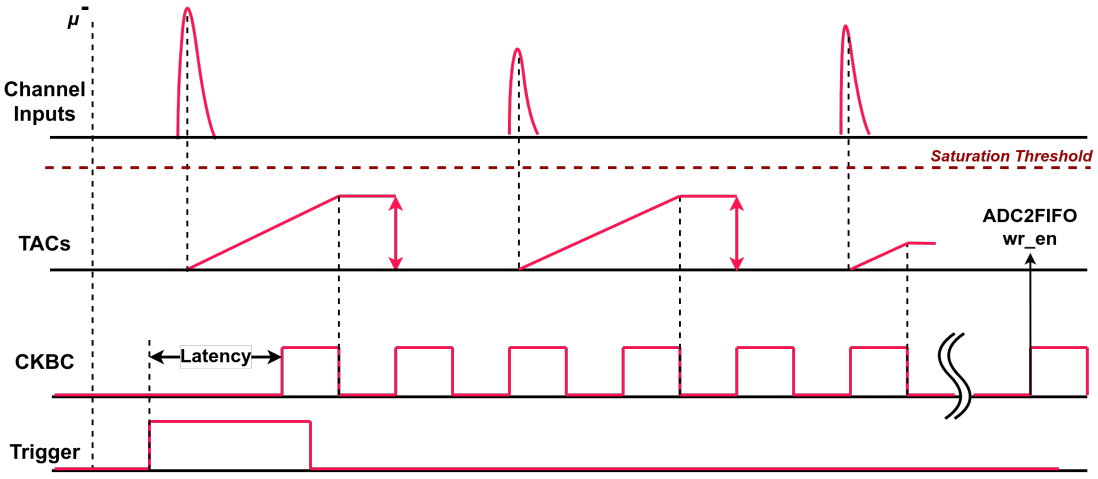


Figure 5.5: Depiction of the CKBC strobing readout methodology, when a muon-related event generates pulses that must be read-out. After the muon (denoted as μ^-) crosses the setup, a *Trigger* is generated, and pulses are created across several VMM channels, because of the cluster of charge that the detector generates. Note that the pulses depicted correspond to different VMM channels. Their associated TACs will start ramping-up, and meanwhile, the FPGA releases *CKBC* pulses, because it registered the trigger input. Different pulses halt different channel TACs, and the last strobe parses the digitized words into the VMM FIFOs. After that, the FPGA reads-out the data from all channels.

5.3 Level-0 Readout Mode

The *Level-0 (or L0) Readout Mode* is the data extraction method that is going to be used in ATLAS, since it allows for selective readout of specific data that are stored in the VMM's deep L0 buffers, through the L0 selection logic, implemented in the VMM's digital part of its architecture (see Subsection 2.2.1 and Figure 2.8). This functionality is highly desirable, since the Micromegas and the sTGC chambers that the VMM read-out, do not just detect muons originating from the interaction point. Cosmic muons and

other particles that are generated by parasitic reactions that occur between particles and the detector's materials, can also inject pulses to the NSW detectors. These, alongside events related to electronic noise, are not useful for physics and should be discarded. The VMM's L0 selection does exactly that, thus facilitating the analysis of physics data, while also reducing the total data throughput.

When an input pulse crosses a VMM channel's user-set threshold, it will immediately get timestamped with the 12-bit BCID value, originating from the VMM's internal Gray code counter that increments at each rising edge of the reference clock (or *CKBC*), every $\sim$ 25 ns. This event will eventually be forwarded to the 4-event-deep FIFO. When operating under *Continuous Mode* conditions, the channel does not perform any additional data handling regarding this event. In the L0 mode however, the event data are transferred from the FIFO to the *L0 buffers*. In order for that event to be read-out by the external device, it must be *matched*, time-wise, with the trigger. This is the so-called *Level-0 (or L0) Trigger*, driven to the VMM via the *CKTK* pin.

Upon reception of the *L0* trigger, the VMM timestamps it with a BCID, in a similar manner with the actual events. In essence, after the timestamp, the trigger is being "offset to the past", by a configurable amount of bunch-crossings (i.e. steps of $\sim$ 25 ns). The offset trigger timestamp will be checked against the events that have been stored in the L0 buffers, and if a match is found within a configurable window[2], these data are being selected for readout, and are forwarded out to the external readout device. In order for this to work, the *latency* between the actual event, and the trigger that is associated with it, must always be *fixed*. This is true for the ATLAS trigger and data acquisition system, where the triggers that are being generated from the events that are to be read-out, arrive to the front-end electronics after a pre-defined amount of time that does not fluctuate. This amount of time, or *Level-0 Latency*, is parsed into the VMM's configuration registers prior to the run, in order for the scheme to operate correctly. In the FPGA, the same approach is followed, and in Chapter 6, the implementation while under actual detector readout conditions will be described. The VMM's test-pulse readout methodology by the FPGA on the other hand, is described here. In this scheme, the *CKTP* signal is first being issued by the FPGA. The VMM channels that their injection capacitor have been activated, generate a pulse upon the reception of the *CKTP*. The events related to the test-pulse will be timestamped and stored to the L0 buffers. The FPGA, after issuing the *CKTP*, will also issue the *L0* trigger, after a *specific* amount of time. This delay between the assertion of those two signals, must be *equal* to the offset that the VMM applies to the received trigger, otherwise it will not select the data that are related to the test-pulse. If the two delays are the same,

[2]Maximum eight bunch-crossings in width, so $\sim$ 200 ns.

the data are matched with the trigger, and are sent out to the FPGA that eventually forwards them to the software for further analysis. A timing diagram of the scheme can be examined in Figure 5.6.

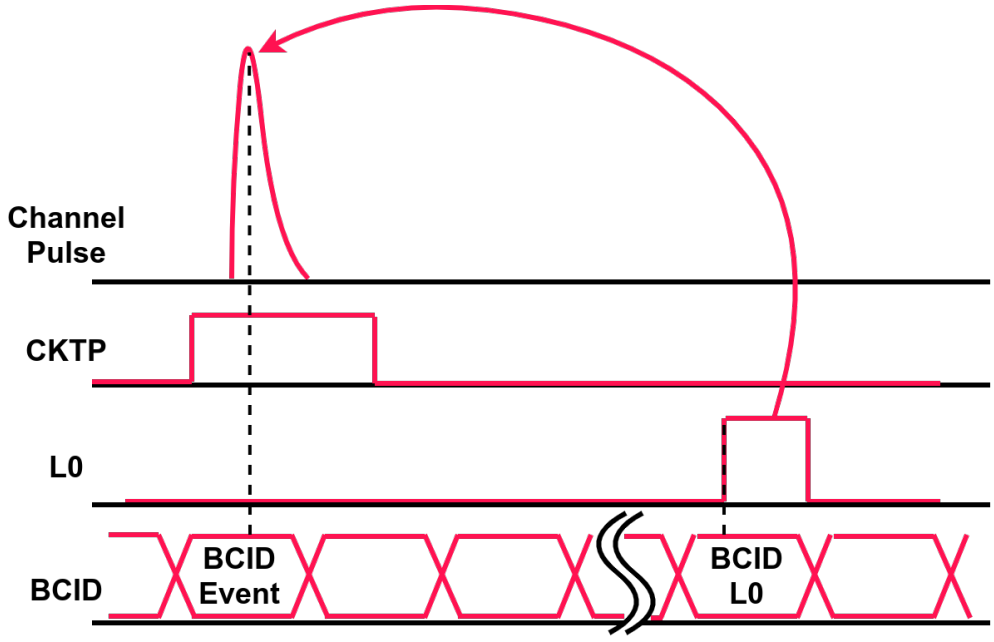


Figure 5.6: Timing diagram of the L0 Readout logic when operating under test-pulse conditions. The FPGA first issues the *CKTP* which generates a pulse in the channel(s). The channel timestamps the pulse *(BCID Event)* and stores its data in the L0 buffers. After some bunch-crossings, or *CKBC* ticks (note that the *BCID* increments at each *CKBC* edge which is not depicted here), the FPGA issues the *L0 Trigger*. The VMM timestamps it as well *(BCID L0)*, and then essentially "looks back in time", by a configurable amount of bunch-crossings. If that amount of time equals to the delay between the event and the associated trigger, then the VMM will send out only the data that are related with the pulse.

To minimize readout latency and dead-time, the data are being forwarded from the VMM to the FPGA via a continuous *8b10b* encoded stream (see Appendix B). The FPGA drives the 160 MHz data clock into the *CKDT* pin of the VMM, and the ASIC uses that clock to synchronize the data bits that are driven to the FPGA. The data are being sent over the two lines *(DATA0* and *DATA1)*, and the FPGA deserializes the stream using the data clock, at double data rate, resulting to a total bandwidth of 640 Mb/s. Note however, that due to the encoding's overhead, the *effective user bandwidth* is 80% of that, or 512 Mb/s. The FPGA deploys two double data rate input primitives *(IDDR)* [47] that are clocked by the 160 MHz data clock, and accept the two VMM data lines at their inputs. Thus, at each cycle of the data clock, four raw bits are being outputted by the two double-data rate primitives into the FPGA fabric. At the next cycle, these four bits are fed into a 12-bit *Shift Register*. A *Comma Alignment* logic

scans the entire 12-bit register for the 8b10b *Comma* characters. These characters are being sent repetitively by the VMM, so that the FPGA recognizes the word boundaries. In essence, this means that once the FPGA aligns to the stream, it knows exactly which are the 10 bits inside the raw data shift register that must be forwarded to the *8b10b Decoder*[3]. Once this has been established, the decoder receives the encoded words, checks them up against a look-up-table that converts them into decoded bytes as per the 8b10b protocol [33], and if these bytes are not special (or Comma) characters, they are being buffered into a local FIFO. The architecture of the *L0 Readout Core* can be examined in Figure 5.7.

After receiving the *L0 Trigger*, the VMM builds a packet that will be sent to the external device[4]. The first two bytes of the packet is the *VMM Header*, which is sent out regardless if the trigger was matched with an actual event or not. This header contains the 12-bit BCID timestamp of the trigger, after the application of the pre-determined offset, plus a 2-bit extension of the timestamp, called *orbit counter*, together with two flags. If the trigger was matched with data, the rest of the packet is constituted of four-byte chunks, and every chunk contains the hit information from each of the 64 channels of the VMM that its event was matched with the trigger. These are the *VMM Hit Data*, and they contain the information that were forwarded from the 4-event-deep channel FIFO to the L0 buffers of the VMM[5]. All of these data are streamed to the FPGA, which stores them to the local readout FIFO, and once all VMM hits have been transmitted, the VMM will start sending comma characters again. Once this happens, the L0 FSM asserts the *vmmWordReady* signal to the *Packet Formation* component, which prompts the latter piece of logic to forward the words from the readout FIFO to the UDP interface via the *FIFO2UDP* module. The final packet format that is being sent to the DAQ software as UDP payload can be examined in Figure 5.8.

5.4 Two-Phase Analog Readout Mode

In this section, the legacy readout method of the VMM, as implemented in the FPGA's logic, will be described. As mentioned in Subsection 2.2.1, since its first version that

[3] Shortly after the VMM is configured, the FPGA checks the stream for the comma characters (equal with 0011111010 and 1100000101 in raw format, or 0xbc after being transformed into a byte), and if many of these are found in a row, then a signal providing information on the data link health is switched to high. This is driven to the *Packet Formation* component, and the latter piece of logic prepends it to the UDP data packet. If that bit is low, then the VMM is not being read-out at all, and the user can receive feedback by examining the *link_health* bitmask (see Figure 5.8) in the data, which is eight bits in length, one for each of the maximum eight VMMs on the board.

[4] For this case, it is the FPGA, for the final implementation, it is the Read-Out Controller (ROC) - see Chapter 7.

[5] There can be a maximum of 64 hit data words in a packet, one for each channel.

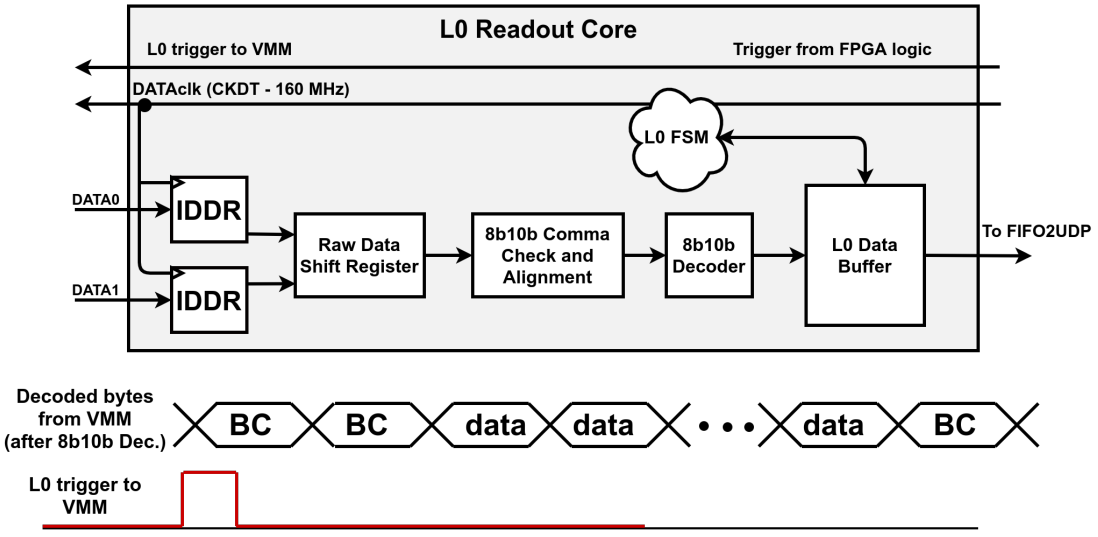


Figure 5.7: Architecture of the FPGA logic that receives the 8b10b encoded stream from the VMM, when operating in *L0 Readout Mode*. The data clock is used by the VMM to send out the datastream, while the FPGA uses that same clock to deserialize it, thus resulting in a fully synchronous communication system. The raw data bits are sampled at double data rate, and are buffered into a shift register where a comma check is performed. The non-comma characters are forwarded to a FIFO. An FSM is responsible for managing the interfacing with the *Packet Formation* component. At the bottom of the same Figure, one can see the timing diagram of the decoded data, shortly after a *L0* has been relayed to the VMM. The VMM sends comma characters that are being decoded into a byte with value 0xbc, but after the L0 signal reception, the VMM will start sending *data* bytes to the FPGA instead. These bytes will be buffered into the FIFO, and eventually reach the DAQ software via UDP.

was introduced in 2012, the VMM features a readout method that does not make use of its internal ADCs to convert the outputs of the peak detector and the TAC. This is the *Two-Phase Analog Readout Mode*, which can be activated by the user by disabling the VMM's internal converters via the SPI configuration. Under this readout procedure, the VMM operates under two distinct phases: the *Acquisition Phase* and the *Readout Phase*. The FPGA determines in which state the VMM is in by controlling the level of the *ENA* signal that is driven to the ASIC. During acquisition, each of the VMM's channels awaits for a pulse to cross its user-defined threshold. When this occurs, the channel processes the pulse using its analog circuitry alone, stores the voltage levels from the peak detector and the TAC into analog buffers, and locks. The channel can no longer process any more inputs until it is being read-out by the FPGA. If at least one of the VMM's channels yields data in its analog buffers, the VMM will pull its *DATA1* pin high (also called a *flag*).

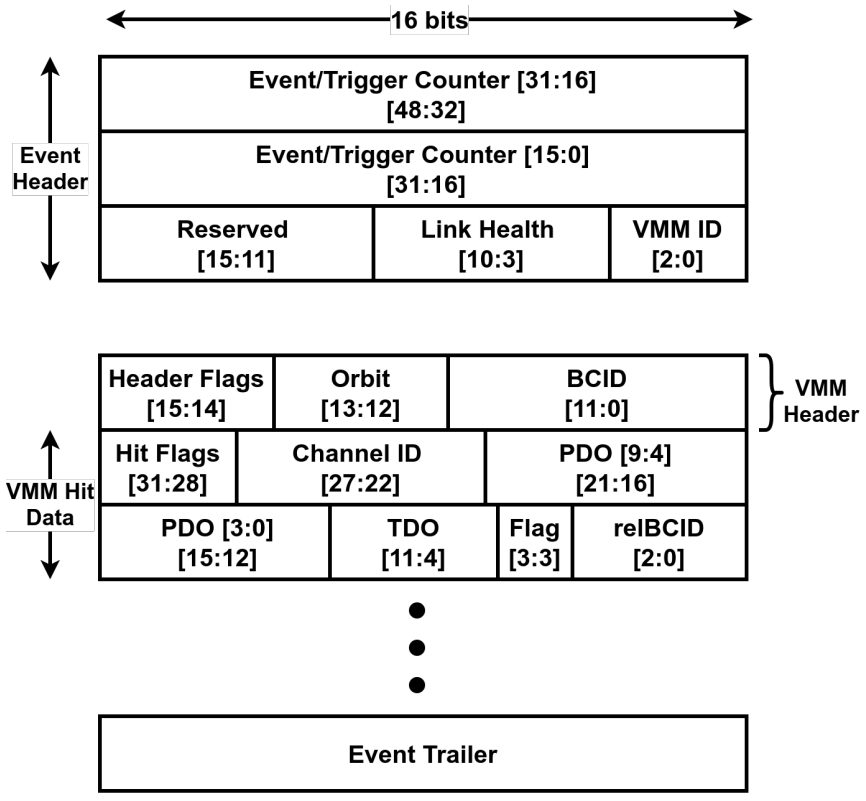


Figure 5.8: Structure of the data packet that the FPGA sends to the DAQ software when operating in L0 Readout mode. The *Event Header* is being generated in the FPGA (by the *Packet Formation* component), while the rest of the data are being sent from the VMM to the FPGA as they are. Note that the *VMM Header* contains the BCID timestamp of the trigger, while each *VMM Hit Data* chunk contains the *relBCID* 3-bit value for each channel hit, which indicates the relationship of the hit's timestamp with that of the trigger's.

The FPGA uses two major internal components in order to extract all necessary data from the VMM. One is the dedicated readout core, that handles the digital logic signals which are driven to the VMM in order to read all the channels out. The other is the FPGA's embedded ADC, called *XADC* (see Section 4.4). It is a 12-bit, 1 MSPS ADC, operated in unipolar mode. The nominal analog input range to the ADC is 0 V to 1 V, with each LSB size being equal to 244 μV. The FPGA implements a supporting logic that operates the XADC, that is being activated (i.e. triggered to sample an input voltage) by the analog readout core that was mentioned above.

Upon the detection of the flag, the FPGA's analog readout logic pulls the *ENA* signal low, thus invoking the readout phase. During this, none of the VMM's channels process any more data, but the channels that had been locked before because they processed a pulse, make their data available to the FPGA. After putting the ASIC into the readout phase, the FPGA issues the token via the *CKTK* pin, and while it is high, it asserts eight

strobes in the *CKDT* pin, and deserializes the $6-$bit address and two flags that the VMM makes available in its data port. The byte is stored temporarily by the core, that then proceeds to trigger the XADC to start sampling. The latter converts both peak detector (PDO) and TAC output (TDO) voltage levels multiple times (i.e. *oversampling*) to eliminate noise induced during that phase. The channel address and the associated samples are then put together in a single UDP packet that follows a pre-defined format, sent to a software suite developed for this readout mode alone [51]. The analog readout mode variant of the FPGA firmware is depicted in Figure 5.9.

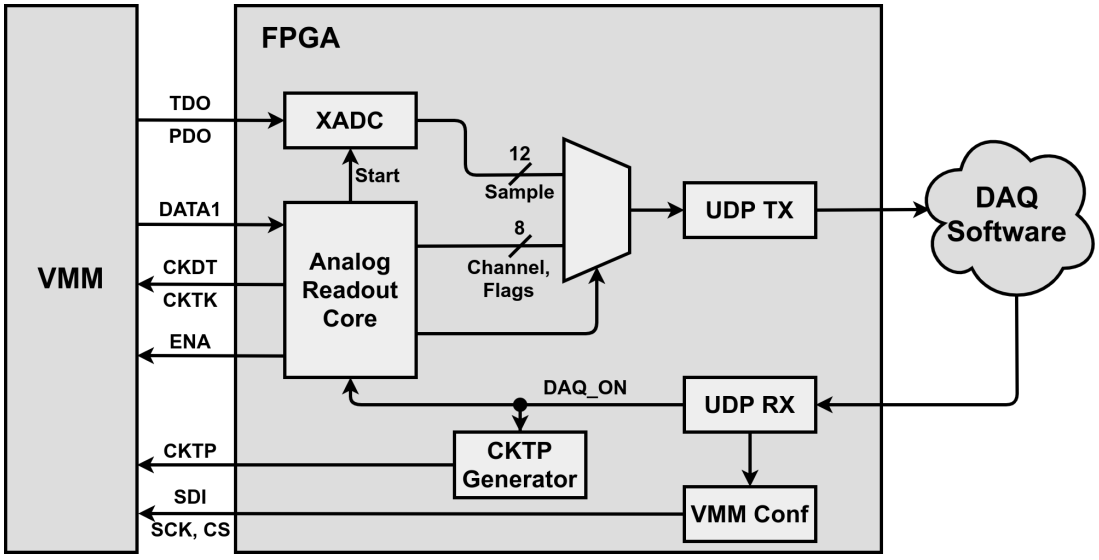


Figure 5.9: Architectural representation of the VMM analog readout firmware. The *DAQ Software* handles the Ethernet traffic to and from the FPGA, which deploys dedicated cores to interface with the network. The FPGA issues the test-pulse *(CKTP)* strobe to the VMM, which injects artificial pulses to the VMM's channels. The *Analog Readout Core* implements the logic depicted in Figure 5.10. Thus, it probes the *DATA1* pin until a flag is raised. Upon the detection of a flag, the VMM is read-out. The *XADC* is being triggered by the readout core to sample the analog voltage levels. The VMM channel address and the samples are being routed to the *UDP_TX* logic, which assembles the UDP payload that is later transmitted over Ethernet.

After sending the packet, the FPGA probes the VMM's *DATA1* pin for a flag again. If another channel has data that can be read, the flag is high and the procedure of deserializing the address and sampling the pulse voltage levels is followed again. If no other channel recorded any data during the acquisition phase, the VMM keeps the *DATA1* pin low, thus prompting the FPGA to reset it. This is being done by issuing three strobes to the token input pin before pulling the *ENA* pin high again, which completes the readout cycle. The VMM can then again process pulses with its channels. A timing diagram depicting the procedure is shown in Figure 5.10.

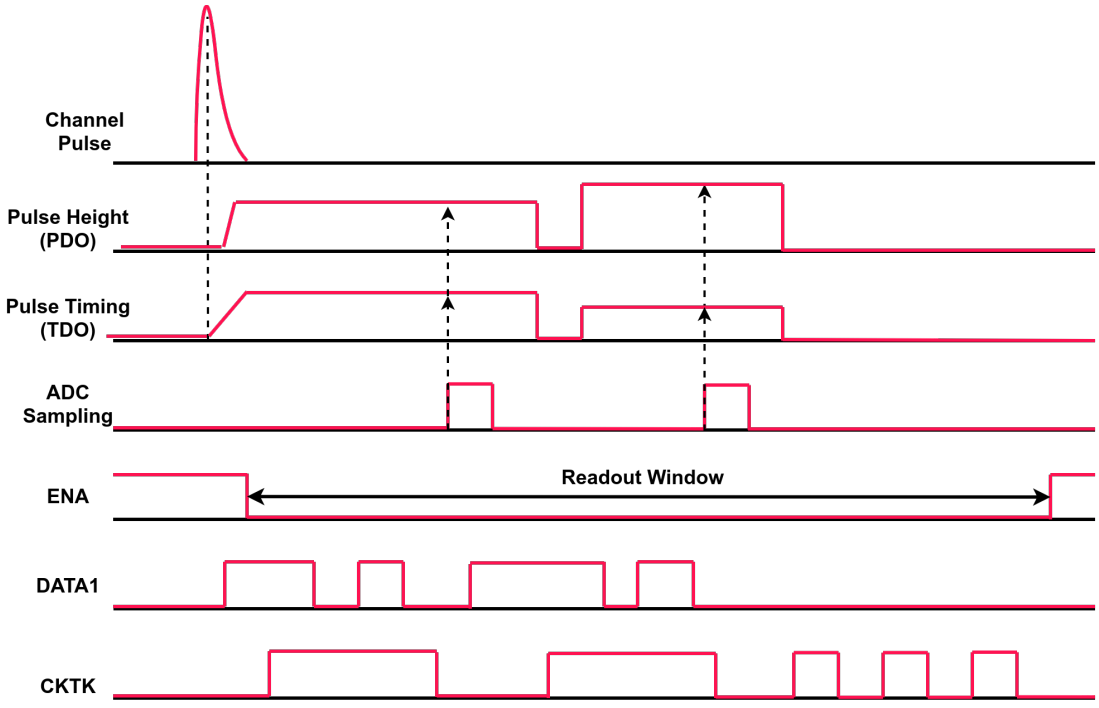


Figure 5.10: Simplified timing diagram of the two-phase analog readout mode. An input pulse is being processed by the VMM's channels, and two voltage levels, one characterizing the pulse's amplitude, and one the pulse's timing are being generated and driven out of the chip via the denoted outputs. The VMM raises a flag at its *DATA1* pin (denoted as *D1* in Figure 2.3), indicating that at least one of its channels yields data. The FPGA grounds the *ENA* signal, thus switching to ASIC into readout mode. During this phase, the $6-$bit address and two flags are deserialized from the *DATA1* port by issuing eight *CKDT* strobes to the ASIC (not depicted), and the *PDO* and *TDO* voltage levels are sampled. The cycle is repeated by issuing another token to the *CKTK* pin, if another channel has data. In this example, two channels have data that can be read-out.

This readout method is primarily used when one wants to read-out the VMM in low-rate conditions, where the need for greater precision surpasses the requirement for fast pulse processing. The VMM was developed for the ATLAS experiment, where the L0 trigger rate will reach $4\,$MHz in Phase-II [7]. Thus, its internal converters were designed in such a way as to be able to cope with the aforementioned trigger rate. Each channel is able to store the converted data within $250\,$ns, a dead-time that is dominated by the speed of the VMM's $10-$bit ADC which digitizes the peak detector's output voltage. However, the chip's excellent analog processing capabilities may be exploited by a slower external converter, that is more precise than its $10-$ and $8-$bit embedded ADCs. This is where the analog readout comes into play, and the implementation in question demonstrates the said scheme, by using the $12-$bit Xilinx® XADC, which also presents excellent linearity, as reported by the associated datasheet [13].

5.4.1 MMFE1 Implementation

The analog readout mode was exercised in the MMFE1 board (see Appendix C), where a single VMM was being read-out under test-pulsing [52]. In Figure 5.11, the results from a test-pulse run on an MMFE1's VMM are shown. Note that this is a similar set of measurements as the one depicted in Figure 4.19, but using the analog readout method instead of the Level-0 mode.

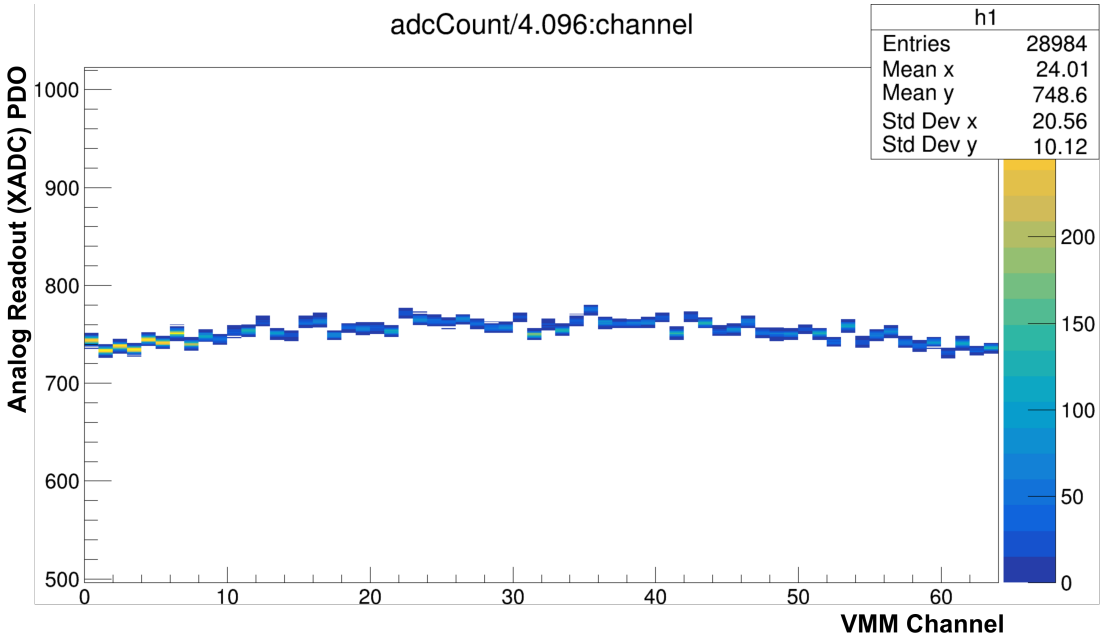


Figure 5.11: Plot depicting the PDO distribution across all 64 channels of a VMM, for one test-pulser DAC value after a test-pulser data-taking session using the VMM's analog readout method. This is the same VMM as the one used when data were taken to produce the results shown in Figure 4.19. The configuration between the two measurements were mostly the same, but the only difference was the readout method, and in the current Figure, the results from the analog readout method are shown. Note that the bell-like structure observed in the Level-0 readout mode is not as pronounced here, which implies that the systematic effect that is present when reading-out the VMM via the Level-0 mode, is probably caused by the VMM's internal ADCs, and not by its analog circuitry.

In Figure 5.12, one can inspect the measured Equivalent Noise Charge (ENC) of a particular channel for different peaking times, when test-pulsing for one DAC value. The VMM was configured at a gain of 16 mV/fC, and four sets of runs with one thousand test-pulses each were taken. At each run, the chip was configured with one of the four available peaking times, equal to 25, 50, 100 or 200 ns. The correlation between the noise and the peaking time is evident; the noise decreases as the peaking

time increases, which is expected[6].

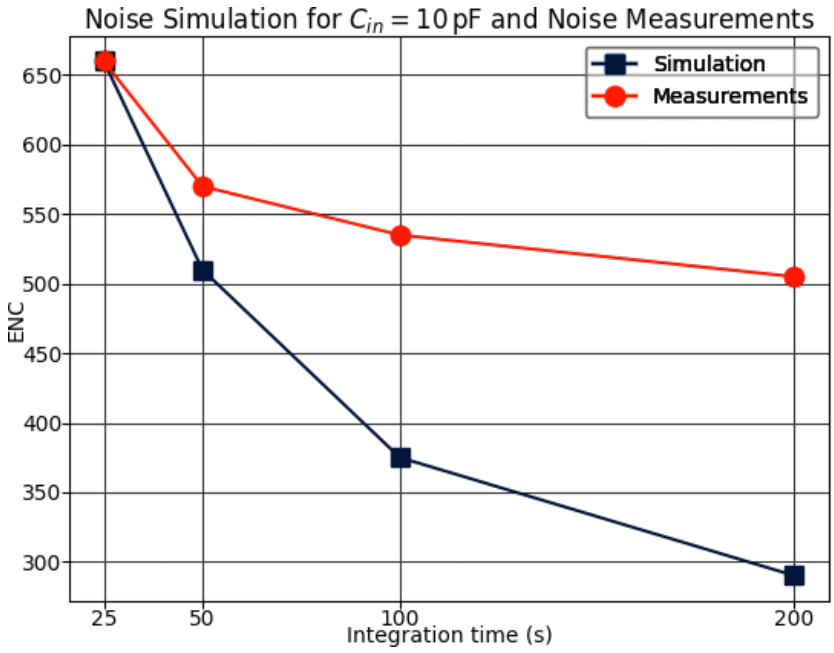


Figure 5.12: Measured (over a single channel) and simulated ENC as a function of the VMM's four attainable integration times, for a gain of 16 mV/fC. Note the discrepancy for longer integration times. It is assumed that the input capacitance to the channel-under-test was $\sim$ 10 pF, given that it was connected to a PCB trace leading to a floating connector.

Even though Figure 5.12 presents the expected behavior, the resulting ENC does not comply with the simulated values. That is, despite the fact the noise follows the anticipated trend, the noise measurements do not correspond to a particular channel input capacitance. This indicates that the channel's analog subsystem has a different response to certain noise components that are present in the system, or simply speaking, external pickup affected the measurements shown in Figure 5.12. Also, parasitics from the channel protection network, might have limited the noise performance of the system for longer integration times [52].

One common source of electronic noise in analog front-end chips for example, are *switching regulators* or *DC DC converters* [34]. These components transform a wide range of input voltage levels into a specific one, designated by the board designer, in order to provide the exact supply voltages to the on-board components. Usually, the converters operate at some internal frequency, that propagates into their output. The MMFE1 deploys four `LT8612` step-down regulators that provide the necessary supply voltages to all of its devices, including the VMM [53]. In order to rule out any interference of the switching regulators to the system's noise performance, a set

[6] Also shown in Figure 2.4, which depicts this relationship with simulation and experimental results.

of diodes that provided the four voltage levels required by the MMFE1's devices were used instead, completely bypassing the converters.

After extensive testing, it was found that another source of pickup was the FPGA's logic itself. The original front-end FPGA readout firmware (including its analog readout variant), utilizes several clocks to operate its logic gates. The firmware deploys two Phase Locked Loops (PLLs), that generate seven clocks, with frequencies ranging from 500 to 40 MHz. Also, the FPGA's transceiver is generating clocks of its own, in order to communicate with the on-board Ethernet PHY. Finally, clock dividers are utilized in the I^2C logic of the design, that create free-running clocks running at kHz range. It is therefore evident that the design has several clock domains that may impede with the analog performance of the VMM's channels. This assumption is validated by inspecting the variation of the baseline voltage of a given channel through the VMM's MO pin, when the FPGA is active and when it is not (shown in Figure 5.13).

The Advanced Analog DAQ Scheme

In order to tackle the FPGA's negative impact on the VMM noise performance, the DAQ system was re-designed as a whole. In the new system, which makes use of the VMM Readout System Supervisory Board [54], the front-end logic was simplified, and the Ethernet interface was omitted altogether. Only one clock domain is used, running at 50 MHz. The front-end FPGA essentially employs a logic, that is subject to control by another FPGA, running at the so-called *motherboard*. The latter device is a Xilinx® VC709 evaluation board that implements a supervisory logic in its FPGA. The motherboard firmware interfaces with the DAQ Software via Ethernet, and receives the VMM configuration data that are to be sent to the VMM. These data are propagated to the front-end FPGA via an E-link[7]. The front-end FPGA logic forwards the configuration bits to the VMM, thus configuring it. The motherboard FPGA can also set the front-end system into acquisition mode, by sending an appropriate command to the front-end FPGA. After this, the *ENA* signal will be pulled high by the front-end FPGA, and the motherboard will *halt* the reference clock to the front-end FPGA. This stops any clock activity on the front-end, thus minimizing noise. The *DATA1 Flag* is propagated to the motherboard by the front-end FPGA. Upon the detection of the flag, the motherboard ends the acquisition procedure, and forces the system into readout mode. It starts the reference clock again, and sends a command to the front-end FPGA to extract the VMM data, in the exact same manner as the original firmware does. The only difference is that the VMM data are not sent out of the FPGA via UDP, but via the E-link interface. On the receiving end, the motherboard deserializes the analog readout

[7] A serial, often bidirectional interface, commonly running over 8b10 encoding. See also Appendix E.

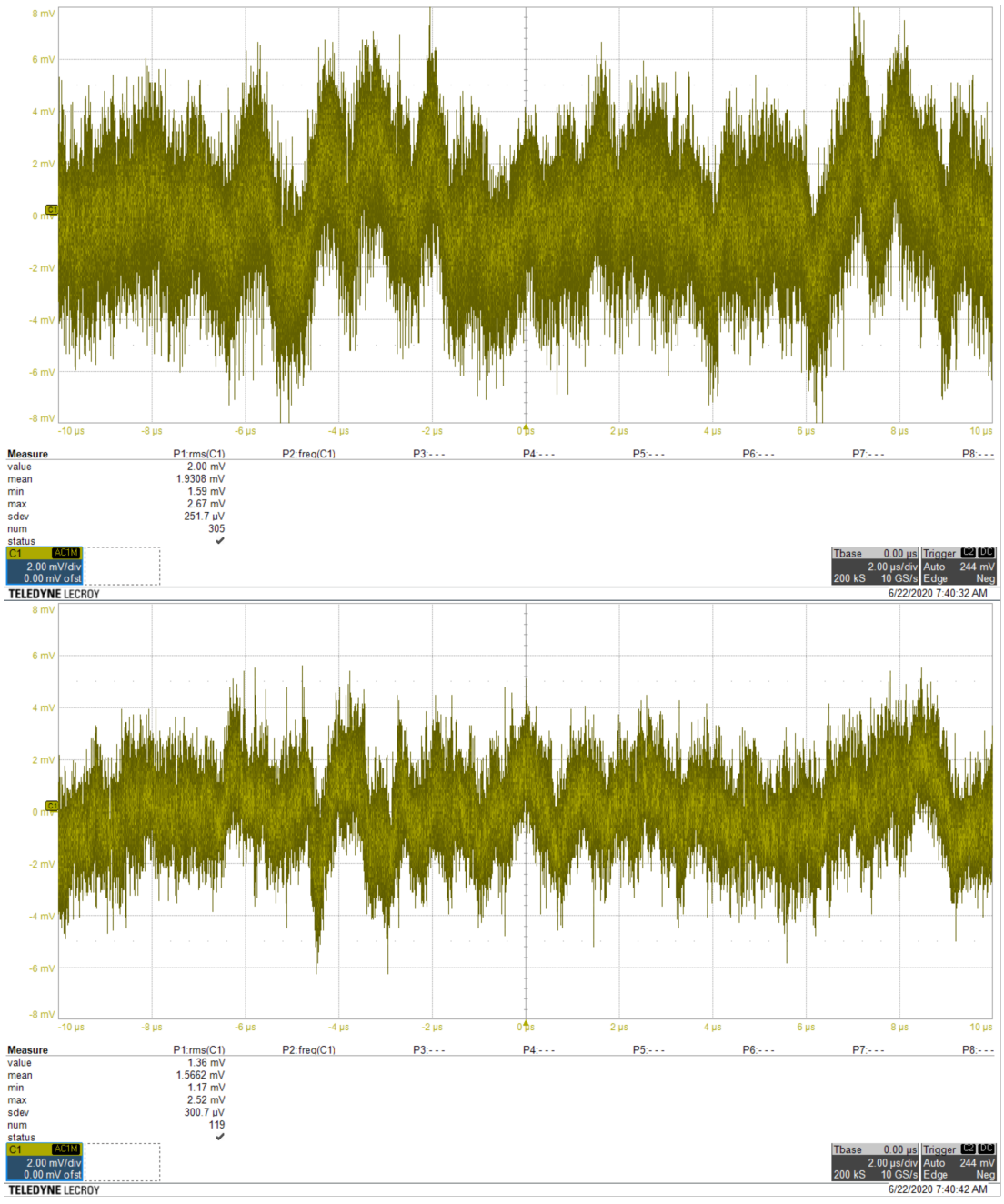


Figure 5.13: *Top:* Baseline voltage variation for a given VMM channel, as measured by a Lecroy oscilloscope, through the VMM's MO pin. *Bottom:* The same measurement, but with the entire FPGA logic inhibited (i.e. clocks grounded, all IP cores held high in reset, and the external PHY in power down mode). The difference in the RMS is evident.

data from the E-link, and relays them as they are to the DAQ software via UDP. After this, the front-end FPGA/VMM pair are again put into acquisition by the motherboard.

The *ENA* signal is pulled up, and the clock is halted to minimize noise pickup during that phase. Finally, it is worth noting that the supervisory board's firmware infrastructure allows for interfacing with several front-end nodes, in a similar manner with the VSB's design, which will be described in Chapter 6. The advanced analog readout scheme is depicted in Figure 5.14.

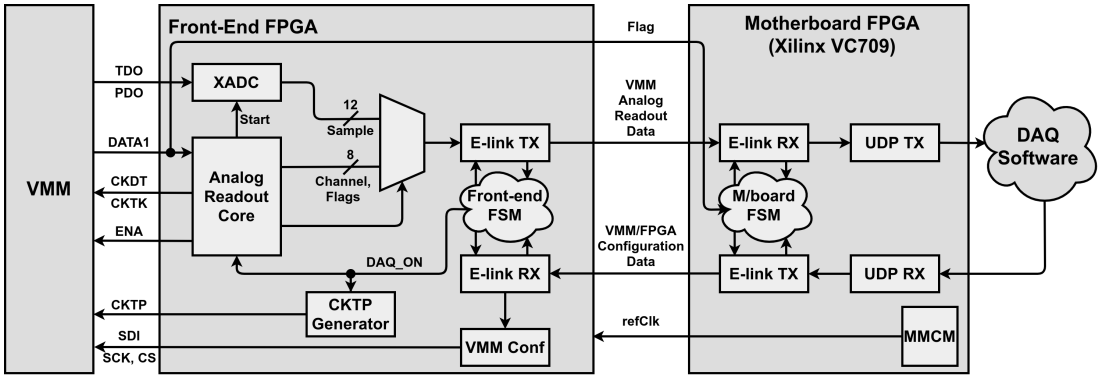


Figure 5.14: The advanced analog DAQ scheme. The front-end FPGA deploys the essential modules for the VMM's readout. It interfaces with the motherboard FPGA via a serial bidirectional E-link. The motherboard in turn communicates with the DAQ software via UDP. It receives the VMM and FPGA configuration data, and forwards them as they are to the front-end FPGA via the serial interface. It also controls the reference clock, starting and stopping it accordingly, in order to prevent any interference with the VMM's acquisition mode. Upon the detection of a flag, the motherboard commands the front-end FPGA to read the VMM out, and receives the data via the E-link, before sending them to the DAQ software for storage and analysis.

The logic was tested extensively under test-pulse conditions, that allowed for system readout efficiency measurements, for various channel occupancies, as a function of the trigger rate. These are particularly useful if operating the system under detector readout scenarios, where the motherboard FPGA may also receive a scintillator coincidence signal, prompting it to probe the VMM's *DATA1* flag upon the reception of an external trigger. Indicatively, the readout efficiency of the system is about 74% for a trigger rate of 1 kHz, if 4 VMM channels are being read-out.

Also, the scheme seems to have successfully reduced noise - to some extent. However, the ENC measurements taken under it, still do not correspond to a specific input capacitance, further indicating that the front-end board picks up noise that is amplified and dampened in a different manner, depending on the VMM's peaking time and gain. Nevertheless, the encouraging results indicate that the concept of the scheme may provide a base for any future work in this readout methodology. A noise comparison between the two systems are shown in the following table. The reduction in ENC for each peaking time and gain combination is shown in parenthesis on the new scheme's

corresponding ENC cell.

Gain (mV/fC) / **Peaking Time** (ns)	**New Scheme's ENC**	**Old Scheme's ENC**
16/25	560 (17.9%)	660
16/50	542 (5.17%)	570
16/100	484 (10.0%)	535
16/200	475 (6.12%)	505

Conclusions

In this Chapter, the architecture and functionalities of all of the FPGA's VMM readout modules were described. The logic behind the blocks implementing the Continuous Readout was laid out, including the procedure where the CKBC is strobed in order to define a more precise timing between the arrival of an event and the VMM's reference clock, and to provide a noise-free dataset, by taking advantage of the VMM's self-reset capabilities. The Level-0 readout methodology was also illustrated, emphasizing on the logic implementing the 8b10b interface in the FPGA, that operates at the maximum speed as per the VMM's specifications, in order to minimize dead-time (as will become evident in Chapter 6). Finally, the Two-Phase Analog Readout Mode was revisited. This data extraction methodology can be used in low-rate scenarios, where an external ADC can covert the VMM's peak detector and TAC voltages with more precision compared to the VMM's internal converters, at a slower speed. The readout method was demonstrated, alongside a more advanced scheme, that focused on minimizing the noise that was being picked up by the VMM's analog front-end.

Chapter 6

The VMM Readout System and its Testbeam Implementation

As mentioned in previous Chapters of this dissertation, the FPGA design supports the readout of the VMM under testbench and testbeam conditions. The main motivation behind using the ASIC off-chamber, is to validate the chip's performance and to calibrate it, prior to its deployment into a detector readout scheme. The FPGA design was developed by taking this into consideration. That is, the architecture of the firmware provides the necessary flexibility to satisfy the needs of both VMM readout use-cases, under different hardware implementation environments[1], without requiring heavy modifications in the design's source files.

This Chapter focuses on the testbeam application of the FPGA firmware dedicated in reading-out the VMM and described in Chapters 4 and 5. It covers two testbeam campaigns, held in August 2017 and July 2018 at the Prevessin H8C SPS CERN site. The main goal of the first testbeam was to validate the ASIC's tracking and triggering performance in a small scale experiment involving two small $10 \times 10\,\text{cm}^2$ Micromegas (MM) prototype chambers [30], read-out by two MMFE1 boards. The second testbeam's goal was to verify the VMM's performance not so much as a stand-alone tracking and triggering agent, but as a means to read-out the first large Micromegas detector prototype, named Small Module 2 (SM2 - see Subsection 2.1.1). This required an experimental setup on a larger scale, that involved many Data AcQuisition (DAQ) nodes. In order to support the scheme, the VMM Readout System Supervisory Board (VSB) FPGA firmware was developed.

[1] The firmware supports many FPGA packages and VMM/FPGA-bearing boards (see Chapter 4).

6.1 The August 2017 Testbeam

The third version of the VMM, named VMM3, was released in early 2017. After verifying the chip under test-bench readout conditions using its built-in test-pulsing circuitry, its performance in real data acquisition scenarios had to be evaluated as well. Hence, a testbeam campaign was carried out, which took place in August 2017 at the H8C SPS CERN site. Two MMFE1 boards were used (see Appendix C), and each was connected to one small $10 \times 10\,\text{cm}^2$ Micromegas prototype chamber (also called the *T-chamber*). The prototypes were placed back-to-back along the beamline, which would carry either pions or muons. These would ionize the gas of the detectors, and due to the applied high voltage, charge would accumulate and drift inside the active area, thus resulting in current, that the VMM channels would pick-up from the readout strips they were connected with (see Section 2.1 for more info on the charge formation mechanism). A photograph of the arrangement is provided in Figure 6.1.

The readout method that was chosen was that of the *Continuous Readout Mode*. To be precise, the technique used was not the conventional that utilizes a free-running reference clock (CKBC), but that of the *CKBC Strobing*[2] (see Section 5.2 and Subsection 5.2.1). The reasons behind using the strobing methodology are essentially two. First of all, this scheme offers *noise suppression*. As discussed in the relevant Subsection, the CKBC strobing indirectly inhibits any data that were generated in the VMM but are not correlated with useful events. Since the necessary reference clock strobes will only be issued if a trigger exists, pulses that crossed the threshold due to an irrelevant causality (e.g. a noise burst, or a cosmic muon stimulating the detector medium), will never be buffered due to absence of a clock, which is directly inferred by the absence of a trigger.

Another reason why the CKBC strobing is preferred, is that it allows for a *precise definition of the event's* t_0. In order to understand this concept, a data acquisition system comparison will be made. In ATLAS, protons collide in the center of the detector at the rising edge of the system's reference clock, which is the bunch-crossing clock, or CKBC, distributed across all the system's nodes, including the VMM. Hence, particles are generated at a precise point in time and space. Given also that the time-of-flight of a muon potentially generated due to a collision is also known[3], the t_0 of the event is in turn well-defined. This aids in *track reconstruction*, as the μTPC method creates tracklets within a Micromegas layer using the fine timing of all strip hits correlated with the muon that ionized the detector's gas [3, 4, 55]. This timing is measured using

[2] The strobing readout mode was actually developed to fulfill the requirements of the implementation in question.

[3] For the NSW, this amounts to about 30 ns.

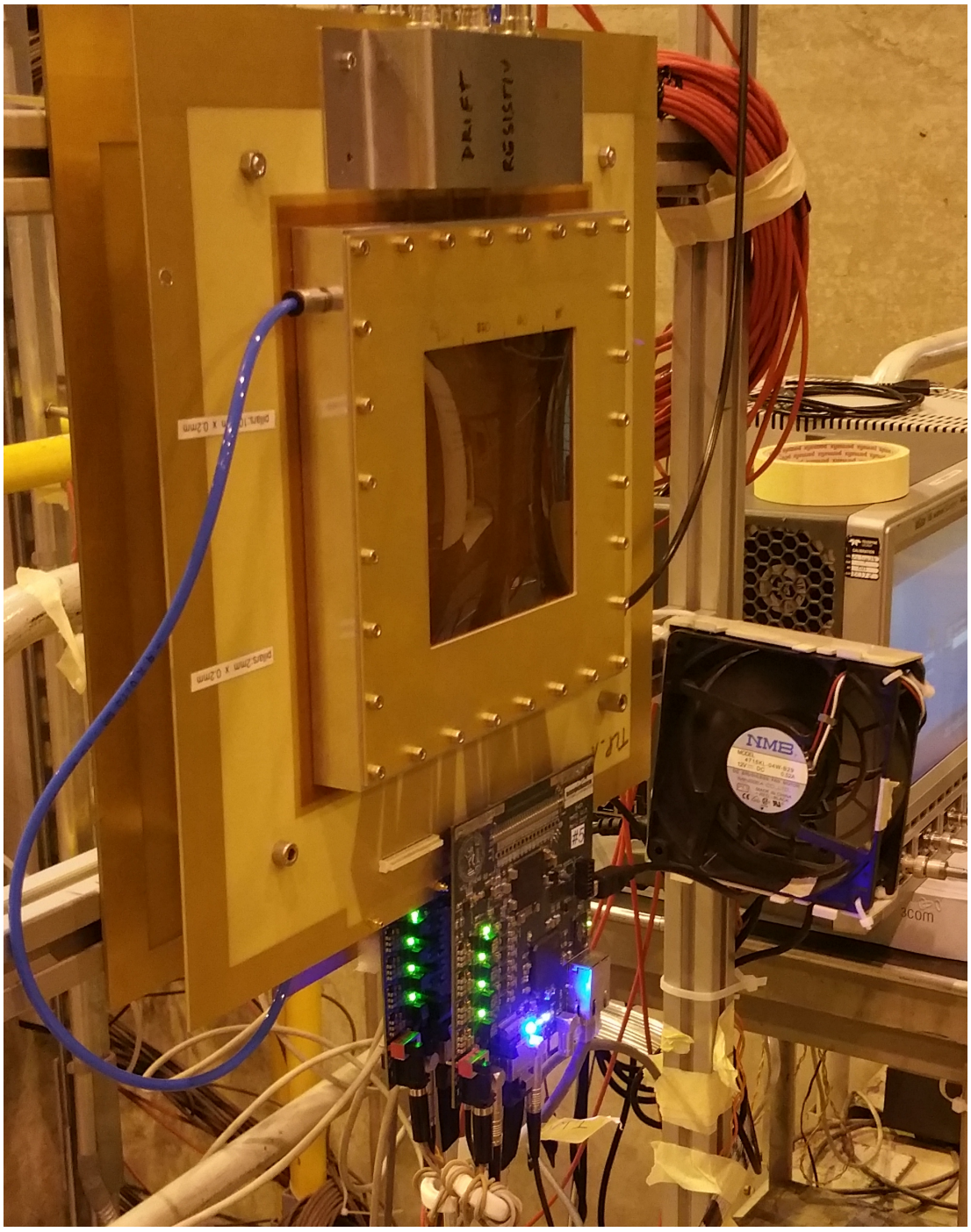


Figure 6.1: Photograph of two Micromegas T-chambers used in the August 2017 testbeam. The two MMFE1 boards, bearing a VMM and an FPGA each, are clearly visible in the bottom of each detector.

the VMM's TAC and TDO. However, in the testbeam setup, the t_0 is not defined

with precision, because the reference clock is not synchronous with the events and each corresponding trigger. The reference clock of the acquisition system is generated locally, and the particle spills have no correlation with that clock. The strobing method though, circumvents this, because the first CKBC clock tick that stops a hit's TAC, has a well-defined latency with respect to the trigger. The trigger is being generated by a scintillator coincidence, which issues a logical pulse (with negligible jitter) when a beam particle traverses the setup. This pulse is driven to the FPGA, which starts issuing the CKBC strobes. The first strobe (and all the following), has a precise latency with respect to the trigger, which infers an exact t_0, similar to the one of ATLAS. Otherwise, the timing of the data would have had a jitter of 25 ns. A timing diagram depicting this logic can be examined in Figure 5.5.

Hence, the experimental setup and the VMM configuration were defined around the concept of strobing. That is, a TAC ramp of 100 ns was chosen, and the trigger setup was arranged in such a way, as to be able to provide a trigger pulse to the VMM (via the FPGA), in under 100 ns. This was imperative, as if the trigger would arrive after the ramp duration, even the data related to the particles of the spill would get discarded due to the TAC's saturation. They would therefore fall into the case depicted in Figure 5.4, but with a CKBC strobe actually arriving, although arriving after the channel had reset itself. The triggering setup consisted of a set of scintillators with Photomultiplier Tubes (PMTs), and an arrangement of the standard Nuclear Instrumentation Modules (NIMs). The raw output from each scintillator-tube pair, drove a discriminating module, which outputted a logical pulse if the signal crossed a user-defined threshold. That flag was fed into a coincidence module (essentially a simple AND gate), that was receiving the output from two scintillators, deposited across the beam line. Charged particles (muons or pions) would therefore stimulate both scintillators, inducing the final *trigger* signal, produced by the coincidence NIM module. This triggering setup, comprised of conventional NIM modules, is common across many experimental high-energy physics testbeam implementations. However, another NIM module was designed specifically for the needs of the VMM-bearing front-end boards DAQ system, and will be briefly described in the following subsection.

6.1.1 The Clock and Trigger Fanout Module

The Clock and Trigger Fanout (CTF) module, is a common reference clock source, that can also convert single-ended LEMO signals to differential (LVDS) outputs. The MMFE1 board [44] has three trigger inputs: one comes in the form of a *LEMO* connector, that accepts a single-ended trigger signal. The other two inputs, are differential μHDMI and miniSAS connectors, that receive up to four discrete LVDS

signal pairs. The CTF has four outputs of each MMFE1 differential connector type, each producing three output pairs. One is the reference clock, originating from a local oscillator, with a frequency of 25 ns. Another one is the trigger output, as received from the CTF's LEMO trigger input. The final one is a *soft-reset* signal, which is another signal used for system synchronization[4]. In the implementation at hand, the trigger signal from the scintillator coincidence was fed to the FPGA via the MMFE1's LEMO connector. The CTF was used as a clock and soft-reset signal source, driving them to the two MMFE1s of the setup via two miniSAS cables of the same length. The trigger was opted to be sent directly to the MMFE1s via the single-ended LEMO input, as if the CTF were chosen to carry that signal as well, extra latency would had been invoked, potentially leading to event saturation. The setup can be viewed in Figure 6.2.

6.1.2 The Necessity of the Busy Signal

The design of the firmware logic imposes a limitation to the system with respect to its availability during runtime. Upon the reception of a trigger, a DAQ node (i.e. the FPGA of an MMFE1), will start extracting data from its VMM, and until that process has finalized, *no other trigger* can be registered. When the front-end FPGA firmware (see Chapter 4) is configured to read the VMM via the continuous readout scheme using a finite number of CKBC strobes, a sequence of events takes place in the FPGA logic. First of all, the *Event Counter* or *Trigger Counter* increments by one. This value is prepended to the main body of the UDP packet that will be sent to the DAQ software (see Figure 5.3). Then, the clock strobes are issued to the VMM, forcing it to halt the TACs across the channels that have registered a pulse. The digitized data are stored into the channel FIFO. The FPGA then starts issuing the token clock via the *CKTK* pin in order to cycle through the data-yielding-channels. In-between the tokens, the CKDT clock is toggling, thus deserializing the information from the VMM's FIFO. All of the event's data are then sent over to the software via UDP alongside some metadata in the form of a header[5]. The FPGA is ready to receive another trigger only after writing the last byte of the UDP payload into the *FIFO2UDP* module. This process lasts several microseconds, and the problem arises from the fact that this amount of dead-time is *not constant*. It depends on the amount of channels that are read-out by the FPGA.

Consider a hypothetical scenario where a trigger is sent to the two MMFE1s of the system. They would both read-out their VMMs, and send the data packet to the

[4]Upon reception of the soft-reset signal, the FPGA forwards it to the VMM, by grounding the *ENA* pin of the ASIC for one bunch-crossing. This causes the BCID counters of the VMM to be reset. Since the reset signal is received in a synchronous manner by all FPGAs of the system, the counters of all VMMs are essentially reset a the same time, thus synchronizing them.

[5]As mentioned before, the header's main content is the trigger sequence number.

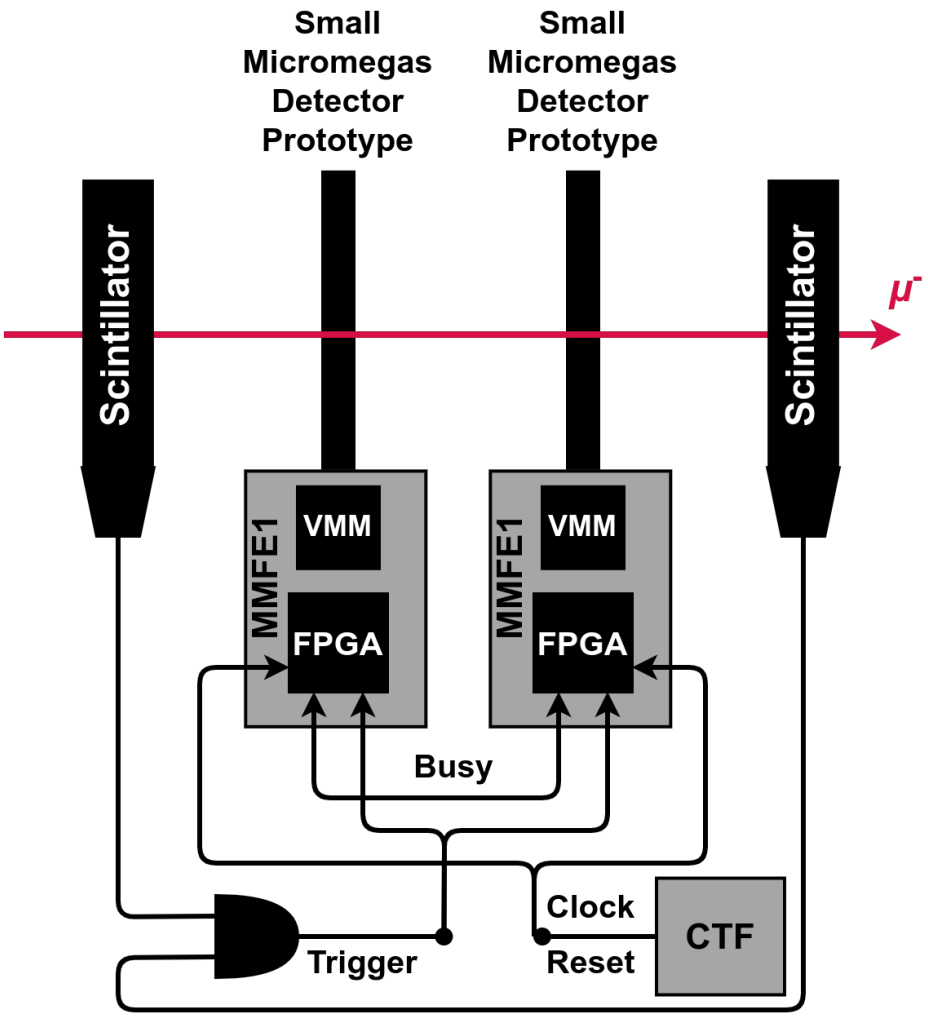


Figure 6.2: Block diagram of the August 2017 testbeam experimental setup. The CTF is providing two signals to the MMFE1s, via miniSAS cables, over LVDS differential signaling. These are the common *reference clock*, and the *soft-reset*. The latter signal, is inputted to the CTF via a LEMO connector on the module's backplane, and its state is managed remotely from the testbeam's control room. The two scintillator signals are fed into a coincidence module (after passing from discriminators which are not depicted), and the coincidence's output is driven to the MMFE1 FPGAs via the front-end board's LEMO connector. Finally, the MMFE1s exchange a *busy* signal via their μHDMI ports. The importance of that signal is discussed in Subsection 6.1.2.

software. The software would then register the packets, and group them under the same event, as the trigger counter of both packets have the *same value*. If however one MMFE1 read-out six VMM channels, but the other one read-out ten, the second FPGA would take more time to become available. What if another trigger arrives, when one FPGA is ready to receive another trigger, but the other one is not? The answer is, that

they would then go *out of sync*. This scenario is depicted in Figure 6.3.

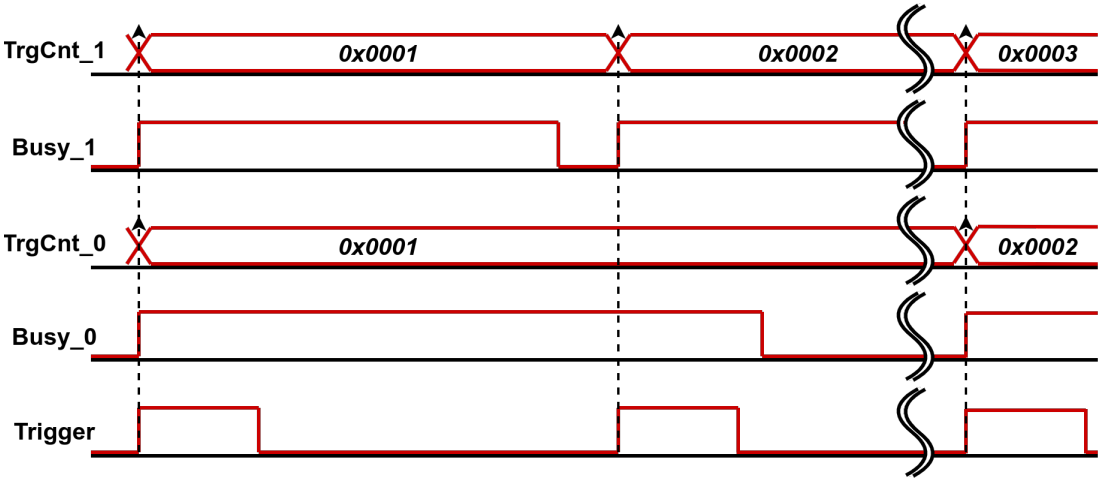


Figure 6.3: Timing diagram depicting a problematic use-case in the said DAQ system. The first trigger arrives, and both nodes take their time to finish extracting and sending-out their data. However, one is slower than the other, and when another trigger is issued, the faster node registers it because it finished earlier, but the slower one simply ignores it. The end result is a loss of synchronization. The trigger counter of the FPGA that accepted the second trigger wanders off by one count. After some time, another trigger is registered by both nodes (since they are both available), and even though the data extracted by both nodes are correlated since they belong to the same trigger, the event counter value of the FPGAs are off by one, and the software will not be able to group these events together.

There are two ways to work around this problem. The simplest is to use a *trigger veto* in the scintillator coincidence output. This veto prevents the issuing of any trigger subsequent to the one that was just asserted, for a configurable amount of time. The amount of forced veto time, depends on the maximum dead-time that an FPGA node might exhibit while reading-out the VMM. This of course, reduces the efficiency of the system, as triggers that could had been registered by both nodes of the system, are instead discarded. The most efficient way to circumvent the issue, given the current setup, is to have a shared *busy* signal between the two FPGAs. While the one FPGA is reading-out data, it holds its busy flag high, and sends it to the other FPGA, which uses it to inhibit inbound triggers, if a use-case like the one depicted in Figure 6.3 occurs. This solution ensures that the system is in full sync, while not reducing its efficiency at higher trigger rates.

6.1.3 Results

The testbeam proved to be successful, fulfilling the ultimate goal of validating the third version of the VMM under testbeam conditions. It also aided in defining the optimal

configuration of the front-end ASIC, and the operational parameters of the detector itself. The VMM front-end FPGA firmware turned out to be catalytic throughout the process, through its robustness ease-of-use. The readout system provided datasets with millions of events, recording about 200 K events per spill with a per-channel occupancy of around 5 kHz. The analysis that was conducted [56] mainly revolved around the derivation of the Micromegas *spatial resolution* using the VMM, for perpendicular tracks with respect to the detector's readout plane[6], and for angled tracks as well. The resolution of the small Micromegas prototype using the VMM3 was calculated to be about 65 μm for normal tracks, and around 107 μm for a 30° angle between the chamber's mesh and the beamline, after some preliminary analysis. The resolution for these angles is performed using the μTPC method [3, 4, 55], which requires the precise definition of a t_0 for a given event. The CKBC strobing readout method aided in this cause, thus helping to validate the VMM's readout performance, despite the fact that the beam was not natively synchronous with the reference clock of the system.

6.2 The VMM Readout System and its Supervisory Board

Besides validating the front-end ASIC's performance, the July 2017 testbeam also proved the robustness of the FPGA-based readout system that was used. Within a few minutes, a run with millions events could be taken, thus easing the process of evaluating the performance of the VMM and the Micromegas chamber under various operational conditions.

However, the system described in the previous Section, was simple in the sense that only two DAQ nodes were deployed. Usually, in testbeam scenarios, more front-end boards need to be used. Given the architecture of the readout firmware though, this poses a severe limitation to the system design that was illustrated above. After the FPGA logic receives an external trigger (usually originating from a scintillator coincidence), it will go into a state in which it cannot accept other triggers until a particular sequence of events has completed. To be precise, the final step is to write the very last byte into the *FIFO2UDP* module (see Chapters 4 and 5), and the firmware is ready to accept another trigger only after this occurs. This issue was first described in the previous Section, where the continuous readout mode with CKBC strobing was used, but is true for all readout modes that the FPGA supports. However, different readout modes result in different *dead-times*. Careful design[7] of the logic that extracts

[6] This type of dataset processing is covered in slightly more detail in Subsection 6.3.3.

[7] Essentially, this means that the gateware must be arranged in such a way as to minimize latency.

the data from the VMM and forwards it to the software via UDP, diminishes that time, thus increasing the readout efficiency of the system. In a multiple-node DAQ system comprised of the FPGAs that deploy the given front-end VMM readout firmware, the only way to prevent the problem depicted in Figure 6.3, that stems from what was just described, is to have a common *busy* signal. However, it is difficult, or even impossible, to share a busy signal between multiple nodes. One solution would be to use a trigger veto, which forces the system efficiency to take a heavy toll.

The most efficient solution is to have an independent module, that supervises the DAQ state that each front-end FPGA is in. That common node must generate and distribute the system clock, fan-out the trigger and a reset that synchronizes the counters across the system, and must receive the busy signal from each front-end FPGA, and use it to inhibit triggers that are sent to the FPGAs, if at least one of them is occupied in reading-out VMM data. These are the basic requirements it must satisfy, in order to meet the criteria of the DAQ orchestrator. The VMM Readout System Supervisory Board (VSB), fulfills all of the above, and will now be described.

6.2.1 System Overview - VSB Firmware Architecture

The VMM Readout System (VRS) Supervisory Board [44] is an FPGA-based system that interfaces with other front-end boards that bear VMMs and FPGAs which implement the firmware described in Chapters 4 and 5, with slight modifications. These front-end boards are also called *VRS front-end nodes*. The VSB firmware has been implemented successfully in a Xilinx® VC709[8] evaluation board. Connectivity is achieved through a custom FPGA Mezzanine Board (FMC). The design is flexible enough to be ported to any board which is able to connect to several VRS front-end nodes. The general scheme is shown in Figure 6.4. The VC709 implementation also receives signals[9] from the CTF module (see Subsection 6.1.1), that are relayed to the VRS front-ends accordingly. It connects to up to eight front-ends, and to one CTF, via nine miniSAS connectors, through the FMC. It also implements an Ethernet interface to communicate with the DAQ software via UDP. The software forwards commands to the VSB logic, which can steer the DAQ system's state according to the user's needs.

The main modules of the VSB FPGA design [54] are shown in Figure 6.5, and are listed and briefly described below.

- `Main DAQ FSM`: This component is the supervisory state machine of the VSB FPGA's logic. It interfaces with all other building blocks of the design and

[8]The FPGA's package ID is *XC7VX690T-2FFG1761C*.
[9]Reference Clock, Soft-Reset and Trigger.

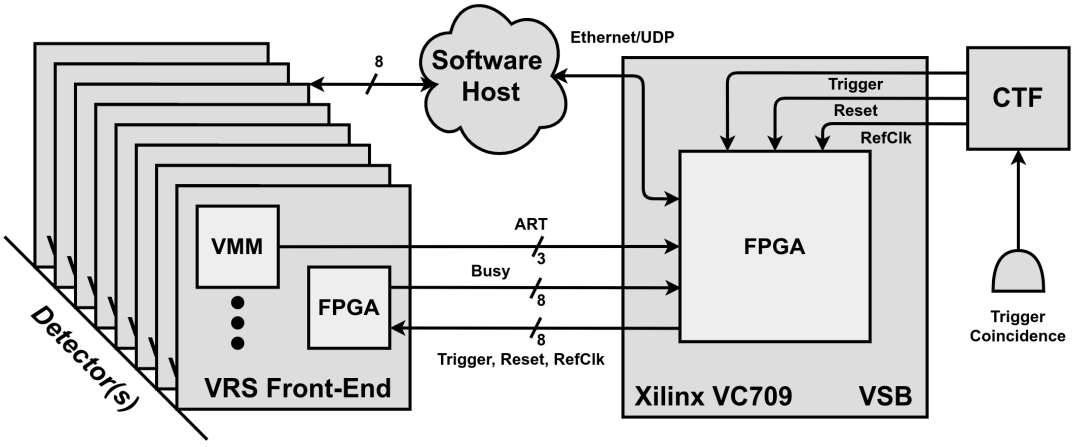


Figure 6.4: Overview of the VRS, when using the VSB in conjunction with the front-ends. The VSB firmware implemented in the VC709 FPGA can support up to eight front-end node connections. These connections are implemented through differential pairs over miniSAS cables, and carry the *Reference Clock, Soft-Reset and Trigger* signals. These originate from the CTF, which generates the system clock internally, receives the reset signal from a LEMO cable that is driven from a module outside of the radioactive area of the setup, and receives the trigger from a scintillator coincidence. The VSB can also receive the *ART* trigger primitive datastream from up to three front-ends (see Subsection 6.3.2). Finally, it receives the *Busy* signal from all eight front-end FPGAs. All (up to nine) FPGAs of the system connect to the network via UDP over Ethernet, and the DAQ software (named *VERSO* - see Appendix D), controls the system's overall state and receives VMM data from the front-ends.

defines the overall state in which the system is, depending on the user's directive as it is forwarded by the *UDPdin_handler*.

- `Ethernet Wrapper`: Partially based on the 1Gb/s UDP/IP Stack design from OpenCores [46], this sub-module provides interfacing capabilities with the software host. Modifications have been made to the original design to accommodate some Internet Control Message Protocol (ICMP) functionalities. The block also deploys the Xilinx® GTH Transceiver and Tri-mode Ethernet Media Access Controller (TEMAC) IP cores that interface with the on-board PHY chip to provide $1\,\mathrm{Gb/s}$ Ethernet-over-copper connection with the software host.

- `UDPdin_handler`: This module registers the incoming UDP payload data from the software and applies the values in internal configuration registers accordingly. It also forwards any commands for switching the system's state to the Main DAQ FSM.

- `Busy Logic`: Essentially a simple OR gate that registers all single-bit *Busy* flags

as they are driven to the VSB FPGA by the (maximum eight) front-end FPGAs connected to the VSB. If at least one VRS front-end node cannot accept triggers, the *Global Busy* signal will be high, thus inhibiting any triggers issued by the Trigger Module.

- `Trigger Module`: Accepts a trigger signal from the CTF, and if the inhibiting signal from the busy logic is low, the trigger is forwarded as-is to the TTC block.
- `Trigger Timing and Control (TTC)`: Emulates the ATLAS trigger signal propagation scheme. Using only two bits instead of eight (which is the standard followed in ATLAS), this module accepts two signals from the CTF: trigger and soft-reset. If a trigger is received, a single-bit L0 trigger that is high for 12.5 ns is issued along the TTC line. If a soft-reset is received, a single-bit reset signal that is high for 12.5 ns is issued along the very same TTC line that also carries the L0 trigger. The only difference is that the L0 trigger's positive edge is aligned with the positive edge of the reference clock, whereas the reset's positive edge is aligned with the negative edge of the reference clock. The front-end FPGA deserializes the inbound TTC stream using the 40 MHz reference clock at double data rate. If a high bit is sampled on the positive edge of the clock, an L0 is registered, and if a high bit is sampled on the negative edge of the clock, a soft-reset is registered. This scheme is used to reduce the number of required connections in the system. More information about the ATLAS TTC system can be found in [57], while a brief description is also provided in Chapter 7.

6.2.2 Level-0 Readout within the VMM Readout System

The front-end readout firmware supporting the VSB is more or less the same as the one described in Chapters 4 and 5, but with a few modifications and some application-specific details that will now be discussed. The first difference comes in the form of a new addition: the TTC reception logic, that decodes the inbound 2-bit trigger stream originating from the VSB, using the 40 MHz reference clock it also receives from the VSB, at double data rate. The logic resembles that of the ATLAS TTC system, but at a lower frequency. Upon the reception of a trigger, the TTC receiver forwards it as-is in the form of a Level-0 trigger to the VMM. If a soft-reset is received, the FPGA grounds the *ENA* signal driven to the associated VMM pin for one clock cycle, thus resetting the BCID counter of the ASIC. Because the reset is broadcast to all front-end FPGAs in a synchronous manner, the end result is a system where all events of the same trigger have equal BCIDs.

The only readout method that is used in this scheme, is the *Level-0 Readout Mode*,

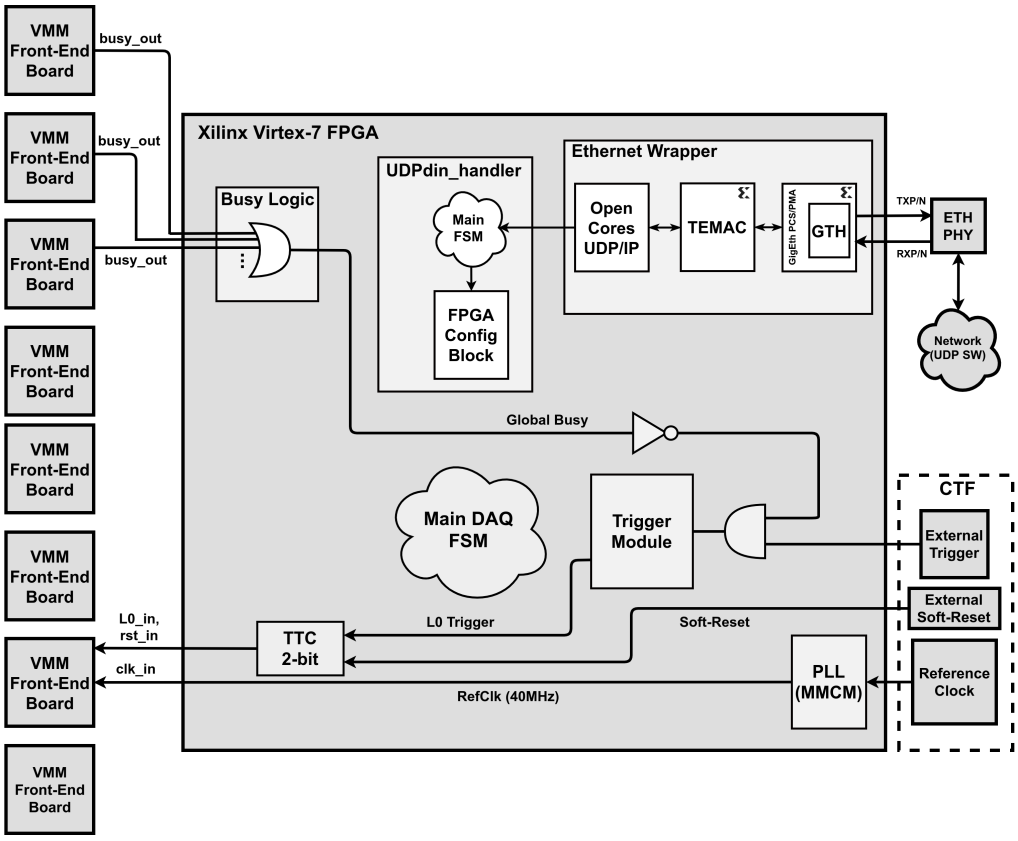


Figure 6.5: Block diagram of the VSB firmware, including its interfacing with all other components of the system.

because it is optimized with respect to the imposed dead-time, as will be discussed later in the current Subsection. In essence, the VMM that is attached to the detector processes input pulses, agnostic if these are related to a beam particle or not. The scintillator coincidence will send a trigger-high signal to the VSB via the CTF, if a particle traversed the setup. The VSB will then register that trigger, and *wait* for a configurable amount of time. Then, the trigger will be issued to the front-end FPGAs in the form of an L0 trigger via the TTC module, and the front-end FPGA will immediately forward it to the VMM, and go in busy mode until the last VMM data word has been buffered in the UDP FIFO. Naturally, in order for the L0 signal to be matched with the particle data it is associated with, the VSB's waiting time must be set by taking into account both the VMM's L0 latency configuration, the intrinsic latency of the trigger system, and the VMM's integration time which shifts the shaped pulse forward in time. A timing diagram depicting the system is provided in Figure 6.6.

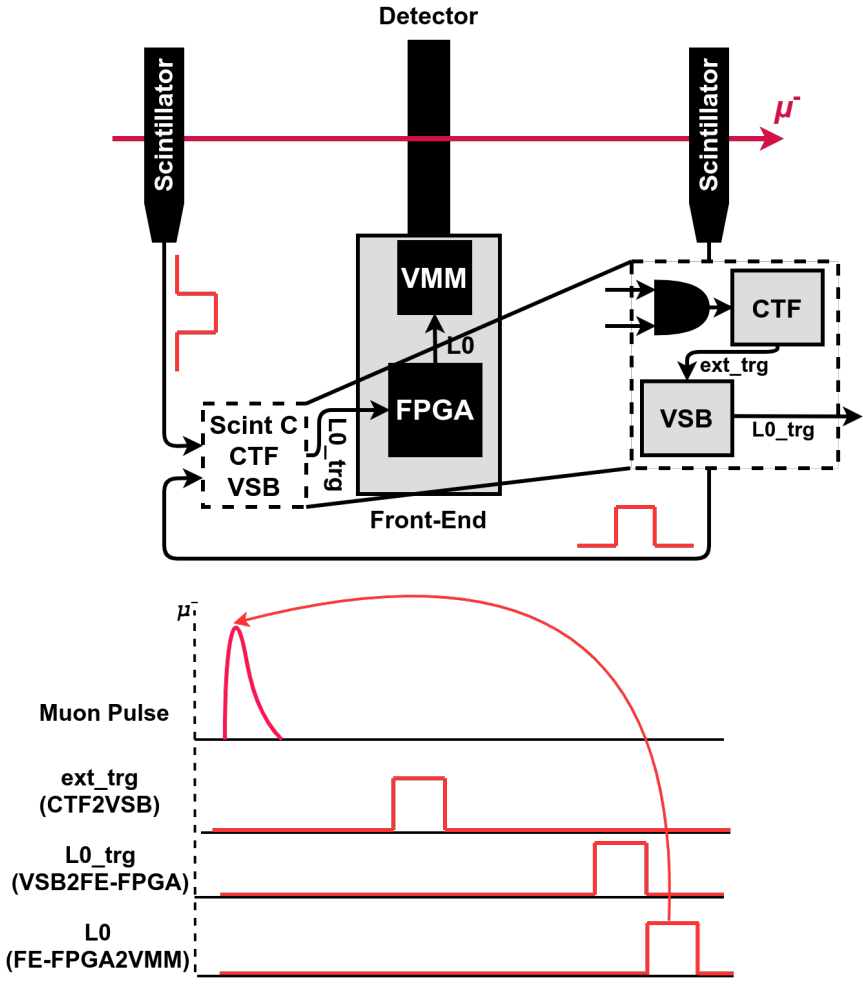


Figure 6.6: Timing diagram of the VRS using L0 readout during a testbeam. Only one front-end node is used in this example. A muon from the beam excites the detector and the scintillators, and the VMM registers the chamber-generated pulse(s) related to that muon. The raw scintillator signals drive a coincidence module, which in turn sends the trigger associated with that muon to the *CTF*. The latter module relays the trigger (denoted as *ext_trg*) to the *VSB*. The VSB waits for a configurable amount of time, its amount determined by the user, which depends on the setup's trigger latency (from the scintillators down to the reception of the trigger by the VSB), and the VMM's configuration. The VSB will then drive the *L0_trg* to the front-end FPGA, which after registering it, will send it to the VMM in the form of the *L0* signal. The VMM will then "look back in time" based on its L0 selection logic (see Section 5.3) and forward the muon hits to the DAQ software via the FPGA.

L0 Readout Efficiency and Dead-Time

The main advantage of the L0 mode, is the fact that it has minimal latency as opposed to the other VMM readout methods. The VMM data extraction interface of the L0

readout, transfers the data at an effective bandwidth of $512\,\mathrm{Mb/s}$[10], and each channel hit is $32-$bit long, whereas the continuous readout mode is much slower, with a bandwidth of just $100\,\mathrm{Mb/s}$, and with each channel hit being $38-$bit long.

When an L0 trigger is issued, the VMM starts the transmission by first sending the header (two-byte long), and then the channel hits[11] (four-byte long) to the FPGA, over its two data lines. It takes five cycles of the $160\,\mathrm{MHz}$ CKDT to clock-in a $10-$bit word from the VMM, and after two cycles the word will have been decoded and buffered to the FPGA's L0 FIFO. Therefore, a hitless data packet will take $75\,\mathrm{ns}$ to get stored into the FPGA, a data packet with one hit will take $200\,\mathrm{ns}$ to get buffered, with eight hits approximately $1\,\mu\mathrm{s}$, and so on. However, the post-trigger dead-time is not equal to that latency, as there is also the overhead of transferring these words from the readout core's buffer to the FIFO2UDP memory, which is a slower process (see Figures 5.1 and 5.7 for logic block visualization), with a duration that increases as a function of the VMM's channel occupancy. This data forwarding procedure is carried out by the *Packet Formation* component, which also has to write the FPGA header into the UDP memory, and utilize some handshaking signals between the FSMs. The end result is that a one-channel-hit packet results in a dead-time of about $2\,\mu\mathrm{s}$[12]. Although larger than just $187.5\,\mathrm{ns}$, the dead-time exerted by the system is still considered small for most testbeam implementations, since the trigger rate of these applications does not usually exceed the few tens of kHz range. Figure 6.7 shows the efficiency of the system as a function of the trigger rate.

6.3 The July 2018 Testbeam

After verifying the VMM's functionalities as a front-end ASIC for data-taking using a small Micromegas detector prototype in 2017 (see Section 6.1), the next step that had to be taken, was to use the chip to acquire data from a large Micromegas prototype. That detector, called *Small Module 2* (see Section 2.1), was the first large Micromegas prototype ever produced, and had been tested extensively by its production group using older versions of front-end electronics. Hence, the performance of the prototype (hereinafter also referred to as *Device Under Test (DUT)*), had to be evaluated while being read-out with the 3rd version of the VMM, which did not differ much from its final, production version.

[10] The base bandwidth is $640\,\mathrm{Mb/s}$, but the 8b10b encoding comes with a 20% overhead.

[11] No hits are are sent in the absence of L0 trigger matching.

[12] It is worth pointing out that these times refer to single-VMM front-end boards (e.g. the MMFE1). If data from more than one VMM have to be extracted (e.g. in an FPGA-MMFE8 - see Figure 4.1), the time to forward out the data from all readout cores, each one of which is associated with one VMM, is a multiple of the latencies that are given above.

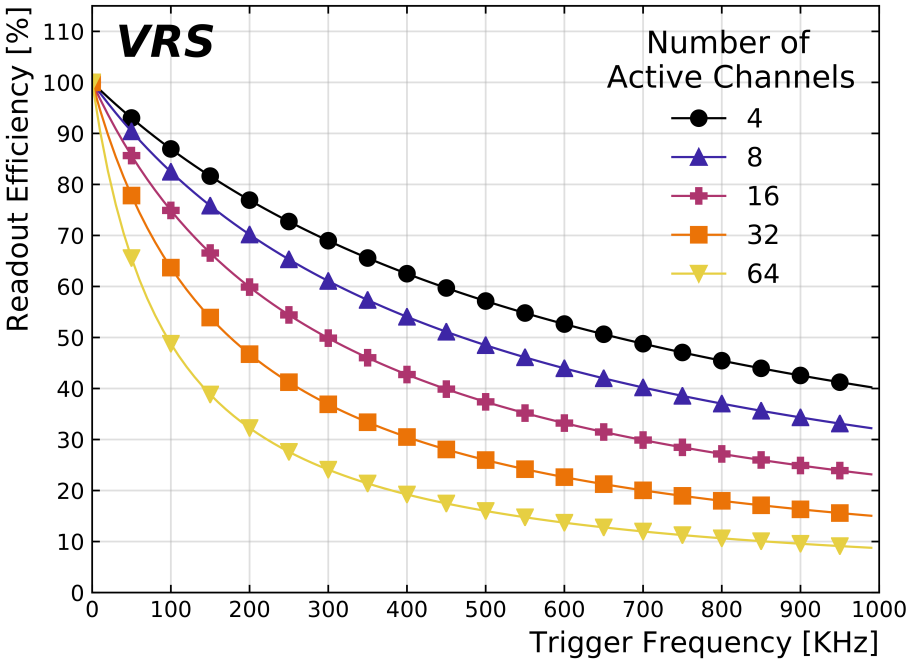


Figure 6.7: Readout efficiency of the VRS readout system as a function of the trigger rate, for several channel occupancies, when running in L0 readout mode [44].

The testbeam campaign lasted for the entire month of July 2018 at the H8C SPS CERN site. In order to characterize the DUT's *efficiency*, a tracker had to be set-up. A tracking apparatus usually consists of more than two independent detector media, of known performance, that are placed at the beam line alongside another chamber (for the current case, the SM2, or DUT). If hits are recorded on all tracker elements, then a track is reconstructed based on their geometrical arrangement and the position of the charge cluster in each chamber. That track is extrapolated onto the DUT, and hits are probed in it, to determine its hit yielding efficiency. The tracker consisted of four small T-chamber Micromegas (the same type that was used in the August 2017 testbeam), read-out by an eight-VMM[13] FPGA-bearing front-end, named MMFE8 (see Figure 4.1). The DUT was read-out by the same type of front-end boards, also four in total. Each board was being placed to one detector *layer*. The SM2, as in all production Micromegas modules, consists of four distinct layers (see Section 2.1). The first two are called *eta*, and the other two *stereo*. The eta's readout strips are arranged in parallel with respect to the detector's horizontal plane, while the stereo layers have a 1.8° angle, in order to reconstruct tracks in two dimensions. A Photo of the tracker's T-chamber and a diagram of the SM2 are provided in Figure 6.8.

[13]Hence, 512 channels or strips per T-chamber were read-out.

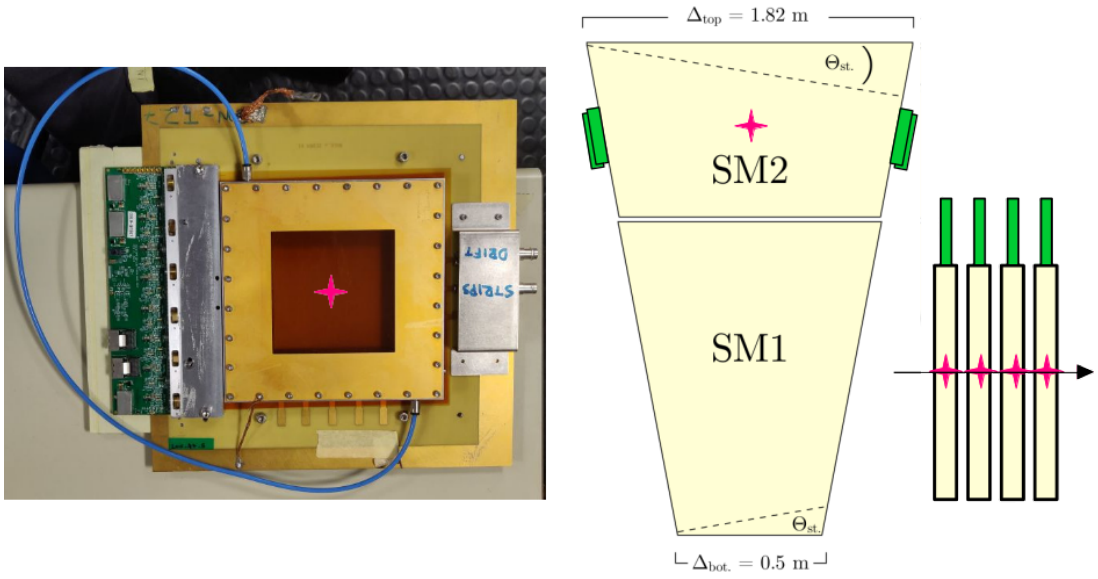


Figure 6.8: *Left:* The T-chamber used in the tracker with the MMFE8 readout board attached to it. *Right:* Diagram depicting the SM2 detector and its four read-out boards (in green rectangles). On the far right, the four SM2 layers alongside their respective readout boards can be seen with respect to a beam traversing the chamber. All detector depictions feature the point where the beam is crossing, denoted by a red star.

Due to the amount of DAQ front-end nodes, the VRS-VSB system was deployed (see above, in Section 6.2). The arrangement is a scaled-up version of the one depicted in Figure 6.6, and its diagram can be viewed in Figure 6.9.

6.3.1 The Fine Triggering Method

In August's 2017 testbeam, one of the main reasons behind using the CKBC strobing readout method, was to synchronize the incident particle timing (i.e. the trigger) with the reference clock of the system (see Section 6.1). ATLAS performs this by definition, as the protons collide at each rising edge of the bunch-crossing clock. Having a constant phase relationship of the particle with respect to the system clock, leads to a jitter-less t_0, which facilitates angled track reconstruction, as it relies heavily on a precise definition of the strip's hit timing within a given layer.

However, the design of the readout system in July's testbeam, prevents from using the CKBC strobing methodology. The first reason is that the L0 readout mode of the VMM is not compatible with the CKBC strobing, as the ASIC's digital logic that performs the L0 matching requires a free-running clock. The continuous mode could have been chosen instead, but then the readout efficiency would have been impacted negatively, because it is much slower as a data extraction scheme, thus resulting in larger dead-

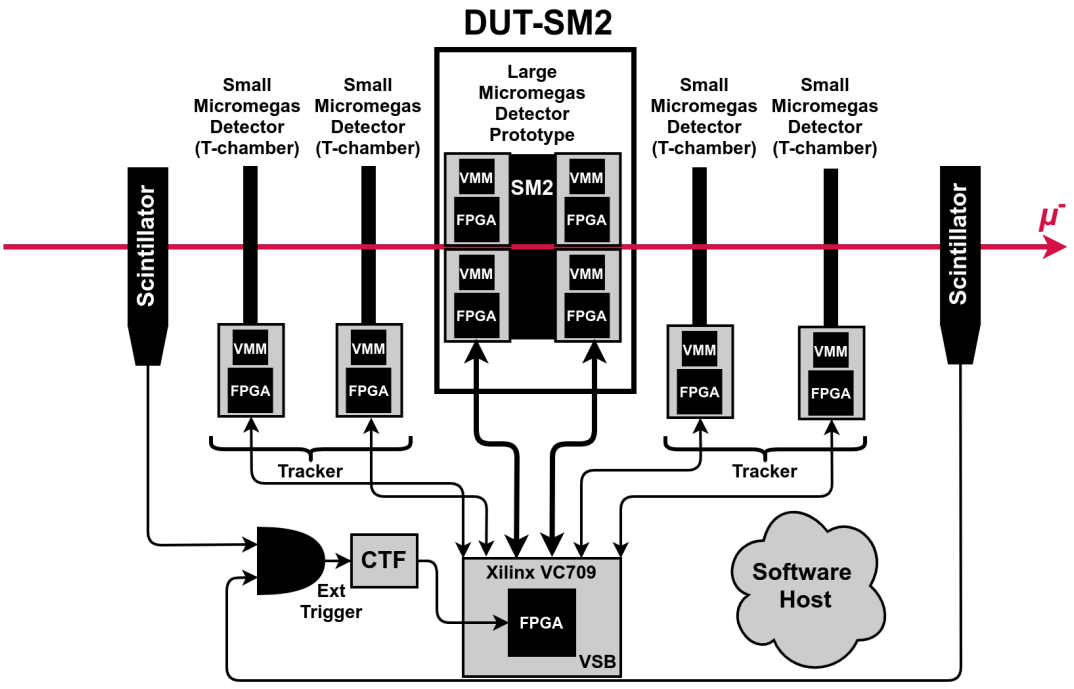


Figure 6.9: Diagram of the experimental setup of the July 2018 testbeam. In the middle, the SM2 chamber, or DUT, is being read-out by four front-ends. Four T-chambers used as a tracker are arranged on the left and right, each being read-out by one front-end as well. The VSB connects to all DAQ nodes, and fans-out the reference clock, trigger and soft-reset. The scintillators provide the trigger signals to the coincidence module, which drives the trigger input of the CTF. The DAQ software host connects to all nine FPGAs of the system via Ethernet/UDP.

time[14]. If the increased dead-time could have been afforded, then using the strobing method would have forced to use a longer TAC ramp, as the addition of the VSB and the extra cabling increases the trigger latency, thus setting a more restrictive limit to the trigger's time-of-arrival, which must be injected to the VMM prior to the TAC's saturation (see Figure 5.4). Also, a longer TAC ramp means that the timing resolution is worsened, which is incompatible with the implementation's requirements.

Hence, the L0 readout mode had to be used, but the trigger had to somehow be synchronized with the reference clock of the readout system. The solution was implemented in the VSB's *Trigger Module*, via the *Fine Trigger Logic*. This component samples the 40 MHz reference clock with a much faster one that has a constant phase relationship with it as it originates from the same PLL. That fast clock has a frequency of 640 MHz. The fast clock in question also samples the *trigger input* from the CTF. A simple logic

[14] Also, the continuous readout mode does not feature "clean" readout, which the L0 selection logic provides seamlessly.

then decides if the system's clock rising edge is aligned with the rising edge of the raw input trigger signal. If it is, then that trigger is forwarded to the front-end nodes, provided the busy signal is low. If the trigger is not edge-aligned with the CKBC, then it is discarded. A rough block diagram of the logic, alongside a timing diagram are shown in Figure 6.10.

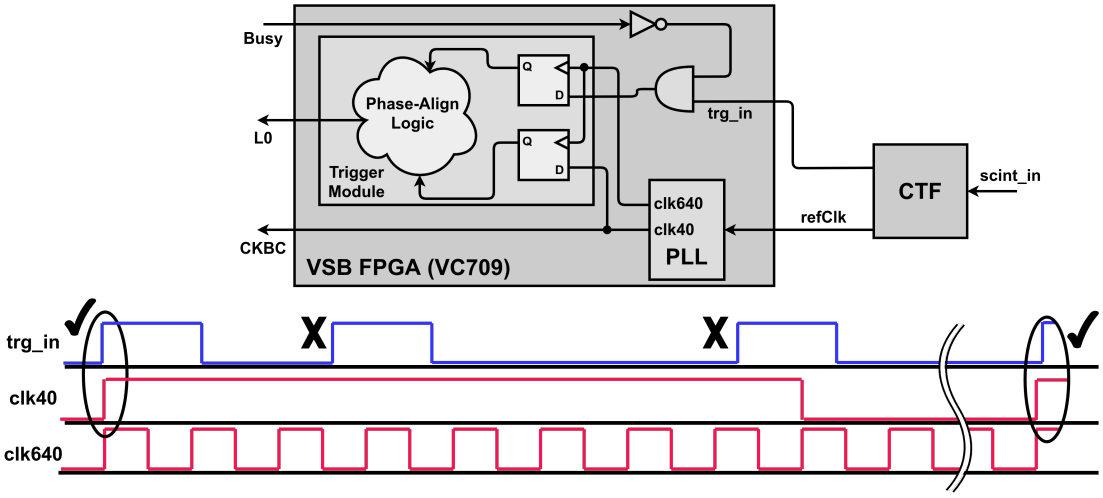


Figure 6.10: Top: Block diagram of the fine triggering logic. The VSB's *Trigger Module* accepts the trigger from the *CTF*, alongside the reference clock (or *CKBC* in the VMM's jargon), and the 640 MHz fast sampling clock which registers both the CKBC and the raw trigger *(trg_in)*. The Phase-Align logic (also clocked at 640 MHz), takes both registered signals, and determines if their rising edges coincide. Bottom: Timing diagram depicting the behavior of the Phase-Align logic. The encircled and checkmarked input triggers will be forwarded to the front-ends as an *L0* signal, thus initiating readout.

The logic described above, creates a *pseudo-synchronous* system, where the triggers injected to the front-end nodes are aligned with the reference clock, or CKBC, that the VMM uses to perform its precise timing. Therefore, since the scintillator trigger is in full sync with the particle's time-of-arrival (with negligible jitter), the fine triggering scheme emulates the clock-particle synchronization that characterizes the ATLAS DAQ. This defines a precise t_0, thus easing the track reconstruction of angled tracks in the SM2.

In order to validate the scheme's performance, a VMM channel was used to measure the timing of the raw trigger input. An MMFE1 was employed, in which a PCB workaround had been conducted: the connection of the VMM's last channel with the board's connector to the chamber had been removed, and the LEMO trigger input was being routed to that channel, after passing from a voltage divider that was bringing the trigger logic level down to the channel's effective range. The scintillator coincidence signal, was being routed to both the CTF, and that channel. Since the VSB was issuing

L0 triggers to the front-ends only when the scintillator coincidence was edge-aligned with the reference clock (with 1.5625 ns accuracy), the channel would only forward the data of the digitized trigger signals that fulfilled this prerequisite. Hence, since the channel measures the raw trigger's timing with respect to the reference clock using its TAC, the *TDO* value of that channel would yield the phase relationship between the *CKBC* and the *trigger coincidence*. A small TDO spread would indicate phase-alignment, and a large TDO spread would imply that the two signals are not phase-related. Two plots depicting this are provided in Figure 6.11, thus proving that the method indeed aligns the system clock with the trigger.

6.3.2 The ART Processing Core

As mentioned in Section 2.2, the VMM is a combined tracking and triggering agent. It is capable of performing precise measurements that are selected using the L0 signal and propagated to the back-end tracking data aggregating components, and at the same time provide signals that are used by the trigger chain of the experiment that employs it. In the NSW upgrade, which is the implementation that the ASIC was originally designed for, the VMM is attached to the Micromegas and the sTGC chambers, to process the muon-induced pulses generated by the detector media. The readout (or tracking) path of both detector technologies follows the L0 readout logic. The trigger scheme however, differs between the sTGC and the Micromegas. The sTGC path uses the direct timing outputs of the VMM, while the Micromegas trigger path utilizes the Address in Real Time (ART) functionality, where in each bunch-crossing (i.e. every 25 ns), the $6-$bit address of the channel that fired first (if any event was recorded by the chip's channels) is serialized over a dedicated pin to a trigger primitive aggregating component.

In the current part of this dissertation, where the VMM has been employed in several FPGA-based readout schemes, only the L0 readout/tracking path has been demonstrated. Its trigger primitive outputs have never been used in any part of the FPGA logic. The *Trigger Module* of the front-end FPGA firmware for example (see Figure 4.3), is designed to accept a trigger signal either from the *CKTP Generator* when test-pulsing the VMM (see Figure 5.6), or from an external source, which is usually a scintillator coincidence source (see Figure 6.6). Whichever the case may be, the trigger is forwarded to the VMM in the form of the L0 signal, and the VMM uses that signal to match it with any buffered data that are correlated with that trigger. The matched data are then collected by the FPGA and sent to the DAQ software for processing and storage via a UDP connection.

In an attempt to fully demonstrate the VMM's functionalities, the *ART Data Processor* was developed. This module, implemented in the VSB firmware, is capable of

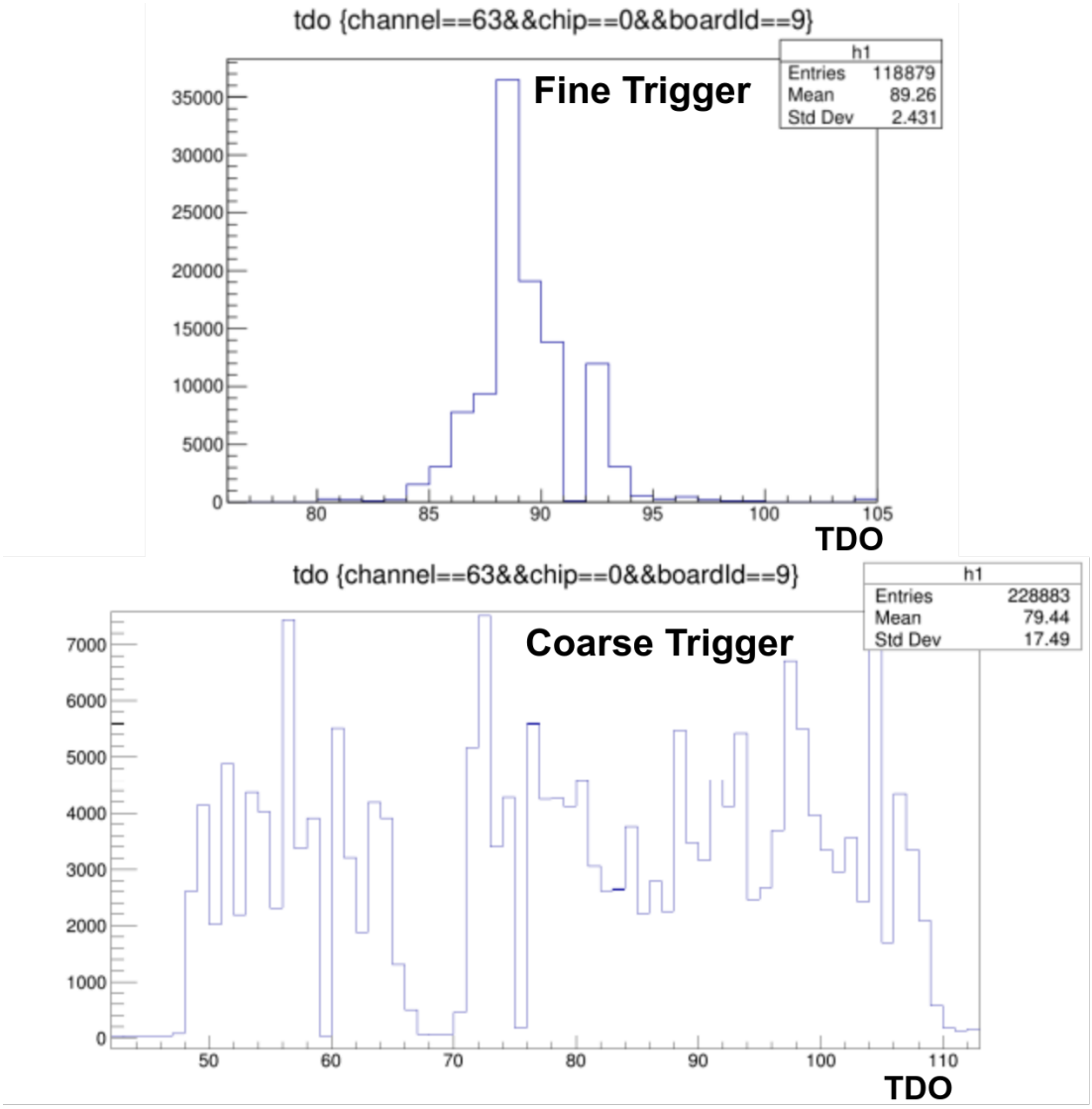


Figure 6.11: TDO spreads of the DAQ system's MMFE1 VMM channel 63, that digitizes the raw trigger, and measures the phase relationship of that signal with the reference clock. Top: The fine trigger methodology results in a 600 ps standard deviation of the TDO distribution (each bin corresponds to 250 ps), but if the original, or *coarse* triggering scheme is used (where all triggers are forwarded to the front-ends, regardless of their phase relationship with the CKBC), the spread is much wider.

receiving the ART data from up to three front-end VMMs. These VMMs are situated in successive layers of the SM2 detector. The reason behind choosing these VMMs over the ones that read-out the tracker chambers, is the fact that the strips of the SM2 across its layers are aligned with each other (with negligible error), thus facilitating the trigger primitive processing, as it will become evident later on in this section. The

ART processing logic combines the ART addresses from all front-ends, and if these resemble a track, then a trigger is created inside the FPGA, and is relayed back to the front-end nodes. The end result is a *self-triggered DAQ system*, which makes full use of both core functions of the VMM, and does not make use of any external triggering stimulus (i.e. scintillators) at all. A rough block diagram of the VSB firmware when the ART processor is deployed, is shown if Figure 6.12.

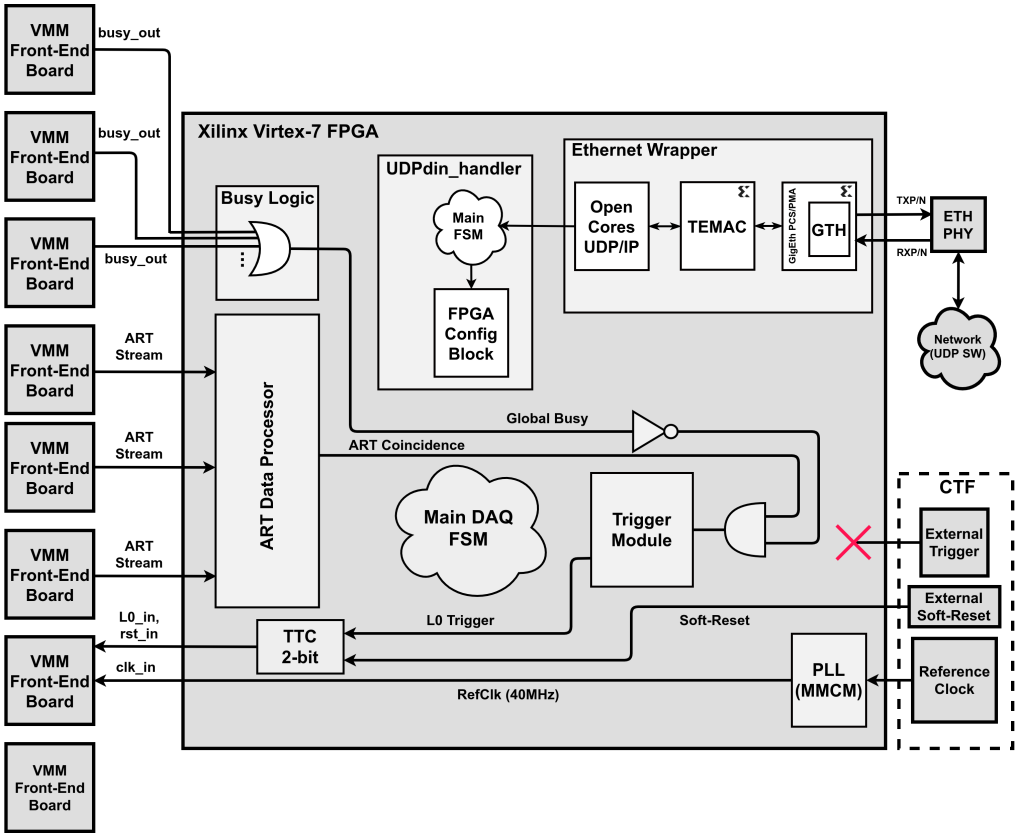


Figure 6.12: Block diagram of the VSB firmware, when the *ART Data Processor* is implemented. Three front-end boards send their ART data to the VSB, and the processor aggregates them. If addresses are received by every input, then the core performs a trigger decision. That decision is positive if the address combinations resemble a particle track, therefore the *ART Coincidence* signal is forwarded to the *Trigger Module*. The latter component does not use the *CTF* trigger input at all, but only employs the ART to issue a trigger to the system, which initiates a readout cycle that will extract the data that are correlated with that ART coincidence.

The architecture of the ART processing core follows that of a pipelined design, where each stage performs one particular task. An overview of the sub-components is provided in Figure 6.13. The *ART Shift Registers* deserialize the ART stream, and buffer them temporarily. The VMM outputs the ART data using a dedicated pin, which

sends the data out of the chip and the front-end board (MMFE8 in this implementation) at $320\,\mathrm{Mb/s}$ rate. The VMM uses a $160\,\mathrm{MHz}$ clock *(CKART)* to clock the data out, which is provided by the VSB. The ART trigger primitive is constituted of a flag and the channel's six-bit address. Upon the deserialization of a flag and address, the ART is timestamped using an internal BCID counter, that increments at each rising edge of the system clock, similarly to the VMM's BCID counter. The address and timestamp combination enters a shift register, clocked by the CKBC, with five positions. There is one shift register per front-end VMM connection. If an ART exists in each of these three registers, they will be combined into one $18-$bit address combination word, which is propagated into a network of FIFOs, alongside their BCIDs. In essence, this component performs a time cut of $125\,\mathrm{ns}$. If a muon or pion traverses the SM2, the ART primitives will not be generated at the same bunch-crossing across the layers, but because of in-layer effects (e.g. diffusion) they will be spread in time. Also, the timestamping of each address is imperative, as it will be used later on in the process to issue the L0 signal in a timely fashion.

The $54-$bit bus containing the ART and timestamp information enters a network of FIFOs, where an arbitration logic queries each buffer and forwards the data from non-empty buffers to the next stage accordingly. If the ART address combinations differ with respect to their timestamps, then only one BCID is chosen, after a simple calculation of their mean value. Then, the $18-$bit ART combination is checked against a Look Up Table, implemented in the FPGA's blockRAM primitives. The RAM is loaded via a bus that is connected to the *UDPdin_handler* (see Figure 6.12, although that particular connection is not depicted there). The RAM has a depth of $2^{18}-1$ positions, and a width of one bit. It yields valid ART address combinations, as written to it by the user which can send this info to the RAM via UDP. The processor is designed to probe for vertical particle tracks. Hence, a valid ART address combination would be $0\,,0\,,0$, or $10\,,10\,,10$, or $42\,,43\,,41$[15]. Each of these decimal numbers is represented by a $6-$bit string. Three appended ART addresses, represent a *RAM's address*, and in that address a value of *one* is written, if that is a valid ART combination. Upon the arrival of an address-combination-to-check, the $18-$bit word is used as an address to the RAM's port, to read a high (valid track) or low (non-valid track) bit that is stored there. If a low bit is read, no further action is taken. If a high bit is read, the BCID of the valid track-like ART combination is forwarded to the next pipeline stage.

The final stage is the *L0 Issuer*, that receives the BCID and stores it into a FIFO. That FIFO is read and then the logic waits. It waits for a particular amount of time, that depends on the VMM's configuration (i.e. its *L0 Latency* setting) and the total latency

[15]The address value difference tolerance is controlled by the user, which may either be more, or less restrictive with respect to what constitutes a track or not.

of the triggering scheme. The user fine-tunes both the VMM's L0 latency, and the ART processor's, so that the *L0* signal that will be sent to the front-ends via the VSB, actually matches the data that are related with that ART address combination. In modest-occupancy situations, the latency between the reception of the first ART and the writing of the timestamp to the L0 Issuer's FIFO is 650 ns. Given that, and the time it takes for the ART addresses to reach the VSB, and the time it takes for the L0 trigger issued by the ART processor to reach the front-ends, a compatible L0 latency combination for both the VMM and the ART processor can be chosen. However, the VMM's L0 latency setting should *not* be equal to the sum of the intrinsic latency of the system. This is because in higher-occupancy situations, it is possible that the aforementioned latency will be larger than 650 ns. Hence the existence of the buffer in the L0 Issuer, which acts as a *derandomizer*, and in conjunction with the core's latency setting, issues a well-timed L0 trigger to the front-ends, and extracts the data that is supposed to.

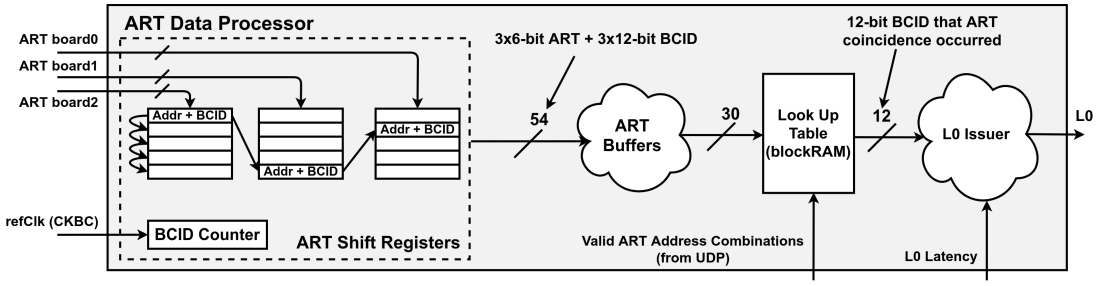


Figure 6.13: The *ART Data Processor* architecture. It receives data from up to three VMMs, connected to successive layers of the SM2. If the ART addresses from all three VMMs resemble a track, then an *L0* trigger is issued back to the VMMs of the system, thus resulting in a self-triggering scheme.

Note that eight parallel implementations of the processor are deployed in the VSB. This is because each front-end board has eight VMMs. Each miniSAS cable that carries the ART primitives from the front-end to the VSB, propagate the addresses from all eight ASICs that are on the MMFE8 front-end board. Hence, each ART processor implementation probes three VMMs that are, in essence, on top of one another, and the whole system is evaluating the trigger data from all $64 \times 8 = 512$ channels of each layer. Each processor's L0 output, drives an OR gate with eight inputs, that in turn forwards the ART coincidence signal to the *Trigger Module*, which will relay the L0 trigger to the front-ends accordingly.

Prior to the testbeam, the logic was successfully validated using test-pulses, and as it will become evident in the next Subsection, the beam data also verified its core functionalities.

6.3.3 Analysis - Results

The general goal of the July 2018 testbeam was to validate the SM2's performance, while being read-out by the third version of the VMM front-end ASIC. In particular, the operational parameters of both the detector and the ASIC needed to be defined. One particular issue that had to be addressed, was the high voltage setting applied to the strips of each detector layer. As discussed in Section 2.1, the higher the voltage delivered to the strips, the higher the detector's gain, as the avalanche effect gets more amplified, leading to higher efficiencies. However, too high-a-setting may lead to sparking, a phenomenon that is even more pronounced in large-scale Micromegas modules. Therefore, one important parameter that had to be defined precisely, was the optimal high voltage setting that would mitigate any sparking effects, while maintaining a satisfactory efficiency. Also, different gas mixtures besides the standard 93% Ar : 7% CO_2 had to be tried, in order to determine their impact in detector performance. Finally, the resolution of the SM2 had to be measured. All of the aforementioned studies were performed, with the VRS DAQ playing a catalytic role in the data acquiring and handling process. More than 500 runs under various parameters were taken, amounting to about 100 GB worth of data. With a per-channel occupancy of around 5 kHz, and trigger rates of under 100 kHz for pion and muon beams, the readout efficiency was always above 80% (see Figure 6.7).

One of the most important measurements was the per-detector-layer *efficiency* evaluation. Using the four T-chamber trackers, a track is formed from the clusters formed in each detector. Then, the track is extrapolated to the SM2, and a cluster is probed for, in each of its four layers. Failure to find a cluster within $10\,\sigma$ (where σ is the detector's resolution for vertical tracks, which equals to about 70 μm) of the particle's trajectory, reduces the said layer's efficiency grade. In Figure 6.14, a plot depicting the SM2's layer efficiency as a function of the high voltage applied is displayed.

Another parameter that had to be evaluated, was the detector's *spatial resolution*. In order to fulfill the ATLAS muon spectrometer requirements, the Micromegas should be able to provide hit reconstruction with accuracy better than 100 μm per detection plane [7]. As mentioned in Subsection 6.1.3, the techniques applied to the analysis of track reconstruction differ, and depend on the beam's angle with respect to the Micromegas readout plane. These methodologies have been studied extensively in previous works [3, 4, 55]. One of the most prominent ones are the Centroid Method (mainly used for *perpendicular* tracks), and the μTPC Method (mainly used for *angled* tracks). On the centroid (also called *center of gravity method*) technique, one can also apply the so-called η *Correction* to add another term that optimizes the track reconstruction performance. Also, it has been shown [7] that applying a hybrid

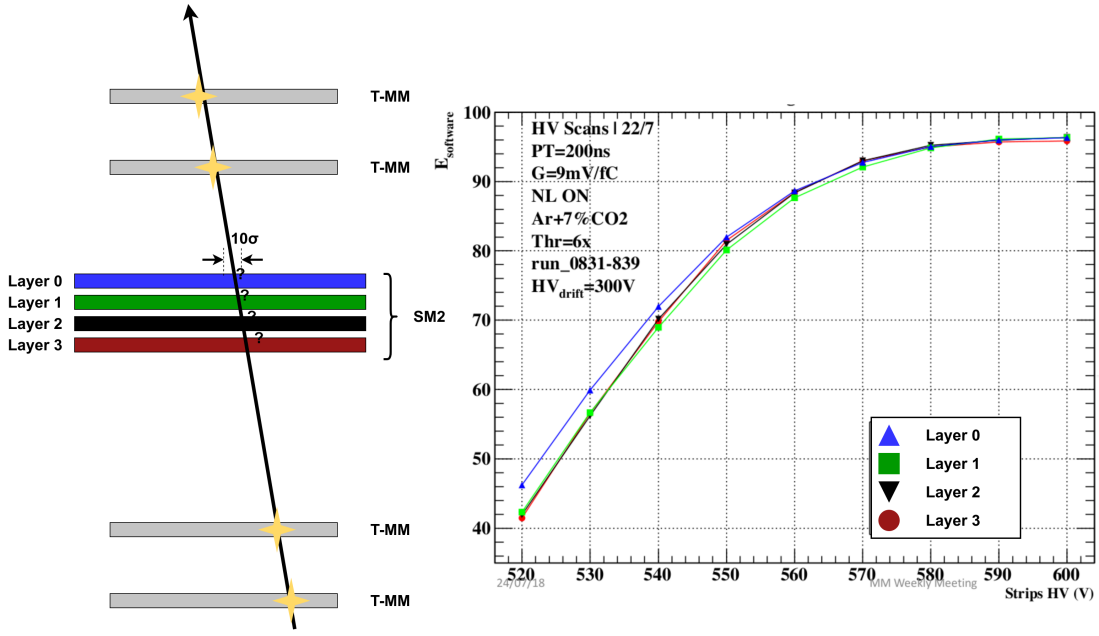


Figure 6.14: *Left:* The testbeam's setup when an SM2-layer efficiency measurement is performed. The particle's trajectory is reconstructed using the small T-chambers (denoted as *T-mm*), and clusters are probed for in the area of the SM2 layers that the particle passed from. *Right:* Efficiency of each layer of the SM2, for a set of standard VMM configuration parameters and detector gas mixture, as a function of the strip high voltage setting.

centroid-μTPC methodology for tracks within the range of $10° - 40°$, achieves more consistent results.

In this work, only the centroid method was performed to estimate the SM2's spatial resolution for the two eta planes. The repository of these C++ based routines that manipulate .ROOT ntuples can be found at [58], which is a branch of the codebase presented in a previous study [55]. In the case of gaseous detectors, a charged particle traversing the detector ionizes the active medium, thus creating a charge that is shared among *several* neighboring strips. This can be attributed to diffusion effects, that depends on the gas properties and the drift electric field, and to charge-sharing between capacitively coupled neighboring readout strips. In the resistive-strip Micromegas detector, the hit position reconstruction uncertainty is not only determined by the fine segmentation of the readout elements, but it is also significantly affected by the resistivity and the geometry of the resistive strip layer. These parameters affect the total induced charge on the readout strip per charged particle, and the number of strips with signal above threshold [4]. Therefore, it is considered standard practice to quantify the spatial resolution of all detector prototypes, to pinpoint any shortcomings (e.g. construction deficiencies) in the production process.

The centroid method revolves around one formula [3]:

$$P = \frac{\sum_{i=1}^{n} x_i q_i}{\sum_{i=1}^{n} q_i} \tag{6.1}$$

Where P is the *cluster's* centroid position per layer. Due to the phenomena referred to above, the charge is shared between adjacent strips, and the number of strips fired is n (for perpendicular tracks, usually $n = 3$.). x_i is the *strip's* centroid position, and q_i is the charge (i.e. the VMM's *PDO*) recorded by the strip. By subtracting the centroid position from successive eta layers over a large number of events, one gets the residual distribution, usually Gaussian in shape. The standard deviation of this plot represents the detector's spatial resolution. The results from a muon beam run, alongside a graphical representation of the centroid method is depicted in Figure 6.15.

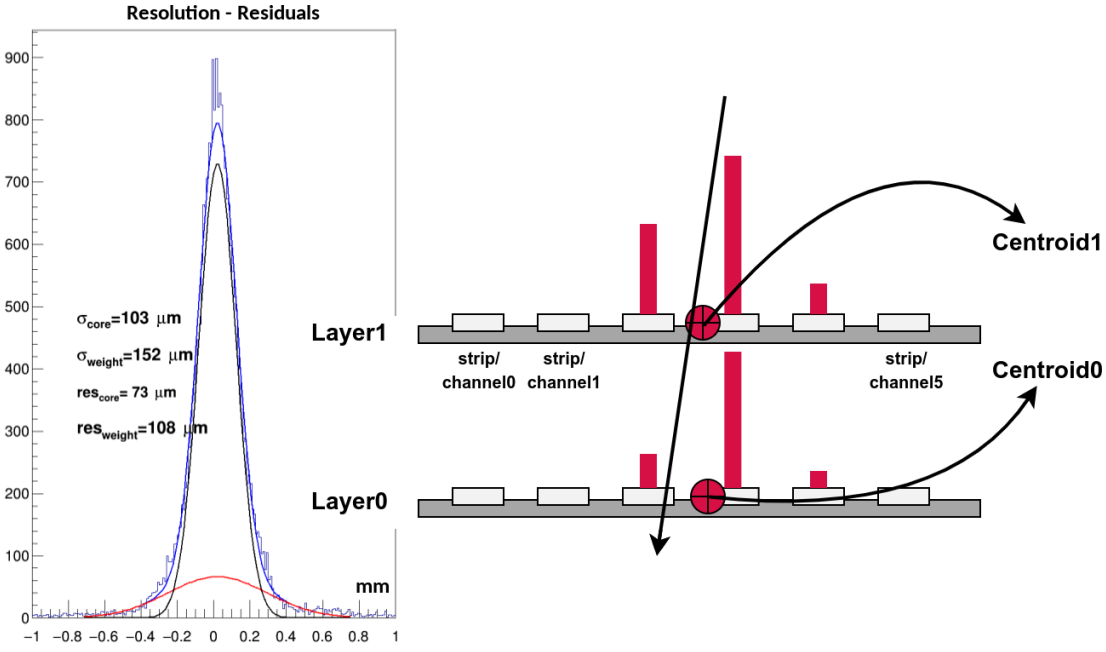


Figure 6.15: *Left:* SM2's residuals from one muon beam run during the July 2018 testbeam. The resolution equals to $\sigma = 73\,\mu$m, well within the detector's requirements. *Right:* Graphical representation of the centroid method for a near-perpendicular muon track traversing the detector. The centroid of the two layers are found, based on the charge distribution across adjacent strips in each corresponding layer (each centroid corresponds to P of Eq. 6.1). Subtracting the two centroids for one event, adds one entry to the left-hand-side plot.

Finally, the functionality of the ART processing core was evaluated. Unfortunately,

due to beam time and hardware limitations, no trigger algorithm efficiencies were performed. However, the basic logic of the VSB's trigger primitive processor was either way validated, by disregarding the scintillator coincidence input to the VSB, and inducing triggers to the front-ends via the ART from three MMFE8s mounted on two eta layers and one stereo layer of the SM2. The results were positive, as the DUT's resolution for muons was calculated to be $\sigma = 70$ μm, proving that the method is able to extract clean data from the detector. Finally, the beam profile in the self-triggered runs was created. The resulting beam profile is shown in Figure 6.16, where the profiles from a regular scintillator run and an ART run are compared. These preliminary results point that given more runs and optimizations to the ART processing logic, the VRS could evolve into an efficient self-triggered system.

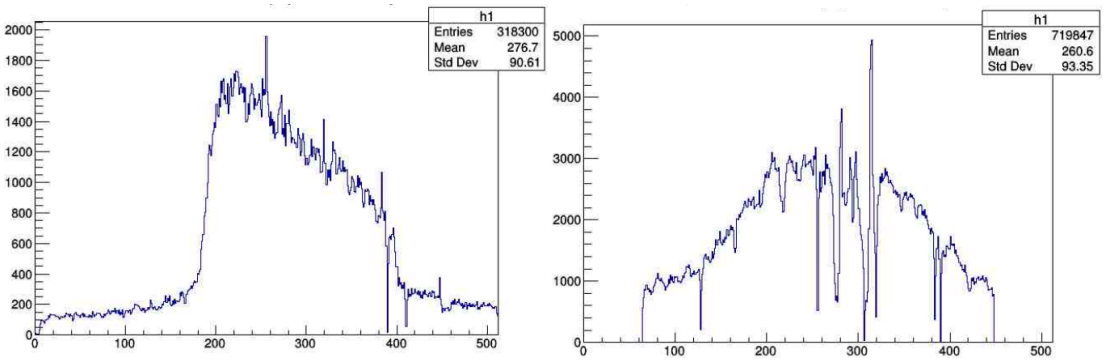


Figure 6.16: *Left:* Beam profile for one of the SM2's eta layers from a regular scintillator run. *Right:* Beam profile for one of the SM2's eta layers from a self-triggered run. In both plots, the x-axis represents the strip number, and the y-axis the number of strip hits. The difference in the beam's shape is evident, and stems from the fact that the self-triggering methodology has a *wider aperture* with respect to the scintillator triggering, because the latter scheme's readout region depends on the geometry of the triggering media. However, the right-hand side diagram presents some inefficient areas, which occur because some channels had to be masked in one of the layers that were used in the triggering. This creates a drop in efficiency, that can only be tackled if more layers are added in the triggering scheme of the firmware.

Conclusions

In this Chapter, two testbeam implementations of the FPGA-based VMM readout system were studied. The 2017 testbeam involved a small-scale experimental setup, that validated the VMM's third version chamber readout capabilities. For the 2018 testbeam where more readout nodes had to be included, the VMM Readout System was developed. The system involved several front-end nodes and an FPGA supervisory board, named VMM Readout System Supervisory Board, that orchestrated the DAQ procedure. Apart from this fundamental role, the VSB firmware is also able to

synchronize the inbound triggers with the global reference clock, and is also capable of processing the VMM's trigger primitives, in order to create a self-triggered system that does not rely on an independent scintillator stimulus. The scheme proved its efficiency by providing data to the NSW's Micromegas collaboration that played a significant role in the development of the Micromegas detector and the VMM.

For more information, the reader is also referred to the VMM Readout System journal publication [44].

Chapter 7

Integrating the New Small Wheel Electronics System with the ATLAS DAQ

In this Chapter, the final Data AcQuisition (DAQ) and electronics system of the New Small Wheel will be described, alongside a set of tools that were developed to ease the integration process. In Chapter 2, the cornerstone of the NSW electronics system was introduced: the VMM. In previous Chapters of the second part of this dissertation, the VMM was read-out using configurable integrated circuits (FPGAs). In the final system that will deployed in the ATLAS cavern though, the VMM will be interfacing with a set of other Application-Specific Integrated Circuits (ASICs), that will configure it, read-out its tracking (or *Level-0/1*) data, and acquire its trigger primitives. Prior to their commissioning, the final Micromegas sectors (see Chapters 2 and 6) are read-out and validated under cosmic radiation at the BB5 site of CERN. Hence, that particular setup will be described in greater detail, since it resembles the final scheme in the ATLAS cavern, but at a smaller scale. After establishing the necessary knowledge on the system's building blocks and functionalities, a description of a set of tools that aid the commissioning procedure will be provided, some of which will also be used in the final system which will read-out the NSW electronics and detectors in the ATLAS cavern.

7.1 ATLAS DAQ and the Front-End Link eXchange

As mentioned in Chapter 1, the future LHC upgrades will increase the beam's luminosity, which will consequently increase the particle flux, and hence, the demand in throughput of the detector's data acquisition system. As discussed in previous Chapters, the front-end data acquisition hardware is buffering data continuously, and is agnostic of their origin. Only a fraction of this information corresponds to meaningful physics data. Therefore, the actual trigger rate in ATLAS is smaller than the bunch-crossing frequency. In Figure 2.7, the general philosophy behind the final readout of the NSW's front-end ASIC (the VMM) is illustrated. The VMM (alongside many of ATLAS' front-end ASICs) buffers tracking data into its memory, and continuously streams rapidly-produced trigger data to the associated electronics of ATLAS. These combine trigger information from different sources across ATLAS, and produce the so-called *L0 Trigger* or simply *L0* that is fanned-out to all front-end nodes. If the latency between the event and the L0 signal associated with is *fixed*, as it should be, each front-end node will go through its memory, and send out the more accurate data (i.e. the *Level-0* or *tracking* data) associated with the said event. These L0 data are buffered in another front-end readout device, that forwards the data towards the back-end either immediately, or upon the reception of the so-called *L1 Trigger*, or simply *L1*. This signal will be the second stage hardware trigger in ATLAS, and represents a result of a more refined processing of the trigger data from the detector's electronics. Hence, the Level-1 trigger's frequency will be less than that of the Level-0's, and its latency will be larger than that of the Level-0's. Hereinafter, the two trigger stages will be considered as merged, and the terms L0 and L1 will be interchangeable, as the introduction of the two-stage triggering is something that only future runtime conditions will determine[1].

To be precise, the HL-LHC which is expected to start operations in the middle of 2026, will ultimately reach a peak instantaneous luminosity that corresponds to approximately 200 inelastic proton-proton collisions per 25 ns. That period, also referred to as *Phase-II*, will set the ultimate limit of the DAQ system, as the Level-0 trigger rate will be at 1 MHz, and its latency will be equal to 10 μs. For *Phase-I* which will take place between 2021-2023 on the other hand, the rate will be at 100 kHz, and the latency equal to 2.5 μs. Finally, it is worth mentioning that these trigger frequencies do not represent the rate at which data are stored in ATLAS. After the back-end electronics, the *High Level Trigger (HLT)* system performs a software-based trigger cut to the inbound tracking data, which further reduces the final storing frequency to 1 kHz for Phase-I,

[1] The two-level hardware trigger design specifies a Level-0 trigger rate of $2-4$ MHz at a 10 μs latency, followed by a Level-1 trigger, with a rate of $600 - 800$ kHz and a latency of up to 35 μs [8].

corresponding to an average output bandwidth at peak luminosity of $\sim 3.2\,\mathrm{GB/s}$, while for Phase-II, these numbers are expected to increase by about one order of magnitude [59].

The keystone of the ATLAS DAQ system will be the *Front-End Link eXchange* or simply *FELIX* [60–62]. FELIX will essentially be a bridge between the front-end electronics of all ATLAS detector subsystems, and their corresponding back-end components, which will mostly be software-based, whereas FELIX is an FPGA-based system housed by a commercial server. Situated in the counting room, or Underground Service Area 15 (USA15), FELIX connects to the front-end electronics of the ATLAS cavern via optical links, or *GBT links*, each one of which is running at $4.8\,\mathrm{Gb/s}$. For the NSW case, FELIX will interface with the front-end nodes over twenty-four optical links. These links carry the GBT frame, which is 84-bit wide, depicted in Figure 7.1. The Giga-Bit Transceiver (GBT) [63,64] protocol is a transmission scheme that involves radiation-tolerant ASICs, capable of handling large amounts of data for high energy physics experiments. It will be re-visited in Chapter 8.

Figure 7.1: Graphical representation of the GBT Frame [63]. The first four bits is the header, then the next four are dedicated fields used for Slow Control, while the Data can be added in an $80-$bit field. The FEC field is $32-$bit wide, and is used for correcting any errors during the reception of the frame.

One frame is transmitted from and towards FELIX at every bunch-crossing. The additional frame overhead depicted in Figure 7.1, results in an effective user bandwidth of $3.36\,\mathrm{Gb/s}$, since 84 bits can be transmitted each $\sim 25\,\mathrm{ns}$. The GBT interfacing is implemented in the FELIX FPGA, which deploys an equal number of instances of the GBT-FPGA firmware as the amount of links[2]. A large portion of the FELIX firmware is handling the data that are being mediated back and forth through the GBT-FPGA transceivers, and encode or decode these data either via the 8b10b or HDLC line codes. The other large part of the design, is the Direct Memory Access (DMA) core, which in essence implements the interface between the FPGA and the custom software that runs in the server's Central Processing Unit (CPU). This is done via standard Peripheral Component Interconnect Express (PCIe). The FELIX software connects to the back-end commodity network via $100\,\mathrm{Gb/s}$ Ethernet.

Three types of data are handled by the custom router which comes in the form of FELIX: L1 or tracking data, slow control data, and Trigger, Timing and Control (TTC) data [57].

[2]The GBT-FPGA is an FPGA implementation of the GBTx transceiver ASIC, part of the GBT chipset [65].

The first two are managed by both the software and the FPGA, but the latter, which must have a deterministic latency, passes only through the FPGA, via a dedicated optical interface. FELIX's board recovers the LHC clock[3] from the said TTC link, and receives all trigger commands, that are forwarded to the front-end devices with minimal and fixed latency. These commands include the aforementioned L0/L1 triggers, and various kinds of resets for the electronics. On the non-deterministic latency path, which goes through the server's Ethernet interface and the card's DMA connection, there are two major network clients connected with FELIX. Each one is associated with either the L1, or the slow control data. The new multi-threaded Software Readout Driver (swROD) infrastructure, receives the tracking data from FELIX, for event fragment building and buffering. In addition, the swROD supports detector specific data processing, and serves the data, upon request, to the ATLAS HLT [61]. On the other hand, a dedicated Open Platform Communications (OPC) server manages the slow control data that run through FELIX, and is primarily used to configure and monitor the front-end devices that FELIX is connected with [66]. The general scheme of FELIX, and its relationship with all surrounding systems is depicted in Figure 7.2.

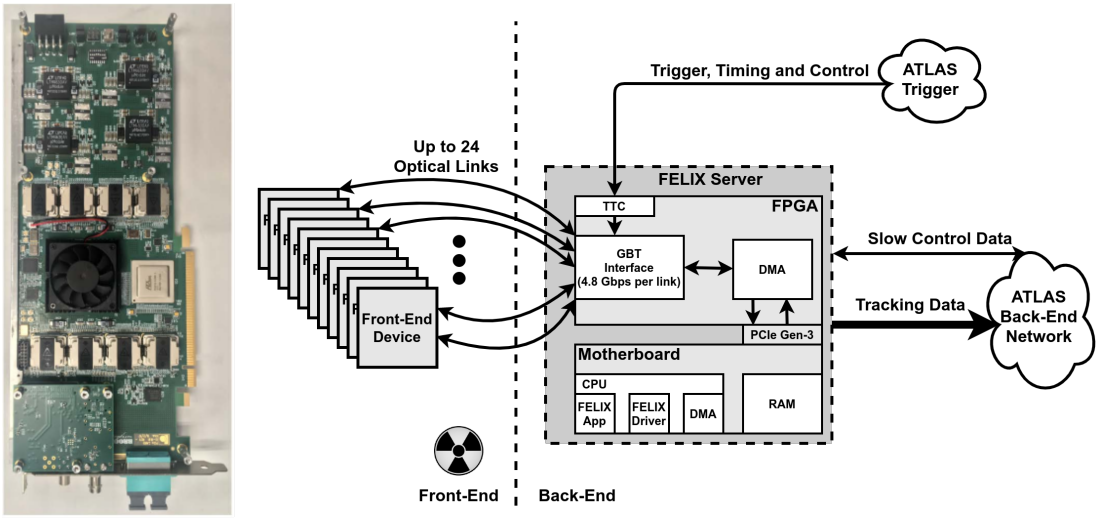


Figure 7.2: *Left:* The FELIX BNL712 FPGA board, which the NSW will use for its needs. It features a Xilinx® Kintex Ultrascale XCKU115 FPGA, a PCIe connector to interface with the CPU and the back-end network, and an optical coupler supporting up to 24 links. A dedicated mezzanine board receives the reference clock and the trigger information from the ATLAS TTC system [67]. *Right:* FELIX and its relationship with other parts of the DAQ system, as described in the main text of this Chapter.

In the Section that follows, more information will be given on the general scheme of the

[3] It is reminded that this clock has a frequency of 40.079 MHz, which is the frequency that the protons collide in ATLAS, and is the reference clock of the entire DAQ system.

NSW's electronics system, which is comprised of several devices, with some already mentioned in this dissertation.

7.2 The New Small Wheel Electronics

Single-Event Upsets (SEUs) that may occur in high radiation environments, such as inter-planetary space, or in high energy physics experiments, pose a significant challenge to all devices that are designed to be used in the aforementioned fields of applications. All of the elements that are used in the front-end electronics system of the NSW for example, have been designed in such a manner as to be practically immune from the negative effects caused by the radiation that is present in the NSW. Due to the proton-proton collisions occurring in ATLAS, high-energy neutrons may be generated by secondary interactions with the detector's materials. When a high energy neutron traverses an electronic device and strikes an atom in its silicon substrate, a reaction occurs. Energetic and positively charged reaction products liberate a cloud of electron-hole pairs. This mechanism is depicted in Figure 7.3. The electron-hole pairs create a Single-Event Transient (SET) in the circuit, which can propagate through combinational logic. If the SET has enough amplitude and duration and coincides with a clock, then an incorrect value is registered and the SET becomes an SEU [68, 69]. Choosing a transistor technology that is not that easily affected by the said effect, or triplicating[4] the registers of a digital device in order to introduce redundancy in case an SEU occurs, are common techniques employed by chips that are designed to operate in these types of conditions. All of the NSW ASICs are considered radiation-tolerant, meaning that their design guarantees error-free operation, despite the fact that for the innermost region of the detector's wedges the total ionizing dose can reach up to 5 krad per year of operation [7].

As stated in the beginning of this Chapter, the VMM is where all of the NSW's data are generated. The VMM (see Section 2.2) serves both detector technologies of the NSW, and continuously buffers L0 data into its memories. Each of the 64 VMM channels may connect to the Micromegas readout strips, the sTGC pads, wires, or strips, and adequately process all types of signals that originate from these elements, mainly because of its large set of configurable parameters and general flexibility. The L0 data that are associated with muon particles stimulating the detector media, are being selected and read-out by another ASIC, situated on the same front-end board that the VMM is on. This is the *Readout Controller* or *ROC*. More information on this device can be found in Subsection 7.2.2. The ROC manages the data from several VMMs

[4] Also referred to as Triple Modular Redundancy (TMR).

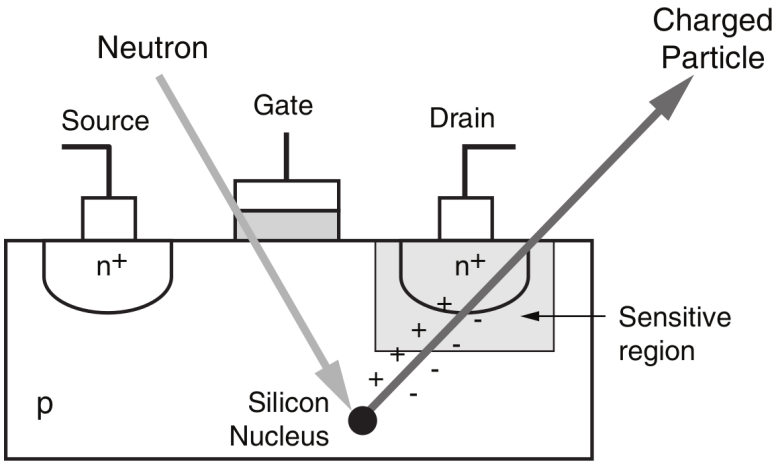


Figure 7.3: A neutron strikes the substrate of a transistor, thus releasing a charged particle that may alter the transistor's state [69].

(since in each front-end board there are usually more than one VMMs, but only one ROC) and buffers them temporarily, before sending them out of the front-end board it is on, upon the reception of the L1 trigger. The L1 data from several ROCs are aggregated by another ASIC, called *GigaBit Transceiver* or *GBTx* [63, 64], which will be re-visited in Subsection 7.2.1 and Chapter 8. In essence, the GBTx is a high-speed transceiver, capable of connecting to many front-end devices (including the ROC) via serial electrical links called *E-links*. The GBTx is the last stage before the back-end electronics. It connects to FELIX via optical fiber, and receives the reference clock, and the slow control and trigger data from it, and distributes it to the other front-end devices it is connected to via their associated E-links, while it also sends the L1 and slow control data towards FELIX, as it receives them from the other front-end devices via their associated E-links. The Slow Control Adapter (SCA) ASIC, which will also be re-visited in Chapter 8, is the chip that handles the configuration and monitoring of all other front-end devices. It allows for interfacing with several types of ASICs (or sometimes FPGAs), via various sub-devices it employs in its fabric (i.e. I^2C , SPI, JTAG primary modules, or ADCs). It therefore receives the slow control data from FELIX via the GBTx, while it also sends monitoring data and configuration replies via the GBTx back to FELIX. The front-end board that hosts eight VMMs, one ROC and one SCA for the Micromegas technology is the *Micromegas Front-End Board 8*, or *MMFE8*. The board that hosts three GBTxs and an SCA, is the *Level-1 Data Driver Card*, or *L1DDC* [8]. This more or less completes the description of the L1 data path, which is common for both detector technologies.

But the VMM also operates at a faster mode, used for triggering (see Section 2.2). Upon the detection of a peak that crosses the user-defined threshold, the chip does not

only buffer the high-precision-measurement data into its memories, but it also outputs the channel address that this peak was detected on. This is called the Address-in-Real-Time (ART) mode and is used in the Micromegas trigger scheme. The VMM also has a fast digitization 6-bit ADC (6bADC), that provides a coarse, but rapid, amplitude measurement of the pulse encoded in 6-bits. In addition, as long as a pulse is above the threshold, the VMM outputs a single-bit Time-over-Threshold (ToT) flag in one of its 64 direct-timing output pins. The 6bADC data and ToT signals are used by the sTGC trigger electronics to determine trigger candidates. The VMMs on the pFEB operate in the ToT mode and the sFEB VMMs operate in the 6bADC mode. One of the main elements of the sTGC trigger electronics is the Trigger Data Serializer (TDS) ASIC [70]. There are two types of TDS ASICs, one for the pFEB and one for the sFEB. The pFEB TDS collects the ToT signals from all VMMs and timestamps them with the BCID. The sFEB TDS deserializes the 6bADC data from the VMMs on the board and buffers them temporarily. Connected to both FEBs is the Pad Trigger (PT) board, an FPGA-based processor that uses the ToT signals, that give a rough estimate of a muon event position, to create a Region-of-Interest (RoI). A continuous stream of ToT flags with their timestamps are sent by multiple pFEB TDSs to the PT and if a 3-out-of-4 coincidence is found by the PT in both sTGC quadruplets of a wedge, a RoI is formed. The PT then signals the sFEB TDS chips within that RoI to send out their buffered 6bADC data towards the sTGC Trigger Processor via the Router Board. This scheme, though complex, reduces throughput towards the sTGC Trigger Processor without compromising the precision. On the Micromegas trigger path side, only the ART mode is being used, resulting in a simpler scheme. The ART signals from the VMM are driven to the ART Data-Driver Card (ADDC) that connects with up to eight MMFE8s (thus, 64 VMMs). The ART ASIC on the ADDC deserializes the ART addresses from several VMMs, appends a 5-bit VMM geographical address and a BCID timestamp. It then forms and forwards a data packet to the Micromegas Trigger Processor (TP) through optical fiber. On the back-end of the trigger path, lie the NSW TPs. The NSW TPs are FPGA-based, housed on Advanced Telecommunications Computing Architecture (ATCA) crates and collect trigger primitives from the ADDC and sFEB/PT/Router boards. The TPs quickly calculate centroids using the ART and 6bADC data, interface with each other to remove duplicates and transmit muon trigger segments/candidates to the Sector Logic (SL) in just under $1025\,\text{ns}$, within the requirement set to the NSW trigger chain by ATLAS [7]. The SL then combines information from the muon subsystem's Big Wheel, to generate a final trigger candidate decision and eventually send it to the ATLAS *Central Trigger Processor (CTP)*, which issues the final L0 trigger via FELIX accordingly. The entire scheme of the NSW electronics is illustrated in Figure 7.16 at the end of this Chapter, while the VMM's connectivity with all other front-end ASICs is depicted in Figure 7.4. It is also reminded

that the current Chapter focuses more on the L1 DAQ chain of the Micromegas detector, therefore the trigger scheme and the L1 DAQ sTGC scheme will not be covered in more detail, as they fall out of scope with the work related to this dissertation.

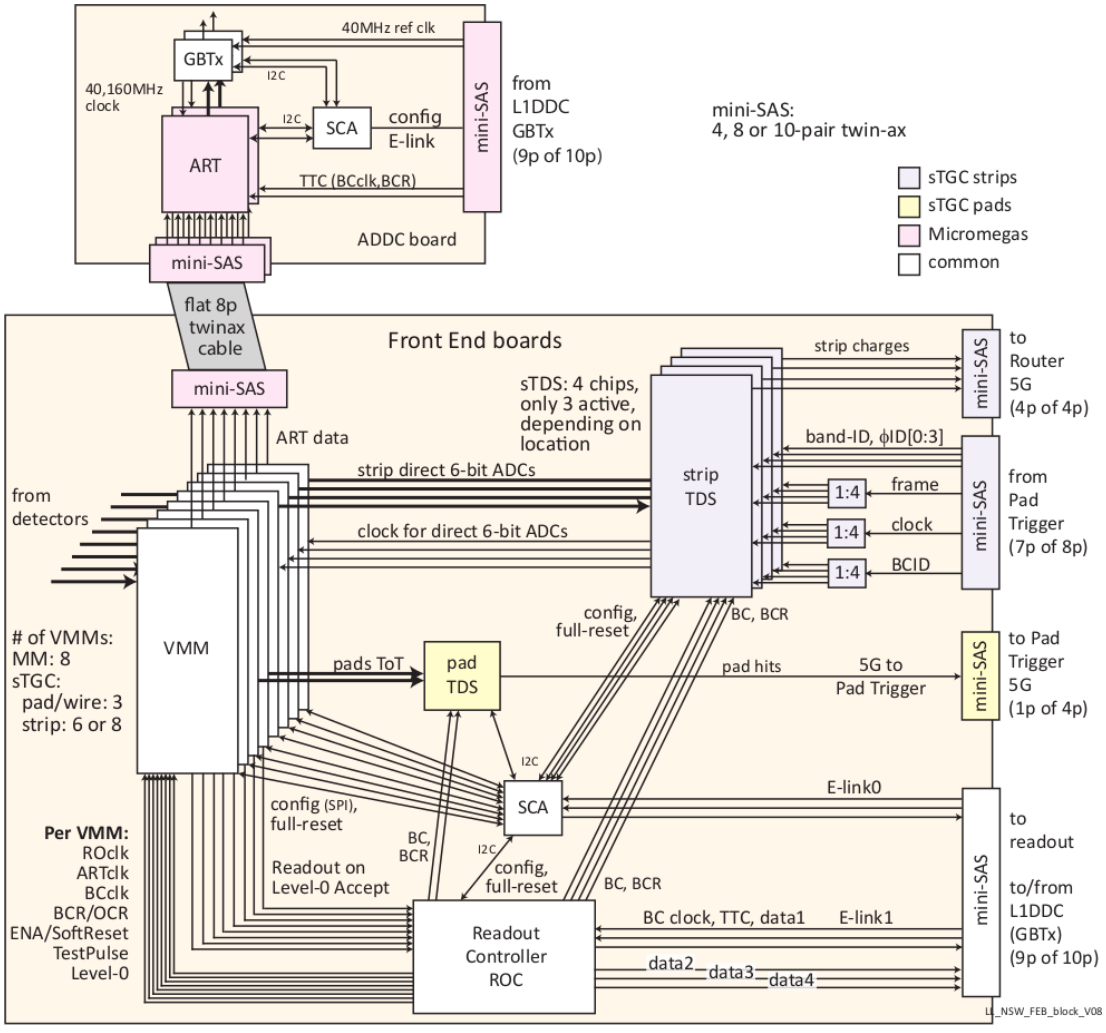


Figure 7.4: The VMM's connectivity with all other nodes of the front-end trigger and L1 readout system of the NSW [50].

7.2.1 The GigaBit Transceiver ASIC

The GigaBit Transceiver (GBTx) ASIC, is a radiation-tolerant transceiver developed by CERN [63,64], specialized in mediating data in the harsh radiation environment of high energy physics experiments. It is part of the GBT chipset, which is comprised of the GBTx ASIC, the Versatile Optical Transceiver (VTRX) (formed by the Gigabit Laser Driver (GBLD) [71], and the GigaBit Trans-impedance Amplifier (GBTIA) [72]), and

the SCA [43]. The GBTx implements a bidirectional optical link, which connects it with the back-end part of the DAQ system, at a bandwidth of 4.8 Gb/s. Through this link, three types of data are mediated: *Timing and Trigger*, *L1 Data* or *Tracking Data*, and *Slow Control Data*. For the NSW case, at each bunch-crossing (i.e. every 25 ns) each FELIX's optical link sends to and receives from its GBTx a frame which is 84-bit wide, depicted in Figure 7.1. On the GBTx side, the physical link is implemented by the two auxiliary radiation-tolerant ASICs, the GBLD and GBTIA, which connect with the GBTx. On the FPGA side, FELIX's GBT-FPGA block that is implemented in its design, implements the actual interface with the GBTx chip that is on the other side of the connection. Hence, the block denoted as *Front-End Device* in Figure 7.2, is the GBTx, which is in turn connected to other devices.

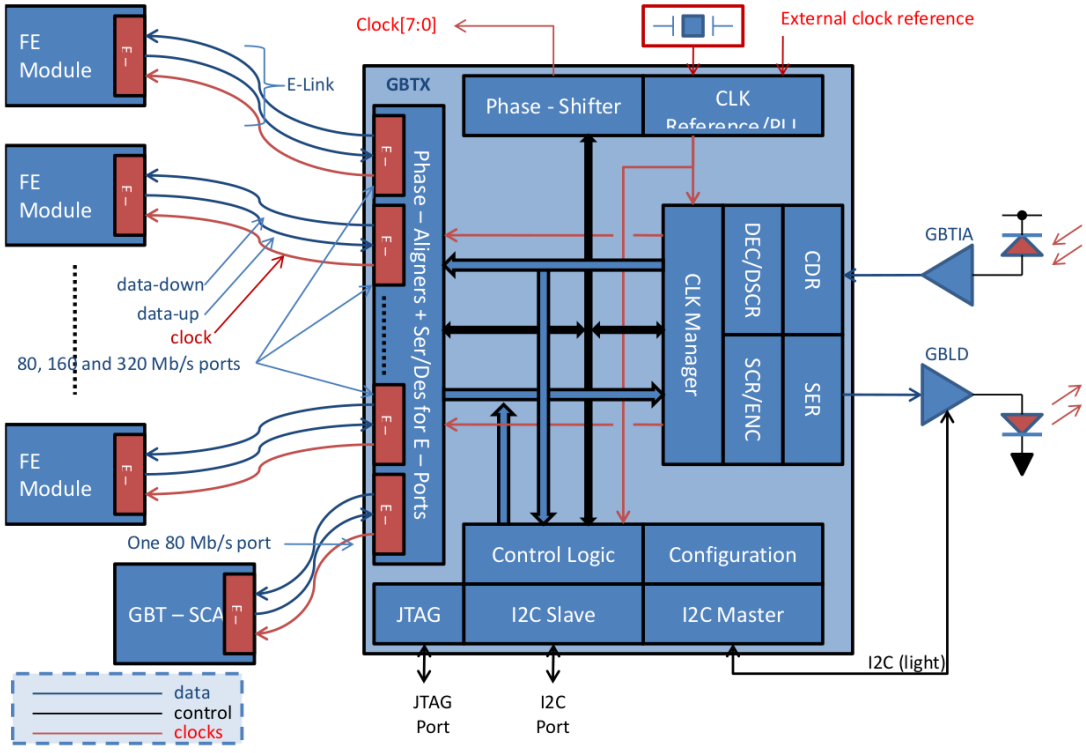


Figure 7.5: GBTx Architecture. The back-end optical interface on the right allows the ASIC to interface with FELIX, while the front-end uplink and downlink serial E-links implement the interface with the front-end devices [73].

The GBTx ASIC employs 130 nm technology, and triplicated configuration registers to protect itself from SEUs. It utilizes a 400-pin BGA package with 0.9 mm pitch, a single power supply of 1.5 V, and a maximum power consumption of 2.2 W. The GBTx recovers a reference clock from the fiber using an on-board oscillator, and uses that to synchronize its interfacing logic with all other front-end devices it is connected

with. That clock (which is the bunch-crossing clock) originates from the TTC system of ATLAS, and is being distributed to all of the NSW's FELIX instances via dedicated modules. FELIX in turn uses that clock as a reference for its GBT-FPGA transceivers, which implement the optical interface with the GBTx. The latter's device large number of pins, allow it to connect with many front-end devices, and provide that reference clock to them via differential pairs, thus resulting in an isochronous system. Apart from the clock, the GBTx receives the data from the $84-$bit frame that FELIX sends at each bunch-crossing, and serializes them to the connected devices via serial E-links, over differential pairs. These are the *downstream/downlink* connections. Each of these E-links may run at 80, 160, or $320\,\mathrm{Mb/s}$, and are in sync with the reference clock. Similarly, the GBTx may also receive data from the front-end devices, via another set of differential E-links, also capable of running at the aforementioned speeds. These are the *upstream/uplink* connections. After the deserialization, the upstream data are then mediated towards the back-end via the optical interface. A complete set of E-links for a given device connected with the GBTx is therefore comprised of a clock sent by the GBTx, the serial data sent by the GBTx (downlink), and the serial data sent to the GBTx (uplink). The ASIC's bandwidth and packaging allows it to connect with up to forty devices, if all of their E-links operate at $80\,\mathrm{Mb/s}$. Usually though, the ASIC is configured to interface with different devices at a variety of data rates, depending on the requirements of the system. A block diagram of the ASIC can be examined in Figure 7.5. More information on the GBTx's functionalities are provided in Section 7.4.1, where an application that manipulates its configuration registers during runtime conditions will be described.

7.2.2 The Read Out Controller ASIC

The Read Out Controller (ROC) is an ASIC designed to forward the L1/tracking data from the VMM chips towards FELIX via the GBTx, while accommodating the expected trigger rates in the ATLAS detector for both Phase-I and Phase-II. The ROC aggregates data from up to eight VMMs. The data from each VMM are serialized to the ROC via two lines, running at $320\,\mathrm{Mb/s}$ each, resulting in a total effective bandwidth of $512\,\mathrm{Mb/s}$ per VMM, due to the applied 8b10b encoding on the hit data transmitted from the VMM to the ROC [42]. As it is evident from Figure 7.6, the ROC has eight VMM L1 data deserializing and buffering instances working in parallel. Each one forwards the extracted data to one sub-ROC (or simply, *sROC*), which buffers the data in a deeper FIFO, and performs the BCID or L1 matching on them. The extra buffering stage exists because of the two-level trigger architecture of ATLAS. The VMM forwards its data to the ROC upon reception of the *L0* trigger, which the ROC sends to it, as it receives it by the TTC stream. Data that are matched within a specific time window, given a

configurable offset, are sent to the ROC[5]. The ROC in turn sends the data it read from the VMM to FELIX, upon the reception of the *L1* trigger by the TTC stream. This trigger is tagged with a BCID by the ROC, as the VMM does with the L0. Then, a configurable offset is applied to the L1 timestamp, and if any VMM data are matched within the sROC's buffer, they are being forwarded out of the sROC, towards the GBTx, via a serial upstream E-link, running at 320 Mb/s maximum, over 8b10b encoding[6]. For Phase-I, the L0 and L1 triggers will arrive to the ROC within the same bunch-crossing, meaning that all data which are read-out of the VMM are sent to FELIX, so in essence, no L1 matching is performed. If however, the two-stage trigger level is implemented in Phase-II, where the L1 trigger will arrive less frequently than the L0, the ROC will be performing the second stage of L1 matching and data selection, to decrease the load to the back-end DAQ network.

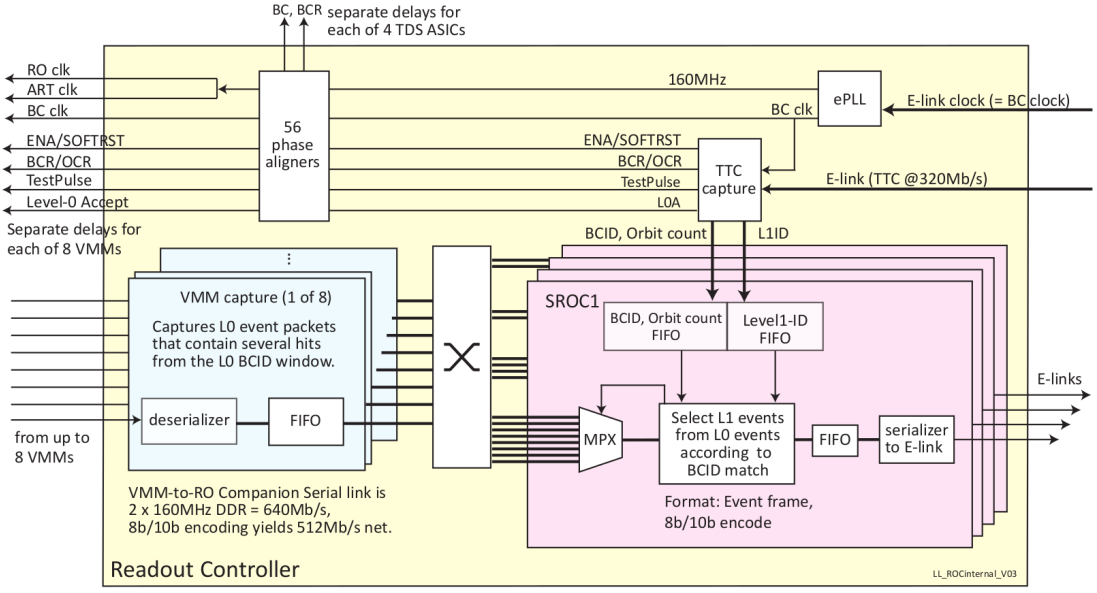


Figure 7.6: The Read Out Controller Architecture [8]. Eight deserializing instances capture the data from the VMM. All of them are connected in a configurable manner with the sROCs, via the so-called *crossbar*. Each sROC performs the L1 matching, and if VMM data are found within the BCID, they are serialized towards the back-end via the E-links. The ROC receives the 40.079 MHz bunch-crossing clock from the GBTx, and uses it as a reference for its PLL (labeled as *ePLL*), that clocks the ASIC's logic. This clock is also sent to the VMM, alongside the 160 MHz data clock. Four possible trigger signals are also relayed to the VMM. They are being received by the ROC over the inbound TTC stream that is distributed by the related system and FELIX, via the GBTx.

[5]This mechanism has been described in greater detail in Section 5.3, and depicted in Figures 5.6 and 6.6.

[6]For the Micromegas detector, all four sROCs will be utilized for Phase-II, resulting in a total effective bandwidth of 1.024 Gb/s.

The rest of the ROC's digital logic mainly comprises the TTC decoding module, which will be described in Subsection 7.4.2, since the tools that were developed to aid the commissioning process involve the said path. Finally, the ROC has an analog core that employs CERN's ePLL [74], to synthesize a copy of the bunch-crossing clock, as received by the GBTx, alongside a 160 MHz clock, used in a variety of processes inside the ASIC's logic.

7.2.3 The NSW Micromegas L1 Readout Front-End Boards

The two main boards involved in the L1 Readout of the NSW's Micromegas DAQ are the *Micromegas Front-End board 8 (MMFE8)*, and the *Level-1 Data Driver Card (L1DDC)*. The L1DDC acts as a data aggregating device that employs three GBTxs. One of these GBTxs, interfaces bidirectionally with the ASICs of eight MMFE8s, while the other two will be receiving the L1 data from the ROCs during Phase-II, when extra throughput will be needed. All GBTxs connect via optical fiber with FELIX. The MMFE8 is a front-end board that employs eight VMMs, reading-out 512 MM strips. Alongside the VMMs, there is one ROC that reads them out, and an SCA that configures both ASICs. One pair of uplink and downlink E-links running at 80 Mb/s, as well as another differential pair for the reference clock, are used for the bidirectional slow control connection of the SCA with the GBTx. The ROC is connected with the GBTx via one downlink TTC E-link running at 320 Mb/s, and up to four distinct uplink E-link pairs, one for each sROC, running at 320 Mb/s each. Finally, the ROC receives the reference clock from the GBTx. The total of nine differential pairs are packed in a twisted flat cable, called *twinax* cable, connected to the MMFE8 through its *miniSAS* connector. On the other end of that connector, lies an L1DDC. The MMFE8 has another connector of the same kind, that carries the ART stream from all eight VMMs to the trigger aggregating board, which is part of the trigger chain and will not be covered in more detail in this dissertation[7]. A photograph of the MMFE8 is shown in Figure 7.7. For the needs of the NSW, 4096 MMFE8s will be fabricated.

The L1DDC employs three GBTx chips. One of them, features a bidirectional optical link with FELIX. It thereby distributes the reference clock to eight SCAs and ROCs, via eight miniSAS connectors, on the same number of MMFE8s. It also distributes the reference clock to the ADDC board through one of its connectors. This GBTx also sends the TTC bits to eight ROCs. It also receives L1 data from some of the sROCs. For Phase-I, only this GBTx is used. For Phase-II though, where the throughput demand is foreseen to increase, the other two GBTxs will be used as well. These feature a

[7] This is the ART Data Driver Card (ADDC), that features a trigger primitive aggregating ASIC, and a GBTx that sends the accumulated data to the back-end MM Trigger Processor via optical fiber.

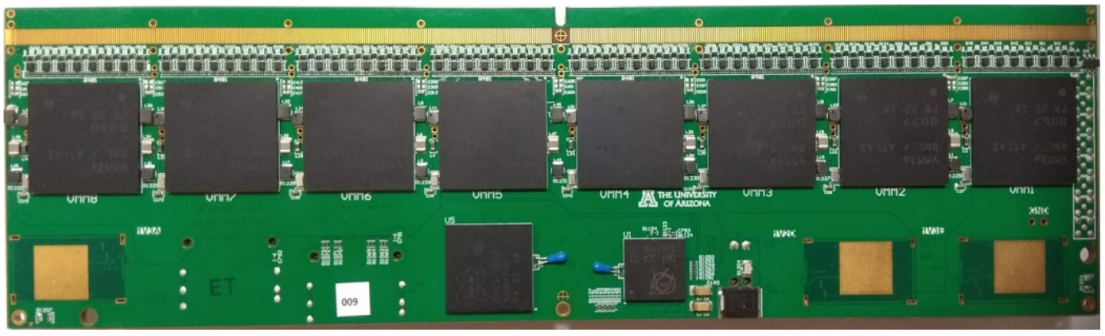


Figure 7.7: Photograph of the production NSW MMFE8 board. Designed in the University of Arizona, the board features eight VMMs (top row), one ROC (bottom-center), and one SCA (on the right-hand side of the ROC). The two miniSAS connectors are on the back side of the board which is not visible here.

unidirectional connection with FELIX. They receive the uplink data from the sROCs (the ones that were not being used for the Phase-I readout) that are connected with them, and send them to FELIX. All of these E-links will be running at 320 Mb/s. Therefore, for Phase-II, the L1DDC will utilize four fibers (one receiving data from FELIX and three transmitting to it) for the interface with the back-end, with the total throughput reaching up to 8.72 Gb/s (from a total of about 10 Gb/s of available user bandwidth). All 512 NSW L1DDCs will handle a rate of about 4.46 Tb/s [8]. A photograph of the L1DDC is shown in Figure 7.8.

7.3 The Micromegas Cosmic Stand at BB5 and its DAQ System

Beginning in 2019, during the LS2 period, the commissioning process started seeing a significant activity increase. An area in building 899 of the CERN Meyrin site, dubbed *BB5 Commissioning Site*, was dedicated in assembling the production Micromegas quadruplets arriving from the construction sites located in Russia, Italy, France and Greece. Each four-layer Micromegas quadruplet (see Subsection 2.1.1), is first instrumented with 64 MMFE8s, 8 L1DDCs and 8 ADDCs. The detector's services, such as the high voltage supply, low voltage supply via 8 custom distribution boards per quadruplet, gas supply, twinax cabling, fiber optics routing, and cooling supply, are also implemented, to ensure nominal operation. Two quadruplets are then combined into a double-wedge, or sector. The sector is then placed in the cosmic stand, where the detector's and electronics performance are validated under cosmic radiation. The total turnaround time for each sector is two weeks, after which the detector is shipped to building 191, for placement in the envelope with the corresponding sTGC sector,

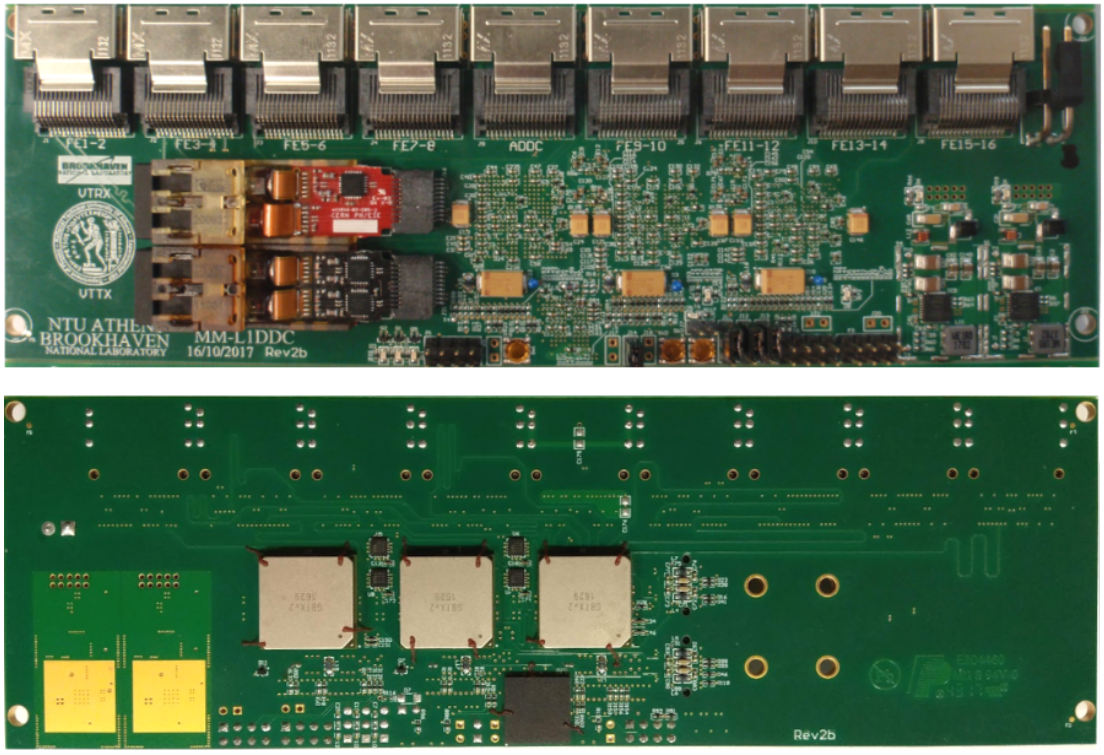


Figure 7.8: Photographs of the production NSW L1DDC board. *Top:* The top side of the board. Nine miniSAS connectors on the top row implement the interfacing with eight MMFE8s and one ADDC. The VTRX (red) implements the bidirectional connection of the first GBTx with FELIX, while the other two GBTxs use the VTTX (black) to send the L1 data to FELIX under Phase-II conditions. *Bottom:* The bottom side of the board. The three GBTxs are visible. Also, the SCA that performs environmental variable monitoring on the board, and the Phase-II GBTx configuration is on the very bottom [8].

ready to be deployed in the ATLAS cavern (see Figure 1.10). At the time of writing of this document, seven MM sectors had been successfully tested and sent to the final integration site. See Figure 7.9 for a picture showing the detector on the stand.

A set of four scintillators are placed on top of the sector-under-test, providing trigger signals from cosmic muons to the DAQ system. The system is comprised of 128 MMFE8s, 16 L1DDCs and 16 ADDCs, that read-out the entire double-wedge under a $\sim$ 80 Hz trigger rate. Naturally, the data are aggregated by FELIX, that interfaces with the GBTxs housed by the L1DDCs via 16 bidirectional fiber connections. For the Phase-II readout validation, an extra set of 32 unidirectional fiber links carry the load of the extra throughput to another FELIX card situated in the same server. The front-end electronics are configured and monitored via a dedicated OPC server, which handles the low-level traffic with all front-end SCAs (160 per sector) via FELIX (see also Section 8.1). The scintillator coincidence signal is injected into the system via the

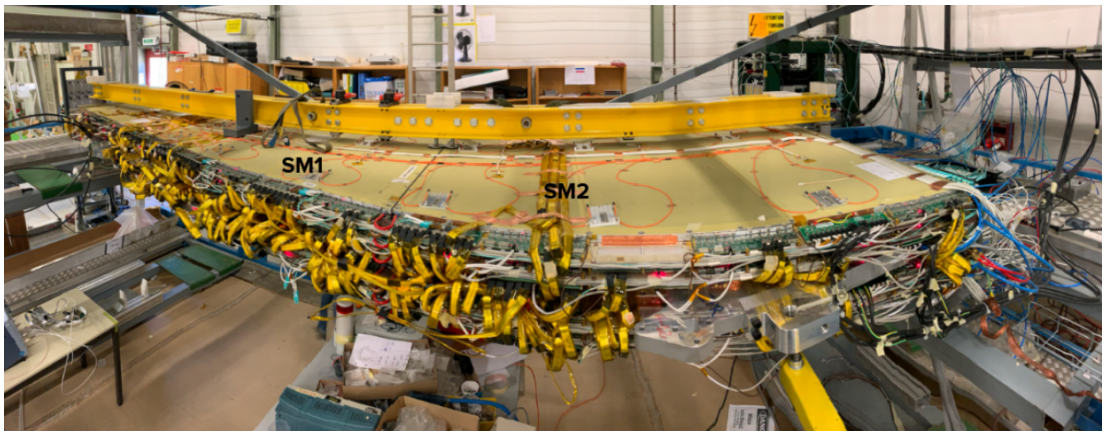


Figure 7.9: Photograph of a Micromegas sector while on the BB5 cosmic stand. Highlighted are the SM1 and SM2 parts of the detector. One can also see the front-end electronics mounted on the detector, and the twinax cabling connecting them together. Four scintillators are not visible here, but are situated on top of the detector medium.

ATLAS Local Trigger Interface (ALTI). The latter is a Versa Module Europa (VME) module, that can accept a raw trigger signal as an input, and convert it into an L0/L1 trigger, transmitted to FELIX over a unidirectional (TTC) fiber connection. The same module provides the 40.079 MHz bunch-crossing/reference clock to the system, while it also relays the necessary reset signals for synchronization. The user can load a pattern file to the device's memory, which includes a trigger bit sequence that is sent to FELIX via the fiber. FELIX in turn translates these into TTC bits, and fans them out to all front-end electronics. This, for instance, is how the VMMs are test-pulsed in the final system. FELIX recovers the clock from the TTC fiber input, and uses it as a reference for the transceivers that connect it with the GBTxs. The GBTxs in turn recover the clock from the fiber connection, and distribute it to all nodes further downstream. Finally, the primordial signals that are induced by the muons traversing the detector, are processed and converted into L1 data by the VMMs, and aggregated by the ROCs and the GBTxs, before being sent to FELIX. FELIX then sends the information to the swROD. The latter gathers the data from a commodity network over the NetIO interface [75]. Because the latency is not deterministic over that connection, the swROD makes use of the so-called *Level1ID* or *L1ID*[8] to group the events together based on their trigger sequence identification number. This is a similar scheme as the one described in Chapter 6, where the FPGAs of the DAQ system were tagging the data with a trigger identification number, so that the DAQ software could group them correctly. The swROD follows the same procedure, but it also makes use of a virtual FELIX E-link, which provides L1ID and Bunch-Crossing ID (BCID) data information.

[8]This is a trigger counter value appended to the data packets by the ROC.

That is, upon the reception of an L0/L1 trigger by the ALTI, FELIX increments an internal L1ID counter, timestamps the trigger with a BCID, and sends these values over that dedicated E-link to the swROD. The swROD then searches its buffers that contain the ROC packets for events with that same L1ID and BCID values, and builds the fragment once it has the corresponding packets from all ROC connections[9]. In the final implementation, the fragments will be sent to the HLT, but for the BB5 site, they are saved into data files for processing. A block diagram depicting all main building blocks of the system is depicted in Figure 7.10.

7.4 DAQ Tools Development - Detector and Electronics Commissioning

The DAQ system that reads-out the Micromegas sector at BB5 is, in essence, a smaller-scale version of the final system that will read-out the entire NSW. The goal of the BB5 setup is therefore not only to validate the performance of the detector itself, but to also make sure that its front-end electronics operate seamlessly in conjunction with the back-end infrastructure. Due to the difficulty of accessing various elements of the front-end electronics after the sector's installation to the cavern, it is imperative to pinpoint any points of failure as early as possible. Hence, one of the other objectives that must be fulfilled during the sector's 2-week tenure at the BB5 cosmic stand, is to ensure that all of its electronics are capable of running in both Phase-I and Phase-II readout configurations. For this reason, several tools that validate various critical paths of the L1 data path were developed. These paths can be seen in Figure 7.11.

In total, 65536 channels are being read-out by 1024 VMMs, which are uniformly distributed across the eight layers and eight PCBs of the sector (see Subsection 2.1.1). Each PCB is covered by 1024 resistive strips, and two MMFE8s connect to them in opposite sides. Each L1DDC serves one side of one layer, hence their total number of 16. When the system was first deployed in the summer of 2019, only eight MMFE8s were used, positioned on eight successive layers of the same side of the same PCB number, in order to be able to reconstruct muon tracks traversing the setup at that area. After scaling-up the system, multiple points of failure were found, that were attributed to timing violations. These include the TTC downlink path, from the GBTx to the ROC, and the L1 data uplink path, from the sROC to the GBTx. Both of these are validated using automated tools that will be described in later Subsections. After validating these critical connections, it was found that the electronics system would mediate the L1 data

[9] In principle, the swROD first receives the L1ID from the virtual FELIX E-link, and receives the data from the ROCs at a later stage.

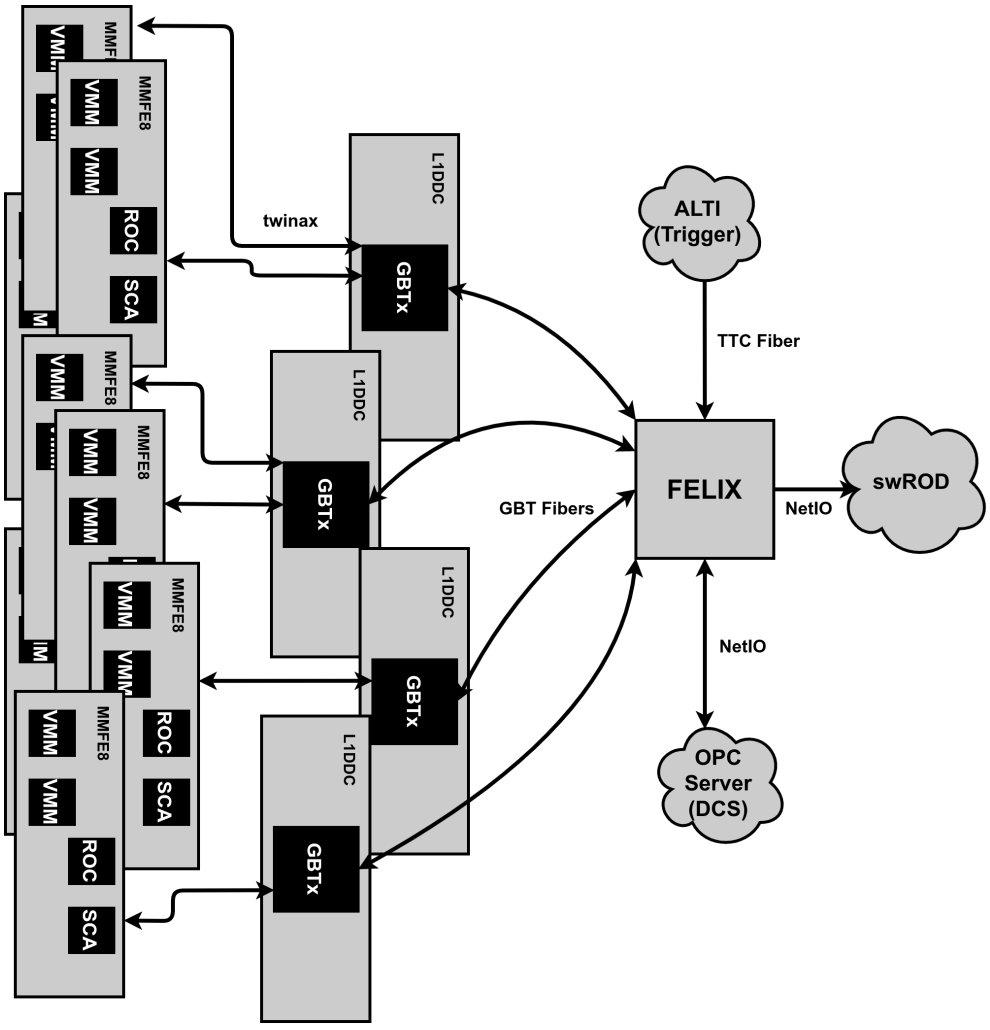


Figure 7.10: Block diagram of the different elements of the BB5 L1 DAQ system. 128 MMFE8s are connected to the MM readout strips across its eight PCBs and layers. 16 L1DDCs are connected with the MMFE8s (each connects to eight via serial E-links over the twinax cables), to distribute slow control, trigger and reference clocking, and aggregate the L1 data from them. Each L1DDC connects to FELIX via one bidirectional optical link (used in Phase-I), and two unidirectional optical links (used in Phase-II). FELIX connects with all 16 L1DDCs through 48 distinct transceiver (GBT-FPGA [65]) implementations. FELIX receives the slow control data that are sent to configure the front-end electronics from the Detector Control System (DCS), which implements a dedicated OPC server [66] to handle the traffic with the 160 SCAs. The scintillator trigger coincidence is sent to FELIX and the front-end ASICs via the ALTI module, while the tracking/L1 data are sent to the swROD for fragment assembly. The ADDCs which implement the trigger path are not depicted here.

smoothly, up to the swROD[10].

[10]The reader should also bear in mind that this is not the complete picture - given the fact that all back-

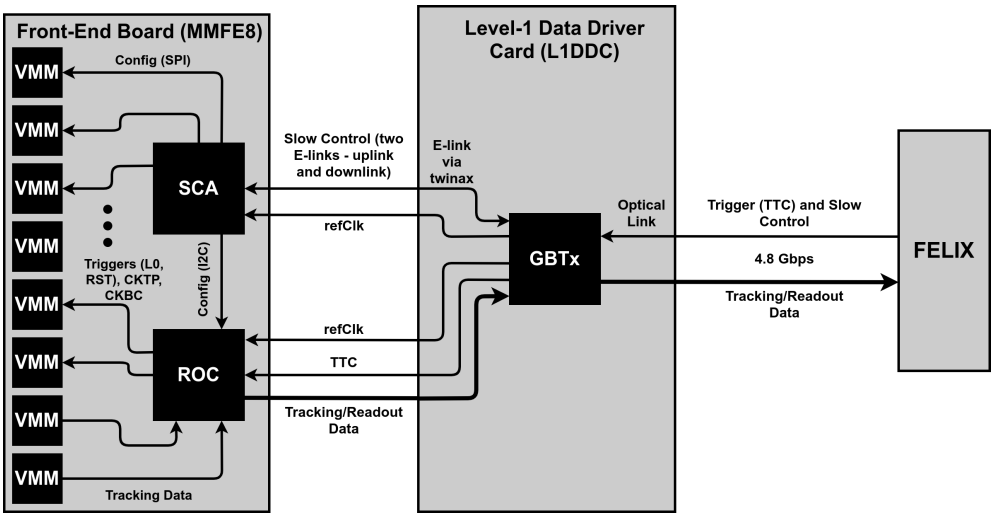


Figure 7.11: Breakdown of the Micromegas L1 readout data paths. On this picture, only one of the sixteen bidirectional FELIX optical links, and only one of the eight GBTx serial E-links via twinax are depicted. Two possible points of failure are: the TTC downlink path, from the GBTx to the ROC, and the L1 data uplink path, from the sROC to the GBTx.

7.4.1 The IC-Over-NetIO Application

The GBTx ASIC (see Subsection 7.2.1), is responsible for distributing the reference clocking, as recovered from the FELIX optical fiber connection, as well as the data that are received from FELIX via serial E-links over the twinax cabling. These E-links, which are all downlinks, have a well-defined timing relationship between their data and the related clock. That is, regardless of the length of the twinax cable[11], the possibility of having variance in behavior across different elements of the system is low, because the data are transmitted alongside the clock. This is particularly true for the slow control data path, which is also slow in terms of data rate (only $80\,\mathrm{Mb/s}$).

However, the GBTx also sends data from the front-end electronics towards FELIX. These uplink data may need individual treatment, because the clock the GBTx uses to deserialize their bits, does not have a constant phase relationship with them. For the SCA's $80\,\mathrm{Mb/s}$ ($12.5\,\mathrm{ns}$ eye opening) connections, the possibility of errors on the associated E-links (128, one for each MMFE8) is low. For the L1 data paths however, which mostly operate at $320\,\mathrm{Mb/s}$, errors at a larger scale are often observed, because

end parts of the system (i.e. ALTI, FELIX, swROD and OPC Server/DCS), are also next-generation DAQ prototypes, a lot of time was devoted to tackle bugs in these elements as well. These issues were being communicated constantly to the relevant experts. Hence, because the development process of these parts of the system is not part of this dissertation, details on this are skipped, and their operation will be characterized as error-less for the rest of this work.

[11] The twinax cables connecting the GBTx with the MMFE8s that mediate the serial data, have two possible lengths: $3\,\mathrm{m}$ and $1\,\mathrm{m}$.

of the less wide eye opening of their transmission scheme (3.125 ns). For Phase-I, 160 E-links are used to transfer the L1 data from the MMFE8s to the L1DDCs. The ROCs corresponding to PCBs 1 through 6 use one sROC operating at 320 Mb/s, while the ROCs on the last two PCBs split their back-end L1 interface over two sROCs operating at 80 Mb/s. For Phase-II, 496 E-links are used instead, most operating at 320 Mb/s. For every sector, all uplink E-links with sROCs need individual calibration, in order to ensure that the clock the GBTx uses to deserialize the data, has an optimal phase relationship with them, to avoid setup and hold violations during deserialization.

Foreseeing this need, the GBTx developers have developed a mechanism that allows for automatic calibration of these paths [73]. At each uplink connection, a *phase aligner circuit* is used to skew the data with respect to the internal sampling clock of the GBTx. The amount of skewing can either be set manually, or even automatically. In the so-called *automatic phase tracking* mode, the actual phase of the received data is estimated from data bit transitions. This is also called *training* mode. After determining the optimum phase, the GBTx locks to it and waits for the user to send a command to "freeze" the process, in order to retain the skewing settings on all individual E-links. The phase settings are then written to dedicated registers of the GBTx address space. A block diagram depicting the logic involved is shown in Figure 7.12.

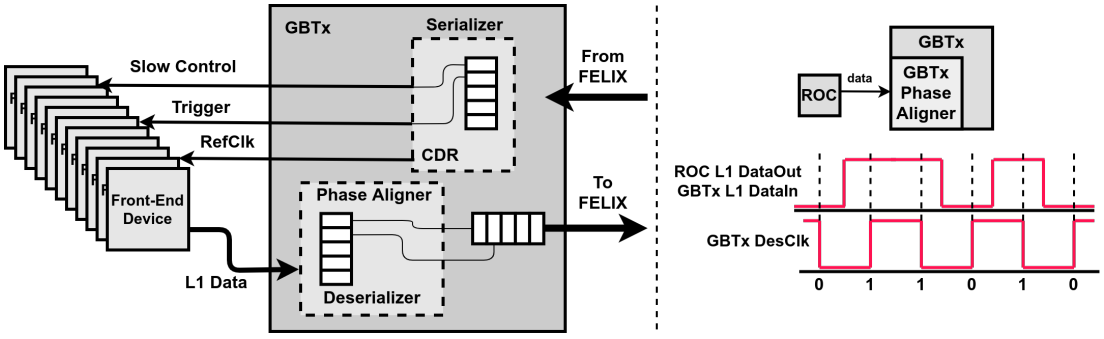


Figure 7.12: Rough representation of the GBTx phase aligning mechanism. The L1 data sent to the GBTx by the ROC must be sampled in the middle of the eye opening (timing diagram shown on the right) by the GBTx's uplink deserializing circuitry.

However, because this mode of operation is not recommended to be used long-term in SEU environments, the optimum phase relationship values determined by the automated logic must be read by the user, and parsed back into a set of different triplicated registers which hold the phase/skew values, and are used by the GBTx when operating in *manual* mode. All of the above imply that the user must be able to access the GBTx registers via the DAQ infrastructure. This is done via the so-called *IC E-link*, which makes use of a dedicated two-bit field of the GBT frame (see Figure 7.1) that is transmitted to and from FELIX. In essence, the bits of the IC part

of the GBT frame that is transmitted from FELIX, are not serialized by the GBTx over any of its connections to another device, but are routed internally to interface itself with the back-end system. The IC part of the frame that is sent to FELIX, contains the bits of the reply that is generated by the ASIC, upon completion of a register access operation by FELIX. Therefore, the only requirement to use this interface is to have a bidirectional connection with FELIX[12]. The GBTx expects a frame with a pre-defined format (see [73]) to be sent to it over the IC E-link, comprised of a delimiter, a field defining the type of command (read or write), the GBTx register's address, and the data-to-be-written for write operations. A checksum is appended at the end of the frame, before the final delimiter, for error-checking purposes. Upon reception of the frame and completion of the desired operation, the GBTx replies back to FELIX via the IC E-link, indicating successful completion.

Therefore, the procedure that must be followed in order to ensure data integrity in the sROC L1 data uplink paths is the following:

1. Configure the entire address space of the GBTx via the IC E-link in training mode.
2. Configure the front-end devices (ROCs and VMMs) via the MMFE8 SCAs.
3. Freeze the training mode of the GBTx. This is either performed by reseting the training logic, or by ”freezing” the training registers. After doing so, the GBTx determines the optimal uplink phase values and locks in the inbound datastream from the ROCs.
4. Read the phase values that were found by the GBTx's phase aligner logic. Write them back to the GBTx triplicated manual mode registers, and set the GBTx to manual mode. This step is only needed when operating in the ATLAS cavern.

The application that will be used in ATLAS to access the GBTx address space is the *IC-Over-NetIO* [76]. It makes use of the FELIX libraries that implement the back-end NetIO interface to communicate with FELIX, alongside parts of the FELIX low-level software tools [77] to build the IC frame. A class diagram of the package is shown in Figure 7.13. The user interfaces with the application via the command line, and provides a file containing the values of the entire address space of the GBTx, alongside the IC E-link addresses (16 in total for the BB5 sector) of the to-be-configured ASICs. In addition, the user can also read the values of individual registers, or write values into specific registers of a given GBTx. The application builds the frame the GBTx is

[12]For the GBTxs used in Phase-II, where data are only sent by them to FELIX, the ASIC's address space is accessed via the L1DDC's SCA's I^2C interface. The GBTx has a corresponding component used to provide access to the user in this case.

expecting, and opens a NetIO socket, through which it transmits the frame. The FELIX core application then forwards the frame to the FELIX FPGA, which sends two bits of it every $\sim$ 25 ns to the corresponding E-link, via the transceiver which implements the associated fiber connection. The GBTx then replies back, and the FPGA sends the reply frame back to the IC configuration application through a reply socket, again via the FELIX core application. Tests that had been conducted prior to the writing of this Subsection have shown that the application can configure all 16 GBTxs of a sector within a few seconds. The said software package is foreseen to be deployed as a service which can be accessed by both the DCS, as well as the DAQ infrastructure. This is because the ability to access the registers of the GBTx is considered imperative by both parts of the system, since the GBTx handles the data-flow of both slow control data (associated with DCS services) and L1 data (associated with the DAQ). If, for instance, the low voltage supply is intermittently lost over some front-end elements, the DCS should be able to request the IC configuration software to configure the related GBTxs, in order to access the SCAs that use the GBTxs to communicate with the OPC server via FELIX. After doing so, the DAQ infrastructure should be able to perform the uplink E-link training on these GBTxs, after configuration of the front-end elements.

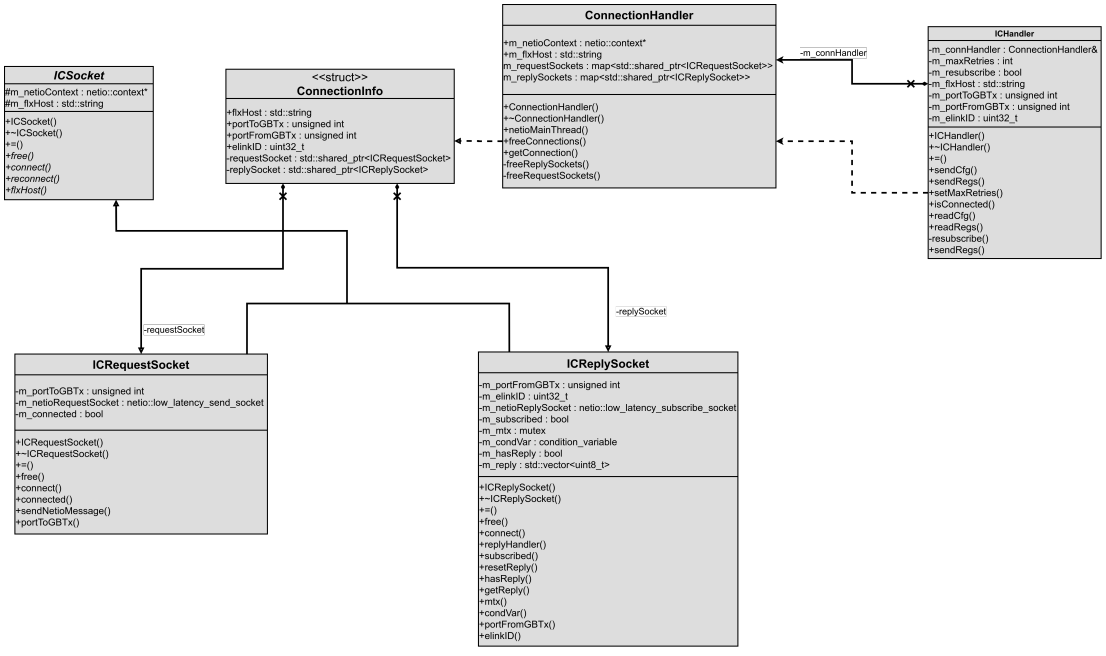


Figure 7.13: Class diagram of the IC-Over-NetIO application [76].

7.4.2 The ROC TTC Clock Phase Calibration and Phase-II E-link Testing Scripts

As mentioned in the previous Subsection, the serial downlink slow control E-links are for the most part error-free, because of their constant phase relationship with the reference clock, and due to their slow rate. No mention was made for the other serial downlink path, which is the *TTC* connection from the GBTx to the ROC (see Figure 7.11). Functionality-wise, the ATLAS Trigger Timing and Control system [57] defines a set of eight commands that are sent by it. Different subsystems adapt their electronics accordingly to interpret these commands (or simply, *bits*) depending on their needs. To optimize for latency, these eight possible commands are mediated at $320\,\mathrm{Mb/s}$, and a device that receives the TTC stream over a serial line (i.e. a downlink serial E-link), determines what command was sent to it by detecting the phase relationship of the received bit with respect to the reference clock (or bunch-crossing clock). The receiver deserializes the TTC stream using a $320\,\mathrm{MHz}$ clock[13] synchronous to the bunch-crossing clock, and depending on the placement of the sampled bit with respect to the $\sim 40\,\mathrm{MHz}$ clock's period, the receiver acts accordingly. For example, the most important bits of the TTC stream that the NSW is using in its subsystem, are the following:

- `Level-1 Accept (L1A)`: The Level-1 trigger bit, is used by the ROC to match the data that it captured from the VMM and send them out towards FELIX via its E-links.
- `Soft-Reset (SR)`: A generic reset signal that the front-end devices use to re-initialize their logic. Upon reception, the ROC fans-out a reset to its registers, while it also grounds the *ENA* signal that it drives to the VMM, in order to reset that as well.
- `Test-Pulse (TP)`: The command that the VMM uses to inject artificial pulses to its channels is issued by the TTC stream in the final system. Upon reception, the ROC pulls the *CKTP* signal that it drives to the VMM high.
- `Event Counter Reset (ECR)`: This signal is used by the ROC to reset its L1ID counter. It is also used by FELIX. Upon reception of the said bit by the ALTI, FELIX resets its internal L1ID counter, and it also sends the command via the TTC E-links to the ROCs via the GBTx, in order for them to reset their trigger counters as well. This is how the system is synchronized on the trigger sequencing ID level.

[13] Or a $160\,\mathrm{MHz}$ clock at double data rate.

- `Bunch-Crossing Reset (BCR)`: This signal is used by the ROC and the VMM to reset their respective BCID counters. It is also used by FELIX. Upon reception of the said bit by the ALTI, FELIX resets its internal BCID counter (its value is also contained in the L1ID packet that it sends to the swROD over the associated virtual E-link), and it also sends the command via the TTC E-links to the ROCs via the GBTx. The ROC then resets its own BCID counter, while it also grounds the *TKI* signal that it drives to the VMM, thus resetting the latter's BCID counter as well. This is how the system is synchronized on the BCID level.

- `Level-0 Accept (L0A)`: The Level-0 trigger bit, is forwarded by the ROC to the VMM immediately upon reception.

The ROC implements CERN's ePLL to recover the bunch-crossing clock, as received by the GBTx, and creates a copy of it for its internal logic. The PLL also generates a 160 MHz clock, which is used by its logic as well. This clock is also employed by the *TTC Capture* module, which receives the serial TTC stream from the GBTx, and using the 160 MHz reference at double data rate, samples the inbound E-link data. Upon detection of a high bit, the TTC capture logic determines its phase relationship with the recovered 40 MHz clock, in order to issue the associated command. The "first" bit after the rising edge of the 40 MHz for instance, is the L1 trigger. If a high bit is found after 3.125 ns of that, then this corresponds to the soft reset command, which the ROC sends to its logic and to the VMM by grounding the *ENA* pin for one bunch-crossing. The scheme is depicted in Figure 7.14.

The TTC path operates at $320\,\mathrm{Mb/s}$, which makes it prone to bit misinterpretations because of timing violations on the receiver's side. However, given the fact that the TTC data are deserialized using the reference clock which is sent from the GBTx to the ROC alongside the stream in question, one would think that if the correct phase relationship between the clock and the data is found, then this setting can be the same for all ROCs of the system. Unfortunately, sometimes this is not true, because the clock recovery mechanism inside the ROC might introduce enough uncertainty to impose individual treatment of each DAQ node. This uncertainty can lead to timing violations, which may cause some bits to be sampled correctly in an intermittent manner, or even entire bit-slips, thus introducing a consistent shifting of the stream inside the logic (i.e. when the system issues a test-pulse, a particular "bit-slipped" ROC always interprets it as a soft-reset, or as an ECR). To circumvent this, the ePLL allows for configuring the phase relationship of all output clocks with respect to the input reference, with 200 ps granularity. Hence, the user of the system may, via the SCA's I^2C interface, skew each individual ROC's clocks with respect to the global bunch-crossing clock. Since the global clock has a well-defined phase relationship with the TTC stream, the user

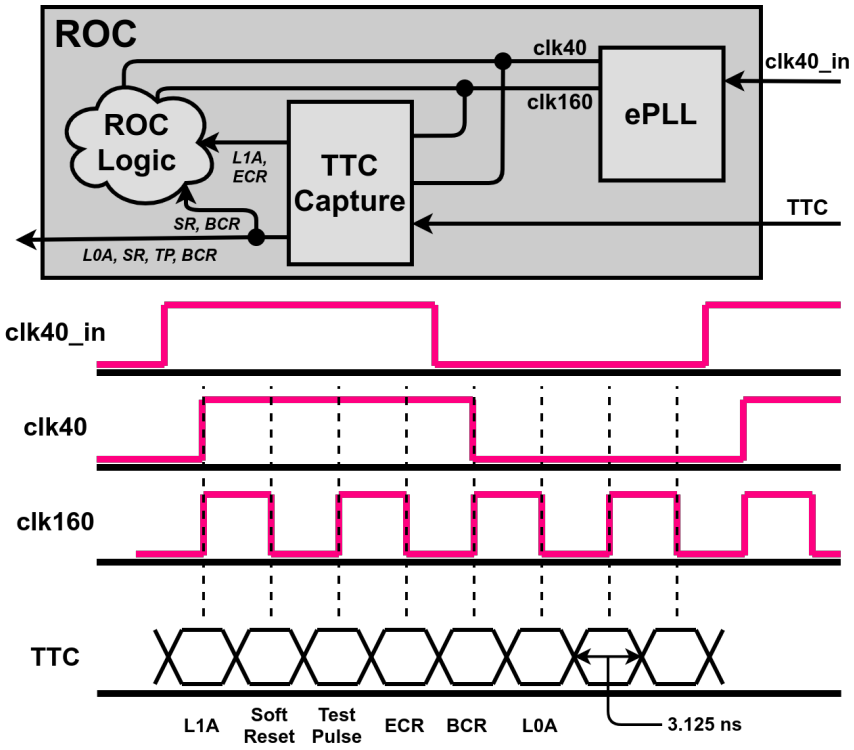


Figure 7.14: *Top:* Rough representation of the ROC's TTC deserialization scheme. The associated module receives two clocks, and uses the fast one (160 MHz) to deserialize the inbound TTC stream, and the slow one (40 MHz) for command interpretation. Some commands are sent both to the VMM and the internal logic, while some are only relayed to either of the two. *Bottom:* Timing diagram of the TTC stream. Each bit is 3.125 ns in width, and is sampled by a 160 MHz clock at double data rate.

essentially skews the internal 40 and 160 MHz clocks of each ROC with respect to the inbound TTC bits, thus allowing for calibration of that path, in order to prevent the potential issues that were mentioned above.

This calibration is performed by a set of scripts, which form the *Netio Traffic Analyzer* package [78]. Upon installation of a sector on the BB5 cosmic stand, the first procedure that is executed is that of the analyzer, in order to validate all of the data paths (uplink Level-1 and downlink TTC) via test-pulsing. The TTC path evaluation is performed by evaluating the ROC's performance while issuing test pulses to its associated VMMs. The main routine of the package is implemented in an `expect` script[14], that configures the front-end electronics of the sector-under-test by following a fail-safe sequence, issues test-pulses to the VMMs via the ALTI, and records the data output of the ROCs in a round-robin manner, using the NetIO interface of FELIX. After recording

[14] A subset of the `TCL` scripting language.

the ROC data output, a `Python` application is used to evaluate all E-links' "health". The said software produces a log file, that lists the E-links (and hence, the associated ROCs/sROCs) which were tested, alongside a grade which is determined by the size of the data packets and their total count. One thousand test-pulses are injected to the system in total, and hence, one thousand packets are expected to be sent by each sROC, with the packet size depending on the number of VMMs connected to each sROC. An example log that summarizes eight MMFE8s connected to one L1DDC/GBTx is given in the table below:

E-link ID	MMFE8 ID	Health	Events	Mean Size
272	L3P1_IPL	A+	1000	2061.0
276	L3P2_IPL	A	1000	1994.0
280	L3P3_IPL	A+	1000	2064.0
284	L3P4_IPL	F	0	0
289	L3P5_IPL	A+	1000	2048
293	L3P6_IPL	A+	1000	2062.2
265	L3P7_IPL_A	A+	1000	1038.0
266	L3P7_IPL_B	A+	1000	1037.0
268	L3P8_IPL_A	A+	1000	1038.0
269	L3P8_IPL_B	A+	1000	1033.0

The procedure that is followed by the scripts is not completely automated, but requires user interpretation of the data. The user inspects the log files, and determines which E-links may be problematic. In the example of the table above, the MMFE8 on PCB 4 seems to not be sending any data at all. One potential reason behind this, is that the ROC of that board does not interpret the TTC bits correctly, because its internal clock phase is off. The user can change the configuration of that given ROC's ePLL to shift its clock phases using another `Python` script, that goes through the configuration file associated with the sector, and shifts the clocks of that misbehaving node by a few hundred picoseconds. The user then runs the routine again, and inspects the log file for any change of behavior. If a board persists in not sending any data, then this may indicate a hardware issue (i.e. that particular E-link connection may be problematic, prompting for changing the MMFE8, the twinax cable, or the associated L1DDC). The said routine cycles through all 128 MMFE8s of a sector in about 20 minutes. The reason it takes that long, is because the `expect` script has to perform Secure Shell (SSH) connections and execute remote commands multiple times. Also, there is a limitation on the FELIX side that is overcome by probing only eight MMFE8s for data

(or one L1DDC/GBTx) at a time. To be precise, if a given E-link is not performing as it should, it may be transmitting noise to FELIX, which is translated into large data-flow. If FELIX is subjected to this for a few minutes (or even seconds), it crashes, because it is overwhelmed by the large amount of data. This is circumvented by turning off all of the L1 data E-links on FELIX, except for those that are currently being probed for data. The entire procedure is listed in the steps below:

1. Configure FELIX to turn on all of its slow control and TTC E-links (to propagate OPC server data for front-end electronics configuration, and trigger bits for initializing the ROC and the VMM).
2. Configure the GBTxs in training mode.
3. Configure the VMMs and the ROCs of all MMFE8s.
4. Freeze the GBTx training.
5. Send all the necessary reset signals via the ALTI (SR, BCR, ECR).
6. Configure FELIX to turn off all of its E-links.
7. Enter the test-pulsing routine:
   (a) Configure FELIX to turn on only the L1 data and TTC E-links associated with the sROCs/MMFE8s (only one L1DDC) that they will now be test-pulsed.
   (b) Execute the `netio_cat` command in FELIX for the E-links that will now be test-pulsed. This command dumps the packets from these E-links into files for later analysis.
   (c) Start the test-pulsing pattern via the ALTI (issues a TP and then a L0A/L1A with a fixed latency in-between them. Repeats that 1000 times).
   (d) Stop the test-pulsing.
   (e) Analyze the `netio_cat` dumps via a dedicated `Python` script. The said piece of software creates a log file that grades the health of the E-links that were just tested.
   (f) Repeat 16 times for all L1DDCs.

The above routine is run before the sector is subjected to cosmic rays, to ensure that the electronics are in a good enough state to perform a flawless data acquisition process, without hampering the physics data-taking or impede the detector's performance. Also, the fact that this is run at BB5 allows for hardware interventions, because if an issue is

found on the hardware infrastructure, the commissioning site is the only place where changes can be made easily[15].

The same software infrastructure is also used to test the Phase-II E-links. For Phase-I, the procedure described above tests all 128 MMFE8s, which corresponds to 128 TTC E-links and 160 L1 data E-links (it is reminded that the ROCs of the last two PCBs split their data throughput in two sROCs, hence the `L3P[7,8]_A` and `B` in the log file table that was added above). After validating that the ROC clocks have the correct phase, the routine may be used to perform a check on the 496 L1 data E-links of Phase-II. The only differences are that now two FELIX boards are used within the same server (the first one performs the bidirectional communication used also for Phase-I readout, while the second one which is used for Phase-II, only accepts data from the GBTxs), that the ROCs are configured in a different manner (all of them split their throughput into four sROCs, most running at 320 Mb/s), and that now all three GBTxs per L1DDC are trained instead of one (the first GBTx is configured via the IC E-link, while the Phase-II GBTxs are configured via the L1DDC's SCA). Finally, the testing routine subscribes to 31 E-links per step, instead of just 10. Similarly to the Phase-I testing procedure, the Phase-II E-link testing takes about 20 minutes to test the entire Micromegas sector, and validate its L1 data-taking performance prioor to its installation into the ATLAS cavern.

7.4.3 The BB5 Fine-Trigger-Selection Scheme

Since one of the main goals of the BB5 commissioning site is to validate the detector's performance regarding muon track reconstruction, evaluating its spatial resolution under cosmic radiation is also desirable. At the time of writing of this Subsection, measurements had shown that on average, the tracks which traversed the sectors that had already been commissioned, were perpendicular with respect to the readout planes. However, since the Micromegas is expected to accept muons with angles $8^\circ < \theta < 30^\circ$ once deployed in the ATLAS cavern, incident cosmic muons that arrive at an angle need to be reconstructed too, in order to evaluate the spatial resolution of the double-wedge on these conditions as well. As mentioned in Chapter 6 however, where the first Micromegas SM module was tested under testbeam conditions but with the FPGA-based readout, the angled-track reconstruction is not optimal when using the simple residuals/centroid methodology (see Subsection 6.3.3). The μTPC technique, introduced in previous works [3, 4, 55] should be used instead. However, in order for this method to work, the events should be synchronous to the reference clock of

[15] Access is limited when the Micromegas sector is mounted on the NSW envelope at building 191, let alone inside the ATLAS cavern.

the DAQ system, that the VMM uses to evaluate the timing of the muon-induced-ionizations[16]. This was not true for the DAQ system that was used in the testbeam, described in Chapter 6, and it is also not true for the cosmic stand at BB5, since the cosmic muons are asynchronous with respect to the clock.

However, in the scheme described in Subsection 6.3.1, the input trigger signal from the scintillator coincidence, was propagated to the front-end electronics only when it was edge-aligned with the system clock (see Figure 6.10), with 1.5625 ns accuracy. This emulated a scenario where the events were synchronous with the reference clock, similar to ATLAS, that allowed for precise μTPC measurements. This scheme, was also deployed at BB5 using the same firmware as a base, with small modifications. For the BB5 setup, the Xilinx® VC709 FPGA accepts a reference clock through a pair of SubMiniature version A (SMA) cables, provided by the GBTx of an L1DDC that recovers the system clock (whose source is the ALTI module), via the fiber connection with FELIX. The trigger scintillator coincidence, does not directly drive the ALTI module, but is used as a trigger input to a Clock and Trigger Fanout (CTF) module (see Subsection 6.1.1), which in turn provides the raw trigger to the VC709 via a twinax/miniSAS connection through the FMC. The VC709 performs the fine-trigger selection in its logic, in the same manner as the system described in Subsection 6.3.1. The VC709 outputs the edge-aligned trigger through a pair of SMA cables, and via a dedicated converting board, the fine-selected trigger is driven to the ALTI trigger input via a LEMO connection.

Conclusions

In this Chapter, the New Small Wheel electronics that are involved with the L1 readout of the Micromegas detector were introduced in greater detail. Other parts of the system, such as the trigger electronics of the Micromegas and the sTGC, were omitted. Based on the VMM and the ROC, which are chips that have been tailored to satisfy the needs of the NSW, the L1 readout of the Micromegas also makes use of advanced electronics developed by CERN (GBTx and SCA), to complete the picture of the readout. These radiation-tolerant ASICs, deployed on MMFE8 and L1DDC boards, interface with FELIX, which will be the next-generation DAQ system that ATLAS will be using in the following years, with the NSW being one of the first detector subsystems to employ it. The back-end infrastructure is also comprised of a dedicated OPC Server (part of the DCS), that communicates with the SCAs via FELIX, in order to configure and monitor all front-end elements. The swROD (part of the DAQ), aggregates the L1 data that FELIX reads-out from the ROCs, and builds fragments, that are propagated to the

[16]This is true for the ATLAS run conditions, since the protons collide at the core of the detector at each rising edge of the reference clock.

HLT. Finally, the ALTI provides the reference clock to the system, alongside the TTC commands, that are fanned-out to all front-end elements with minimal latency.

For more information, the reader is also referred to a general NSW-electronics-related proceeding that was published by the writer on behalf of the ATLAS collaboration [79].

In the BB5 commissioning site, the system described above, is employed to read-out Micromegas sectors that are to be deployed in the ATLAS cavern. With a turnaround time of just two weeks, the BB5 site is dedicated in validating the detector's performance, as well as the electronics that instrument it, before the double-wedge is installed in the ATLAS cavern, where access is limited, and hardware interventions are almost impossible. The process of integrating the NSW electronics with the ATLAS DAQ infrastructure was not a trivial task; significant debugging was needed in all parts of the system. Due to the large scale of the DAQ system, a set of tools were developed that validated and calibrated several parts of the system, in order to ensure that the sector, alongside its electronics, are both suitable to serve ATLAS. In Figure 7.15, the timing distribution of the hits recorded by all VMMs of the sector when read-out under test-pulsing is provided, indicating that the timing of the system is behaving as it should. This is one of the standard tests, performed after validating the electronics functionalities via the supporting tools mentioned in this Chapter.

The commissioning process has proved to be successful, with the table below showing the per-layer efficiency of each sector that had been deployed and validated in the BB5 cosmic stand, when subjected under cosmic radiation, under nominal detector operation parameters. In addition, the percentage of dead channels per sector is also provided, which is derived after reading-out all VMMs under test-pulses.

	A02	A06	A08	A10	A12	A14	A16
Layer 0 Eff.	97.8%	97.6%	97.5%	94.9%	92.2%	85.4%	98.0%
Layer 1 Eff.	96.6%	97.5%	96.8%	94.8%	92.2%	89.1%	91.8%
Layer 2 Eff.	95.4%	91.8%	93.0%	92.1%	92.9%	84.1%	96.6%
Layer 3 Eff.	96.2%	89.9%	86.6%	92.6%	94.4%	85.3%	97.5%
Layer 4 Eff.	94.4%	93.6%	84.6%	92.6%	96.1%	90.8%	81.8%
Layer 5 Eff.	89.3%	96.7%	94.7%	95.3%	89.5%	92.8%	96.7%
Layer 6 Eff.	98.3%	98.0%	97.7%	96.9%	80.2%	97.2%	90.3%
Layer 7 Eff.	95.0%	90.9%	92.6%	96.7%	96.7%	98.1%	95.2%
Dead Ch. Perc.	0.49%	0.53%	0.37%	0.50%	0.62%	1.70%	0.52%

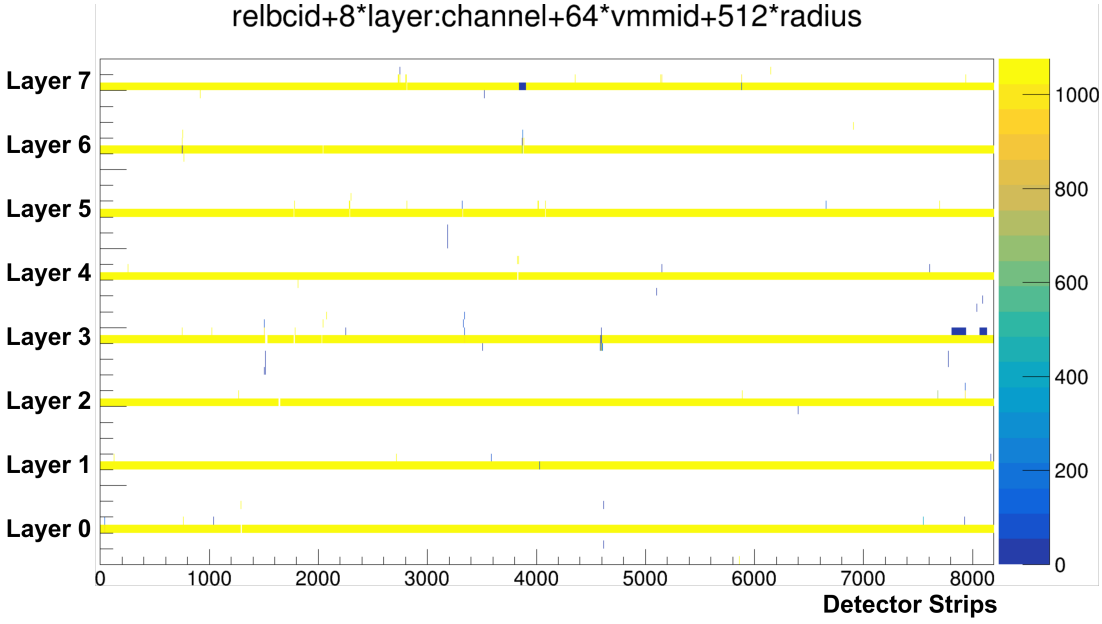


Figure 7.15: *Relbcid* variable distribution of the hits recorded by the VMMs of a sector when test-pulsed. One thousand pulses are injected by the ALTI, which then issues a L0/L1 trigger that reads-out the related data. It is reminded that the *relbcid* (relative BCID) variable is appended by each VMM channel on the data, and indicates the time difference between the event's BCID timestamp, and the trigger's timestamp, in BC counts. A value of three implies perfect timing in the middle of the readout window, while values of zero or seven imply that the event is near the readout window's edge. In this slide, all eight layers of the sector seem to record hits of only one relbcid (of value three). This means that the trigger signals and the reference clocking of the system are in full sync, thus validating the performance of the system under test-pulsing, which is the first step prior to subjecting the detector to cosmic radiation.

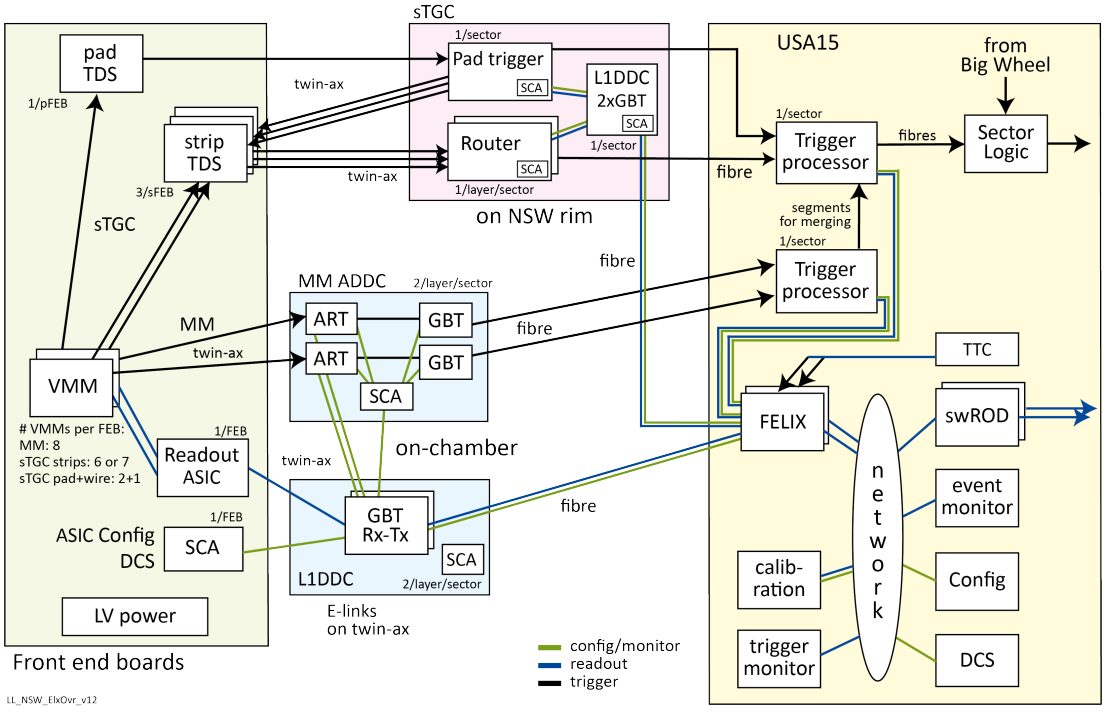


Figure 7.16: Overview of the NSW electronics system. On the left-hand side, the front-end boards that are attached to the chambers bear VMMs, SCAs and ROCs, while for the sTGCs they also host TDS chips. The ROC aggregates L1 data from many VMMs and sends them to FELIX via the GBTx. FELIX also sends trigger signals to the front-end electronics via the ATLAS TTC system. It also sends tracking data from the ROC to the swROD (here depicted to send the data fragments to the HLT), and communicates with the Detector Control System (DCS) for slow control purposes. The Micromegas trigger data are collected from many VMMs by the ART ASIC, which sends them towards the Micromegas trigger processor, alongside geographical and timestamp information. The TDS chips on the other hand, handle the sTGC trigger primitives, and alongside the Pad Trigger and Router boards, are part of the more complex sTGC trigger chain, which has the associated trigger processor FPGA on the back-end as a final destination of its data-flow. Both FPGAs create muon candidates that are then transmitted to the SL.

Chapter 8

The Slow Control Adapter eXtension

As mentioned in Chapter 7, one of the key elements of the NSW electronics scheme is the Slow Control Adapter, or SCA [43]. The SCA is part of the Giga-Bit Transceiver optical link (GBT) chipset, a family of ASICs developed at CERN for high-throughput data transferring purposes under harsh radiation conditions. The main component of the said chipset is the Giga-Bit Transceiver (GBTx) ASIC, which implements the GBT protocol [63, 64] in order to mediate the data between the back-end (i.e. FELIX) and the front-end electronics[1]. The data in question may be trigger or slow control data that are being sent towards the front-end devices (also referred to as *downlink*), or collision-related data that propagate from the front-end towards the back-end (also referred to as *uplink*). The slow control path is bidirectional. That is, the uplink path also forwards slow-control-related packets (configuration replies or monitoring information) from the front-end nodes towards the back-end.

The SCA is usually connected both to a GBTx and to several front-end devices, and has the purpose of distributing control signals to the on-detector front-end electronics, and perform monitoring operations of detector environmental parameters. In order to meet the requirements of different front-end ASICs used in high-energy physics experiments, it provides various user-configurable interfaces, such as SPI, I^2C or JTAG, and is capable of performing simultaneous operations. Its other important feature though, is its radiation hardness. It is designed employing radiation-tolerant design techniques

[1] The GBT scheme however, apart from the GBTx ASIC, can also be implemented in an FPGA [65]. As mentioned in Chapter 7, FELIX deploys the GBT-FPGA instance in its design to communicate with the GBTxs. But other FPGAs may also employ the same module in their gateware to interface with FELIX.

to ensure robustness against Single Event Upsets (SEUs) and is implemented in a commercial 130 nm CMOS technology.

Although the majority of the NSW electronics are subject to high radiation doses [7], there are several parts of the DAQ system that are FPGA-based and are situated in parts of the detector which are not imposed to a particle flux that may disturb the acquisition procedure, while some are deployed outside the experimental cavern, in the counting room (or USA15). Two examples of the NSW electronics are the Pad Trigger board and the NSW Trigger Processor (TP), that are placed on the NSW's rim[2] and at USA15 respectively. These two FPGAs interface with FELIX via optical fiber, with the subtle difference being that for the Pad this communication is achieved via the GBTx on the RIM-L1DDC, and for the TP the bidirectional GBT link is being implemented via a GBT-FPGA instantiation within the TP's FPGA.

Similarly to the front-end ASICs, the aforementioned FPGA systems require configuration before and during runtime. Given a hypothetical scenario where an FPGA must receive slow control data from the back-end via the SCA - GBTx - FELIX chain, one can imagine that a board which bears the said FPGA should also deploy two ASICs for the back-end communication, configuration and monitoring data propagation. This sometimes can be inconvenient, for practical reasons that can be accredited first of all to board design considerations, which increases in complexity with the addition of extra components, and also to cost increase. Therefore, in a similar manner to the GBT scheme, which has an ASIC flavor (GBTx), and an FPGA flavor (GBT-FPGA), the idea that the SCA may be deployed inside the FPGA fabric manifests itself in the Slow Control Adapter eXtension, or SCAX [80–82].

8.1 The SCA Interface with the Back-End

In order to understand the concept of the SCAX as a whole, the reader should have a better idea on how the SCA actually interfaces with the back-end. For the NSW project alone, ~ 6400 SCAs will be used to configure and monitor various ASICs that are part of the general DAQ scheme. All of these SCAs communicate with the back-end (i.e. FELIX) via the GBTx, which implements the GBT protocol in its logic, while FELIX implements the GBT-FPGA core in order to communicate with the aforementioned package. The SCA-to-GBTx interface is the *E-link*, a serial differential line running at $80\,\mathrm{Mb/s}$ over HDLC encoding. The protocol defined by the SCA's requirements is request/respond; that is, for every packet received by the

[2] Situated about four meters away from the beamline, the expected total ionizing dose on the rim is about two orders of magnitude less than areas half a meter away from the beam's axis [7].

SCA, the back-end that originally transmitted the packet awaits for an associated reply by the SCA. In this SCA - GBTx - FELIX communication chain, the last two components can be viewed as data mediators, so there is one piece missing: the back-end logic that actually builds the packets-to-be-transmitted to the SCA, and handles the inbound traffic from the ASIC. This is a software suite, which is a dedicated Open Communications Platform Unified Automation (OPC UA) server [66]. This scheme is visualized in Figure 8.1.

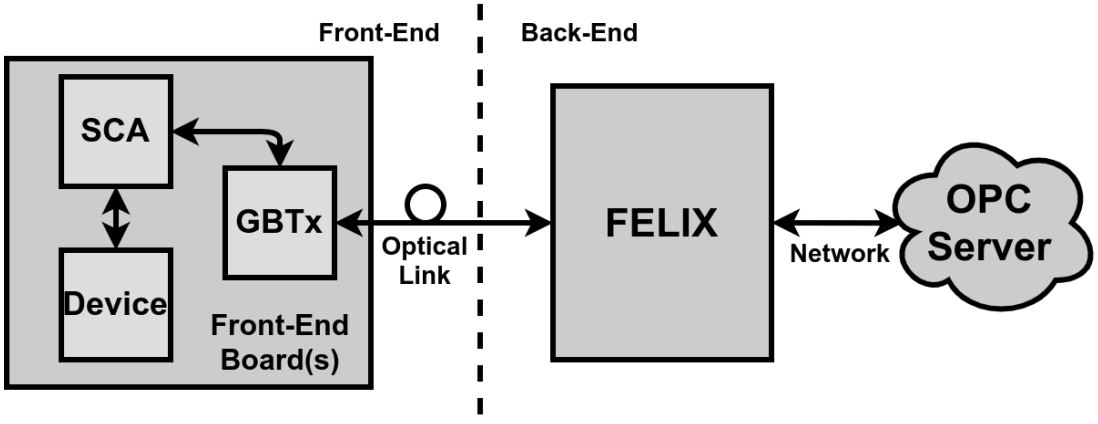


Figure 8.1: Overview of the OPC Server - FELIX - GBTx - SCA Connectivity. The GBTx and the SCA may be situated on different front-end boards. The connection between the GBTx and the SCA is a serial differential electrical link (E-link). The SCA may be interfacing with several front-end chips to configure and monitor them.

The OPC UA server (also referred to as 'OPC server'), can handle multiple SCA nodes that are connected to a FELIX host via the GBTxs. The connection of the OPC server with FELIX is achieved via NetIO [75], which carries the packets being sent to the SCA (request) or to the OPC server (reply). The structure of these packets can be inspected in Figure 8.2.

Each field of the SCA's frame have well-defined purposes:

- `Start of Frame (SOF)`: A specific pattern indicating that a packet follows.
- `Address`: Always equal to 0x00.
- `Control`: Yields information related to the sequence of the transactions between the OPC server and the SCA, or in the case of S-frames (part of the HDLC standard), encodes specific commands directed to the SCAX (e.g. a reset command can be sent).
- `Transaction ID (Tr.ID)`: Yields information related to the sequence of the transactions between the OPC server and the SCA. To be precise, if the server

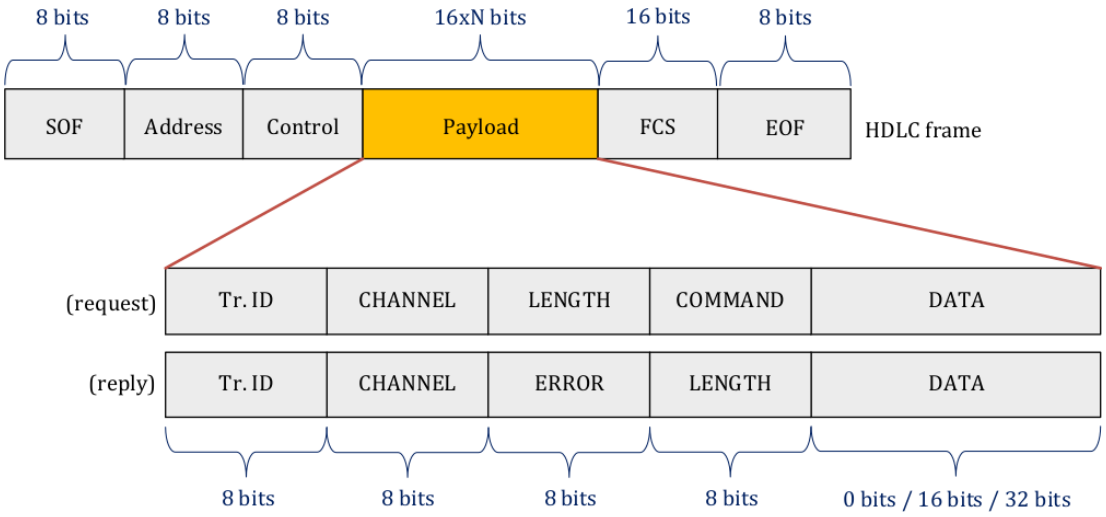


Figure 8.2: SCA communication protocol frame structure that is encapsulated inside a NetIO frame when propagating data via FELIX. In the NSW case, the request is sent by the OPC server, and the reply by the SCA [43].

sends a frame with TrID = 0x00, the SCA shall respond with a frame containing TrID = 0x00. The next packet sent to the SCA should have a value of TrID = 0x01 etc.

- `Channel`: In each transaction, the server addresses one specific *Channel*. Each Channel represents a different interface that the SCA offers to communicate with other devices. For example, there is a GPIO Channel and sixteen distinct I^2C Channels on the SCA.
- `Length`: Length of the *Data* field in bytes.
- `Command`: Each specific Channel has a set of possible functions that an inbound packet addresses. This field designates what kind of functionality the packet is related to.
- `Data`: This field yields the data necessary for each transaction. For example, it may contain the address of the I^2C device that the SCA must access, plus any data-to-be-written.
- `Frame Check Sequence (FCS)`: This field is calculated over the *Address*, *Control* and *Payload* fields using the CCITT standard 16-bit Cyclic Redundancy Check (CRC), and may be used by the receiver to check the integrity of the data.
- `End of Frame (EOF)`: A specific pattern indicating that the preceding packet has ended.

The main concept of the SCAX revolves around *transparency* with respect to the dedicated OPC server that was originally designed to interface only with the SCA. The SCAX interface with the back-end emulates that of the SCA's from the OPC server's point of view. In addition, since the SCAX interfaces with the registers of the User FPGA Logic (UFL) that it is instantiated in, that interface should mimic one of the SCA's Channels, in order to further enhance the transparency concept and reach the final goal, which is to use the same software suite that controls the SCA, to also control the SCAX and the FPGA that it is instantiated in. This is achieved by emulating the I^2C interface of the SCA inside the SCAX's logic. The SCA deploys sixteen distinct and completely independent I^2C Channels. Each Channel can connect to a pair of SDA, SCL lines and communicate with multiple devices. In a similar manner, the SCAX deploys sixteen I^2C Channels itself, and each one is associated with *one* logic block of the UFL, via another component called *Register File*[3]. This allows the OPC software suite to access both ASIC and FPGA address spaces. The general implementation scheme of the SCAX can be viewed in Figure 8.3.

8.2 SCAX - Architectural and Functional Overview

SCAX may be used in any FPGA implementation of the ATLAS DAQ electronics that interfaces with FELIX. Therefore, flexibility and adaptability are considered to be of utmost importance. Given that, several design considerations were taken into account in order to facilitate the module's deployment in a pre-existing design. Figure 8.4 provides an overview of the SCAX's architecture in block diagram form.

A brief description of all major components is provided below:

- `Elink2FIFO`: Originating from the FELIX HDL source files (see Appendix E), this module implements the interfacing with FELIX, either via a GBTx, or via the GBT-FPGA logic. The SCAX supports all three possible data-rates that FELIX does ($80,\ 160$ or $320\,\mathrm{Mb/s}$). The frames can either be 8b10b, or HDLC encoded. Most implementations of the SCAX are implemented at $80\,\mathrm{Mb/s}$ over 8b10b encoding. The Elink2FIFO component aligns to the incoming 8b10b stream originating from FELIX by scanning for the K28.5 comma character. It also recognizes the special characters that indicate a SOF and EOF (K28.1 and K28.6 respectively), drops them upon reception, and stores the frame contents in a buffer; the buffer (a FIFO), allows to cross clock domains, the one being the clock in sync with the incoming datastream, and the other one being the frequency of the SCAX core logic.

[3]Note however that the interface to the FPGA registers themselves is direct, not via an I^2C serial bus.

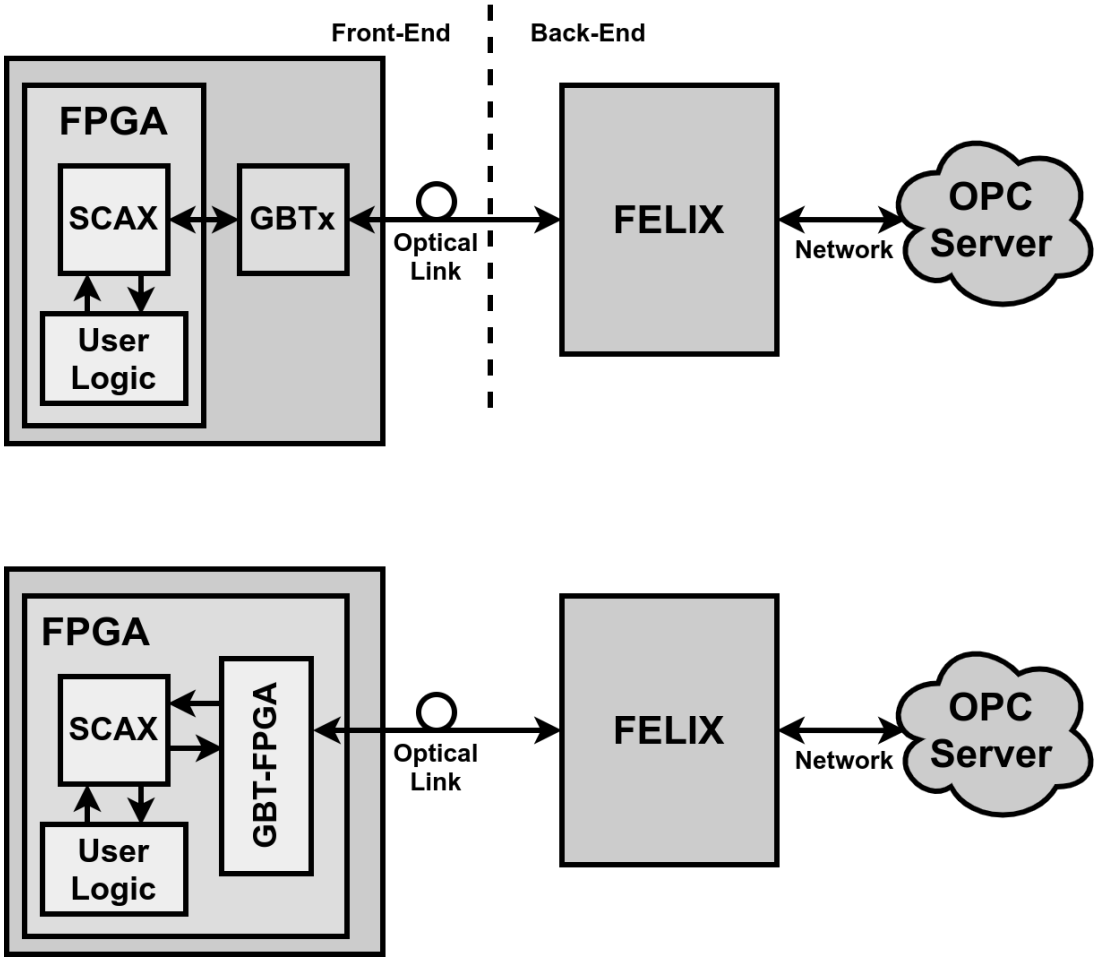


Figure 8.3: Overview of the OPC Server - FELIX - SCAX Connectivity. *Top:* The SCAX can connect with the OPC Server via a *serial* connection through the GBTx. *Bottom:* The SCAX can connect with the OPC Server via a *parallel* connection to a GBT-FPGA instance, often implemented in the same FPGA as the one the SCAX is in.

- `Deframer`: This module continuously queries the *Elink2FIFO's* buffer for data. If a packet has been received, the logic of the Deframer registers the different contents of the inbound frame (see Figure 8.2), and performs some first checks on its contents. For example, the module's logic measures the length of the inbound frame in bytes, which can either be 4, 8, 10 or 12 bytes, as defined by the SCA's protocol. Also, the module deploys an FCS logic block, that decodes the relevant field of the frame, by implementing a 16-bit CRC defined as: $G(x) = x^{16} + x^{12} + x^5 + 1$. If the received frame has no errors, all fields that are of interest for the rest of the logic are sent to the *Traffic Handler*. If an error was detected, it is reported to the aforementioned module, which acts accordingly.

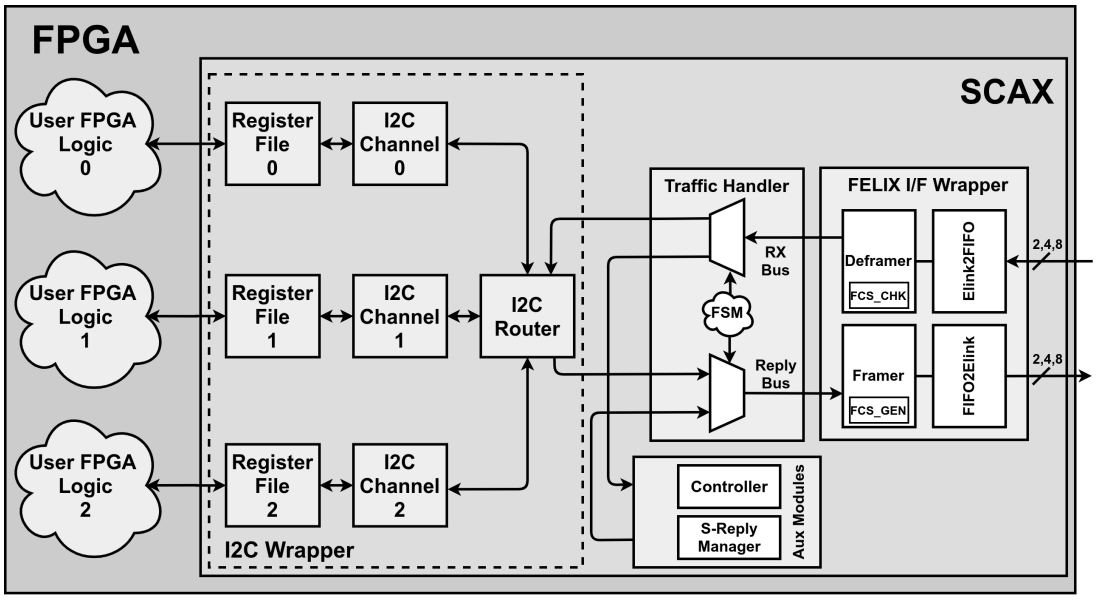


Figure 8.4: SCAX architecture block diagram. The sub-modules that implement the interface with the back-end can be seen on the right, while the I^2C logic alongside the register files that are part of the connection with the surrounding logic, are on the left-hand side of the diagram. The *Traffic Handler* manages the inbound and outbound frame data-flow, and two auxiliary logic components deal with all non-I^2C related types of transactions. Note that only three I^2C register file-channel pairs are depicted here, but the SCAX supports up to sixteen independent instances, each connected to a specific user logic component.

- `Traffic Handler`: The main FSM of the SCAX, its main purposes are the routing of the inbound frame fields to the corresponding sub-module, as well as reply bus arbitration. If a frame with no errors is received, the FSM forwards its fields to the addressed sub-module, which is defined by the value of the *Channel* field inside the frame itself. Possible recipients include the *Controller*, the *s-Reply Manager*, or the *I^2C Router*. If an erroneous frame is received, then the FSM of the handler signals the s-Reply Manager to generate a reply to the OPC server, that usually is a Selective Reject (SREJ) frame, which is a special frame, part of the HDLC standard, that the SCA communication protocol is part of. In any case, the addressed sub-module generates a reply that will be sent to the back-end via the *Framer*. The aforementioned module will go to a busy state once it receives a reply-to-be-sent, and will only become available once it has sent the reply out via the *FIFO2Elink* interfacing component. The main FSM goes back to idle once this transmission has ended. The state transition diagram of the said FSM can be viewed in Figure 8.5.
- `Framer`: This module connects to all channels of the SCAX logic via a

multiplexer controlled by the *Traffic Handler*. The FSM of of the Framer is activated by the SCAX's handler FSM upon the routing of a received frame from FELIX to an SCAX channel. The addressed channel generates a reply that forwards to the Framer. The latter then employs a CRC generator, to create an FCS which is appended to the outbound frame. This field is then evaluated by the OPC server to verify the integrity of the data. All the reply frame fields are then handed to the *FIFO2Elink* module that implements the actual interfacing with FELIX.

- `FIFO2Elink`: Originating from the FELIX HDL source files (see Appendix E), this component implements the interfacing with FELIX, either via a GBTx, or via the GBT-FPGA logic. The SCAX supports all three possible datarates that FELIX does (80, 160 or $320\,\mathrm{Mb/s}$). The frames can either be 8b10b, or HDLC encoded. Most implementations of the SCAX are implemented at $80\,\mathrm{Mb/s}$ over 8b10b encoding. FIFO2Elink implements a Clock-Domain-Crossing (CDC) FIFO, which accepts data-to-be-transmitted on its port that is in sync with the SCAX core clock[4], and then reads and encodes the data at the defined data rate.

- `Controller`: Loosely based on the SCA module of the same name, this component performs auxiliary functions and contains dummy channel registers for correct interfacing with the OPC server. Upon its initialization, the server addresses several SCA channels over a handful of transactions, in order to validate the ASIC's state. Since the SCAX does not deploy most of the SCA's channels (e.g. JTAG or SPI), the SCAX's Controller contains several dummy registers of the non-implemented channels, so that the OPC server can query them. Apart from this, the Controller issues a self-reset signal to the entire SCAX core, upon reception of the associated s-frame from the server. Finally, it is worth mentioning that each SCA chip has a unique ID that the server reads-back through the associated SCA module. The SCAX is also adapted to this. The Controller employs a dedicated Xilinx® IP that extracts the so-called 'Device DNA', a factory-programmed read-only ID [47], unique for each Xilinx® FPGA that the SCAX is implemented in.

- `s-Reply Manager`: The HDLC standard, on top of which the SCA communication protocol with the back-end is implemented, defines some supervisory level commands, transmitted in a 4-byte frame, or s-frame. Three of these commands are used by the SCA and the SCAX. These are the *Connect*, *Reset* and *Test*

[4]For the majority of the NSW FPGAs, this is a $320\,\mathrm{MHz}$ clock, derived from the reference clock which usually arrives to the FPGA either via the GBTx ASIC, or a GBT-FPGA implementation in the same, or another FPGA.

frames, which are handled by the module in question. Upon reception of the *Reset* command for example, the manager signals the *Controller* that a reset has been issued, and the latter subsequently issues a self-reset to the SCAX logic. Also, if a malformed frame was received by the SCAX, the *Traffic Handler* signals the manager accordingly, which then transmits a SREJ frame, that the OPC server evaluates.

- `I²C Router`: An FSM that acts as an extra pipeline stage between the *Traffic Handler* and the sixteen *I²C Channels*. This part of the logic is activated by the handler FSM if the inbound frame's *Channel* field has a value in the range of $0x03-0x12$[5]. It then routes the frame contents to the corresponding I²C Channel, awaits for the generated reply, and drives the reply-frame contents to the *Framer* via the handler FSM.

- `I²C Channel` and `Register File`: Each of the sixteen possible pairs is associated with a separate part of the UFL logic. These two modules will be described in greater detail later in this Chapter.

8.2.1 The Register File and its I2C Channel

The *Register File* provides the interface between the UFL and the SCAX logic. The architecture of this module is what defines the communication protocol between them; the Register File is essentially a multiplexer/demultiplexer, addressed by the I²C Channel, and driven by the UFL or the I²C Channel, depending on the mode of operation. The SCAX logic provides the ability to either write into or read from a register inside the FPGA fabric. The *address* input of the Register File's mux/demux (in a behavioral representation), will be the $10-$bit I²C address, provided to the I²C Channel by the OPC server. The SCA would use this field to address a I²C device that is connected to it, and propagate the bits via the SCA/SDL lines, as defined by the I²C standard. The SCAX however, uses the same address to drive the mux/demux of the Register File, and access a UFL register under that address. For a write operation, the I²C Channel drives the demultiplexer, and depending on the width of the register-to-be-accessed (maximum 32 bits per register), the corresponding contents that must be propagated are forwarded to the register via the Register File. For a read operation, the I²C Channel drives the multiplexer, and samples the contents of the UFL register locally. The SCAX provides the ability to access up to 1024 $32-$bit UFL registers for each Register File - I²C Channel pair, which can be sixteen in total for each SCAX implementation. An overview of the scheme can be viewed in Figure 8.6.

[5] These are the identifiers of each I²C Channel, as defined by the SCA's protocol.

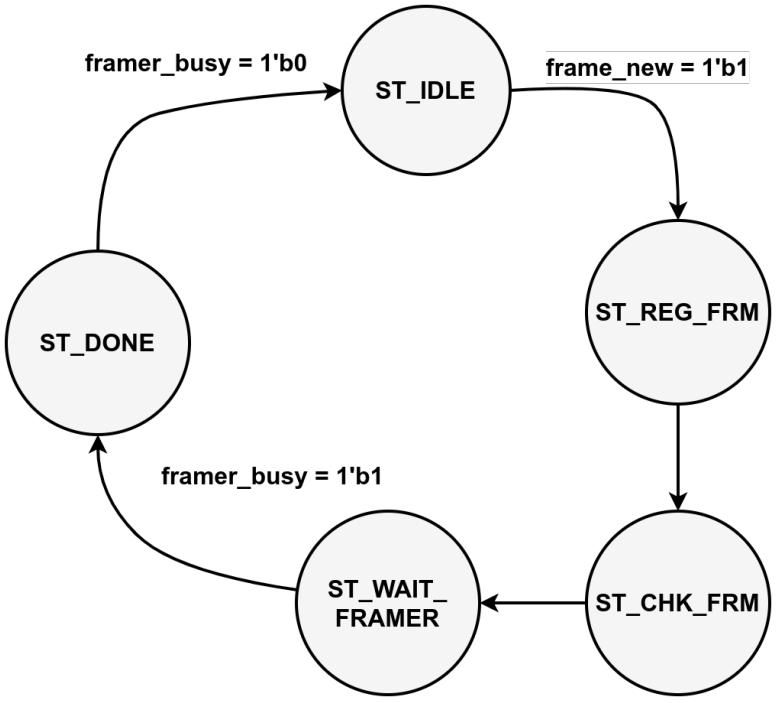


Figure 8.5: *Traffic Handler* state transition diagram. In the *ST_REG_FRM* state, the module registers the inbound frame contents, and in the next state it decides where to send the registered data to. The next two states provide a handshaking mechanism with the *Framer* logic, that sends the reply to the back-end, as received by the addressed sub-module/channel.

8.2.2 I2C Channel Features

Since the I^2C Channel accesses the UFL registers via the Register File by using the latter's sequential elements, the question of crossing clock domains is being risen. The SCAX has a core clock domain, that the user may choose freely upon deployment. The vast majority of the SCAX's logic is under this domain, excluding the FIFO2Elink and Elink2FIFO modules, which operate under the clock of the interface with FELIX, and get their data to and from the SCAX core via CDC FIFOs. Naturally, the data that pass from the I^2C Channel have to be in sync with the destination domain. The user is recommended to place the SCAX under the domain of the majority of the pre-existing logic of the FPGA, so that the I^2C Channel - Register File pair is already within the same clocking scenario as the UFL registers. However, since any design may have multiple domains, the SCAX has to support this use-case as well. In order to address this, each I^2C Channel - Register File pair may be manually switched by the user to *CDC Mode* prior to deploying the SCAX into their design. This overrides the default clocking of these two modules (which is the SCAX core clock), with the clock that drives the UFL registers the pair is tied with. In essence, this option instantiates two CDC FIFOs

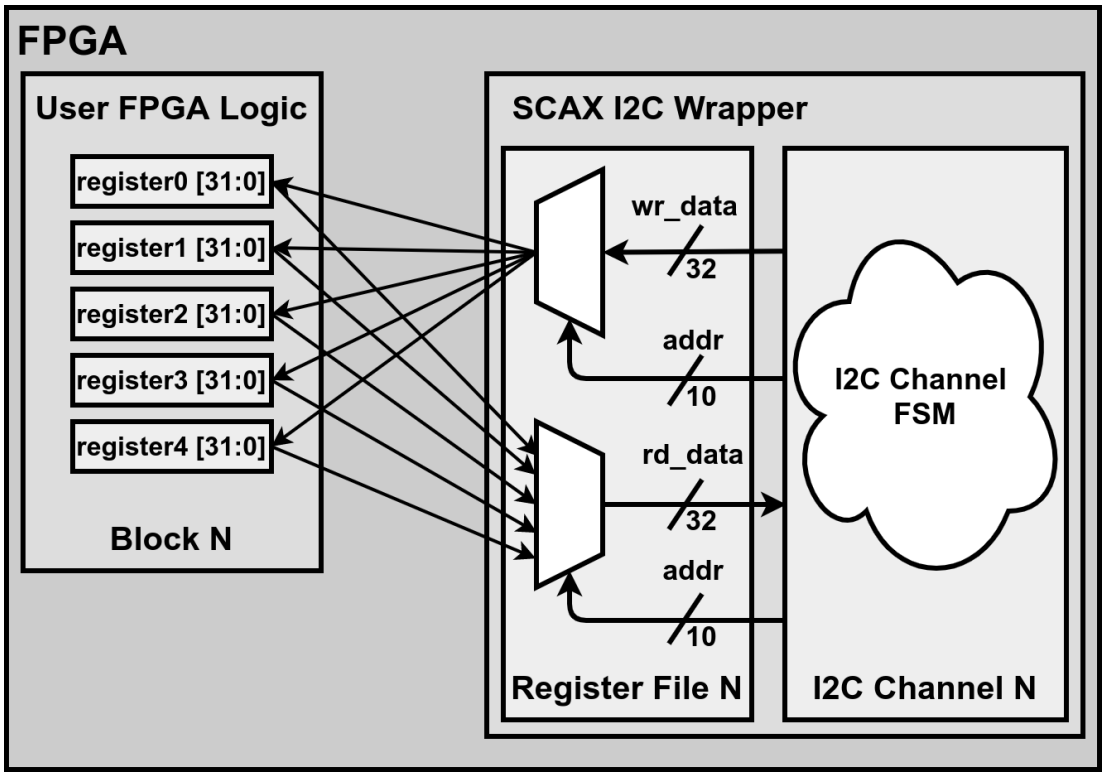


Figure 8.6: The *Register File* allows the *I²C Channel* to access the desired register by using the register address to switch its multiplexer/demultiplexer. The sub-module allows for interfacing with up to 1024 32-bit registers, while the data are presented to the registers in parallel. The scheme implies that each Register File is associated with a specific part of the FPGA's logic.

in the corresponding I²C Channel, one for read and one for write transactions. These FIFOs operate in the UFL register clock and the SCAX core clock domains. Apart from that, the CDC flavor of the I²C Channel operates in a similar fashion as the original one, with the only difference being that the information is always passed through the corresponding FIFO to ensure data integrity and timing closure. The design can be viewed in Figure 8.7.

Apart from providing the ability to access conventional registers implemented in the FPGA's fabric, SCAX allows the user to interface with memory elements as well. The idea behind this functionality is that it deepens the register content space of a given Register File address, as it grants the capability to write a large amount of data into a single address, by utilizing the already available memory elements provided by the FPGA device. To be precise, the user may associate the write and status ports of a FIFO with some addresses of a Register File, thus granting the user writing and reading capabilities into/from the FIFO. Apart from interfacing with a FIFO, the firmware

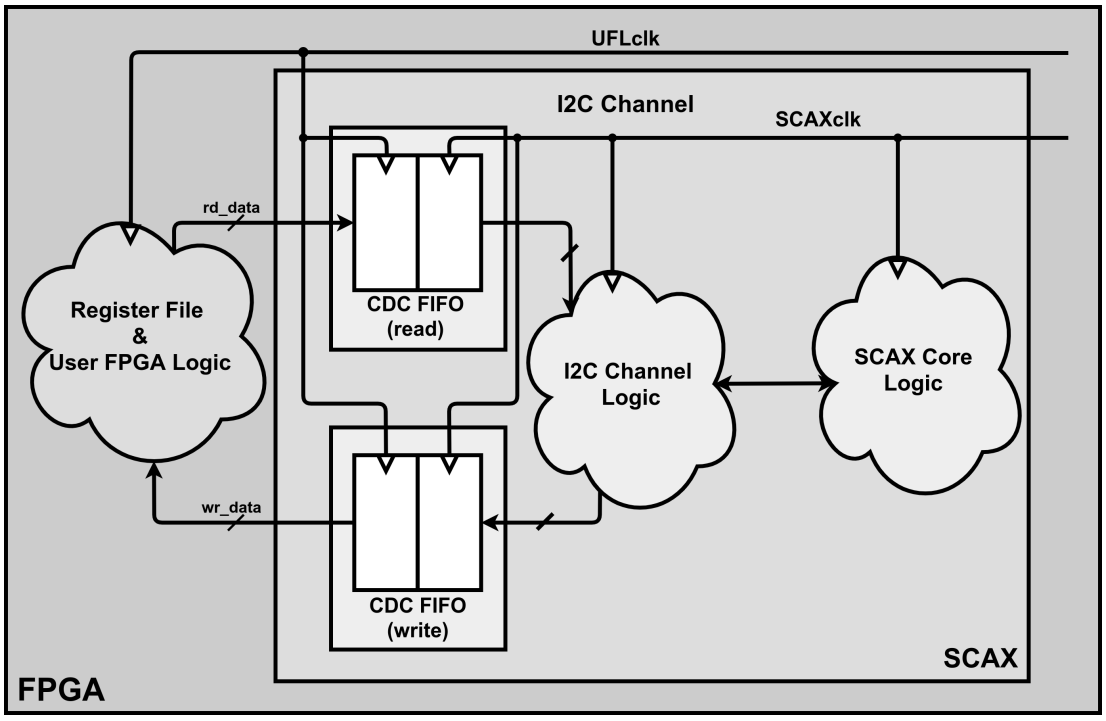


Figure 8.7: The *I²C Channel* implementation when deployed in "CDC Mode". One can see the two CDC FIFOs passing data to and from the SCAX core clock domain *(SCAXclk)* and the UFL - Register File clock domain *(UFLclk)*. When implemented in regular mode, the I²C Channel is essentially the same, minus the two FIFOs, thus retaining the same logic to perform the transactions with the UFL and the SCAX core logic.

package also provides the means to interface with the FPGA's RAM primitives. This can be done via an add-on logic, named *SCAX Memory Controller* (SMC), that connects to both the Register File, and the memory element in question. The user provides a RAM port address via the Register File, and the extension module auto-increments the aforementioned address for each read or write command. This allows the user to access a wide range of the primitive's address space while keeping the amount of transactions with the back-end as low as possible. The scheme can be inspected in Figure 8.8.

8.2.3 Automatic Register File Generation

The SCAX package includes a high-level user interface for the definition and implementation of application-specific registers. This feature provides developers with the ability to provide a database of desired registers in a CSV form. Entries in this database define properties of the registers - their bit width, read and write ability, multiplicity, and whether they correspond to RAM or FIFO primitives.

Upon update of these CSV databases, a Continuous Integration (CI) pipeline triggers

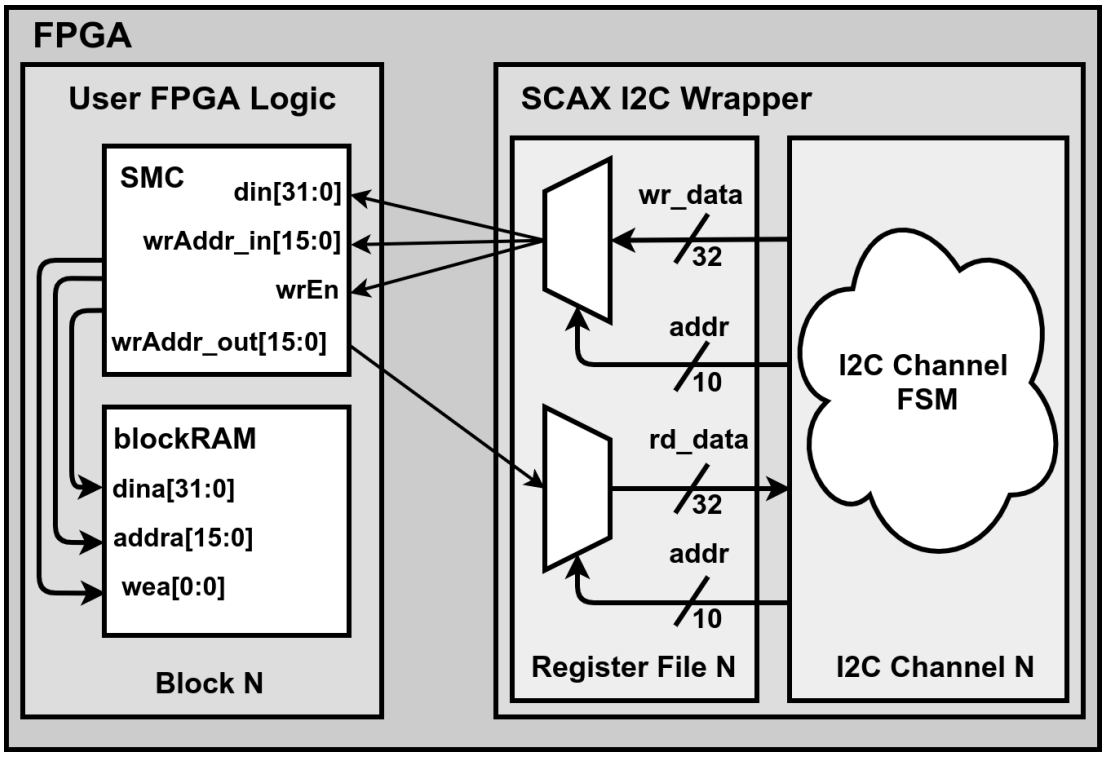


Figure 8.8: The interface between the I^2C Channel and the SMC. Three addresses are tied to the data input, blockRAM address and write-enable strobe, while the current primitive address is tied to a read-only register to the Register File.

the construction of automated VHDL implementations of both the package as well as the register interface itself. Additional code is assembled to define XML configurations for the OPC server, high-level C++ structures for interfacing with the associated OPC client, and LaTeX tables of the register database for automated documentation. Internally, the registers defined in the CSV are used to build a YAML database. Additional registers are created to support interaction with RAM and FIFO registers, and a dedicated handling of trigger registers is implemented. Individual output formats built from this database are defined in templates using the Jinja2 template engine, passed to the `wuppercodegen` package provided by the FELIX project [83].

The automation in the code generation builds an output for every template and for every CSV database in the repository. In this way, prototyping new register configurations is simple and requires only high-level interactions from the users, and creating additional software interfaces is easily done by adding additional Jinja2 templates to the repository. This functionality is included in a separate branch of the SCAX repository *(JinjaYAML)* [82].

8.2.4 Timing Constraints - MultiCycle Paths

As it will become evident, there were several timing constraints considerations during the development process of the SCAX. There are two reasons behind this, and two proposed solutions, described below:

Register File Complexity

The Register File is comprised of a set of combinatorial elements that vary in amount and interconnecting routing complexity, depending on the implementation. For write operations, the module implements a demultiplexer for a write-enable pulse that drives the `CE` pin of the UFL target registers. The data bus is being fanned out to all UFL target registers. Hence, the $10-$bit address only controls the destination of the write-enable pulse. All three signals, write-enable, address, and data are being driven by the I^2C Channel. For read operations, the Register File deploys a multiplexer that accepts all register contents of the UFL registers that is associated with, and the address bus selects the appropriate register-to-route to the I^2C Channel. The latter module in turn determines when it is safe to sample the UFL register contents by issuing a read-enable pulse that drives the `CE` pin of a set of local registers[6]. Hence, it is evident that the multiplexer used for reading will be more complex that the corresponding module used for writing, as the former in a worst case scenario has to select between 1024 $32-$bit buses, whereas the latter has to switch between 1024 single-bit signals. In any case, a *Multi-Cycle Path (MCP)* must be used for both kinds of transactions, as time has to be given for the discrete bus states to stabilize. These MCPs are being controlled by the I^2C Channel logic, which is designed in such a way as to accommodate the signal propagation delay that is being introduced by the combinatorial logic of the Register File. A timing diagram depicting this can be viewed in Figure 8.9. The amount of MCP cycles depends on the implementation. However, a study of the minimum amount of MCP cycles has already been performed, and is described later in this Section.

User Logic Physical Separation

The SCAX is considered to be a standalone IP that may be deployed within the context of any FPGA design that communicates with FELIX. Therefore, the module had to be designed in such as a way as to be as transparent as possible to the surrounding logic that it supports; the addition of the component inside the netlist of a pre-existing design must not hamper the timing closure during the implementation stage. The first step towards achieving this goal has already been described: the SCAX core logic emphasizes on pipelining, by deploying several FSMs that handle the wide data buses. However,

[6] The `D` pins of these registers are driven by the output of the multiplexer.

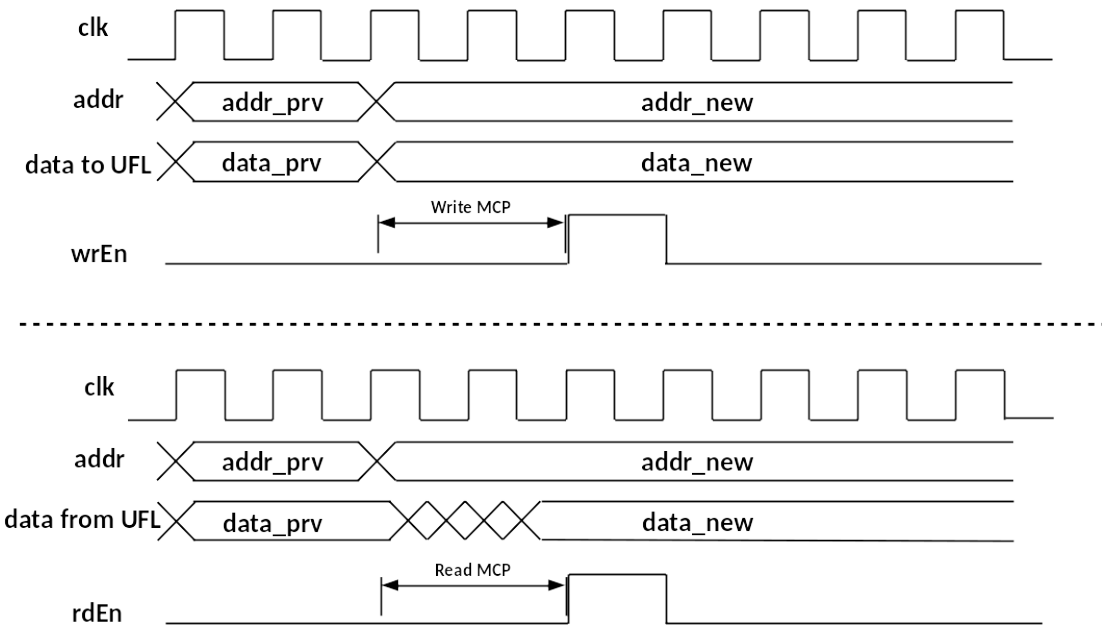


Figure 8.9: Depiction of the MCP logic employed by the I^{2}C Channel to accommodate for the signal propagation delay through the Register File. The write operation can be viewed at the top part of the figure, where the *Write MCP* is the amount of clock cycle delays between the issuing of the data and the issuing of the write-enable pulse driving the `CE` pin of the destination UFL register. Note that the MCP length is arbitrary. The read operation can be viewed at the bottom part of the figure, where the *Read MCP* is the amount of clock cycle delays between the assertion of the address bus and the issuing of the read-enable pulse driving the `CE` pin of a local destination register. The delay between the switching of the address and the stabilization of the data bus is arbitrary, alongside the actual read MCP implemented by the logic of the I^{2}C Channel.

the Register File and the I^{2}C Channel cannot be viewed as standalone modules of the SCAX, and their relationship with the UFL that they interface with has to be taken into account as well.

In a hypothetical scenario where a user had already decided to separate different parts of their logic via FPGA floorplanning, the deployment of the SCAX would create timing issues, since each group of the pre-existing FPGA logic would have to be connected to a specific Register File - I^{2}C Channel pair. In order to tackle this, some freedom of movement to the two aforementioned modules was given, by introducing MCPs on multi-bit buses and pipelines for single-bit control signals on all paths between each I^{2}C Channel and the I^{2}C Router. This additional feature further contributes to the transparency of the SCAX to the surrounding logic. The proposed scheme can be viewed in Figure 8.10.

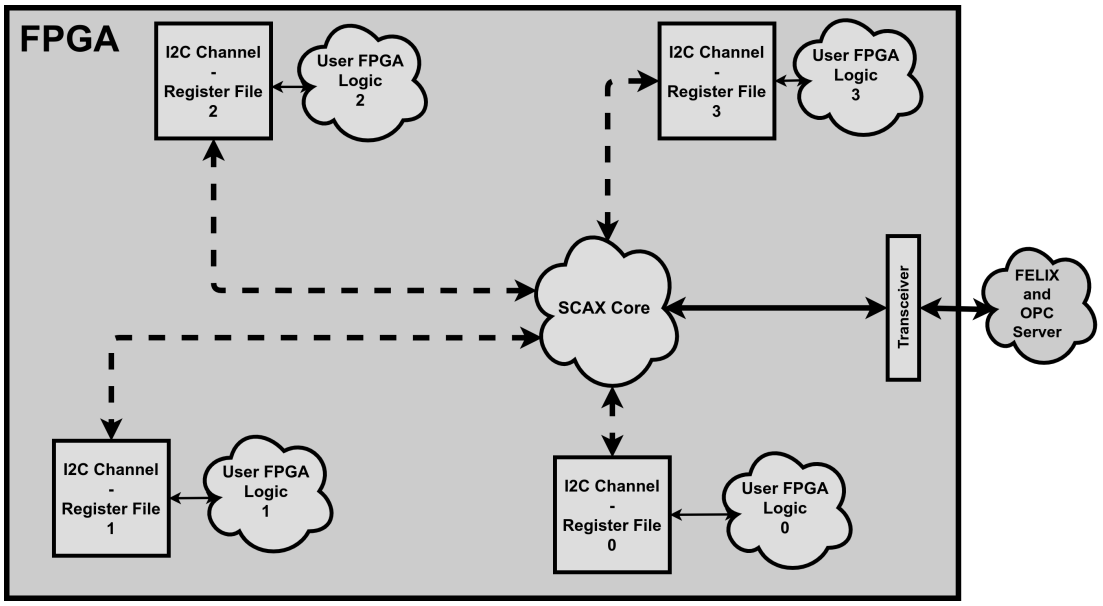


Figure 8.10: Graphical representation of a hypothetical scenario where the developer of the pre-existing UFL logic has physically separated the logic blocks of their design (or the tool itself has already done so automatically). The absence of MCPs between each I^2C Channel and the I^2C Router would impede the timing closure of the design, as the wide buses that carry the data would be unable to propagate to different parts of the FPGA die within only one clock cycle. Therefore, an MCP (dashed lines) was introduced in each of these datapaths, as depicted in the current Figure.

8.2.5 Address Bus Scaling-Up Studies and Resource Utilization

The choice to implement a 10-bit address space depth in each Register File stems from the fact that this addressing mode is already being used by the back-end software to interface with the SCA. However, since the ASIC and the back-end software support other modes of I^2C transactions as well, it would be worth investigating potential modifications in the way the SCAX accesses the UFL registers via its Register File, namely by changing its address space depth. For this reason, the possibility to implement a Register File with different address bus widths was explored. The testing procedure involved implementing a Register File with address bus widths from 8 to 12 bits, and having it interface with as many 32-bit registers as its depth allowed, in a Xilinx® XC7VX690T-2FFG1761C FPGA. The LUT primitive utilization and amount of negative slack for each address space depth can be inspected in Figure 8.11. This study of course, is indicative and does not set an absolute limit of the amount of cycles an MCP must have in the signals connecting an I^2C Channel with the Register File and the UFL. This testing implementation was done under a 320 MHz clocking (standard for the logic of the NSW FPGAs), on an otherwise *empty* chip. The addition of surrounding

logic would reserve logic primitives and routing - thus further hampering the timing closure. However, the indicated amount of timing failure in nanoseconds does set a lower limit on the amount of MCP cycles that must be included in the design and the relevant constraints. The user of the SCAX is responsible for increasing these cycles (essentially adding a safety factor), depending on the occupancy of the logic inside the chip.

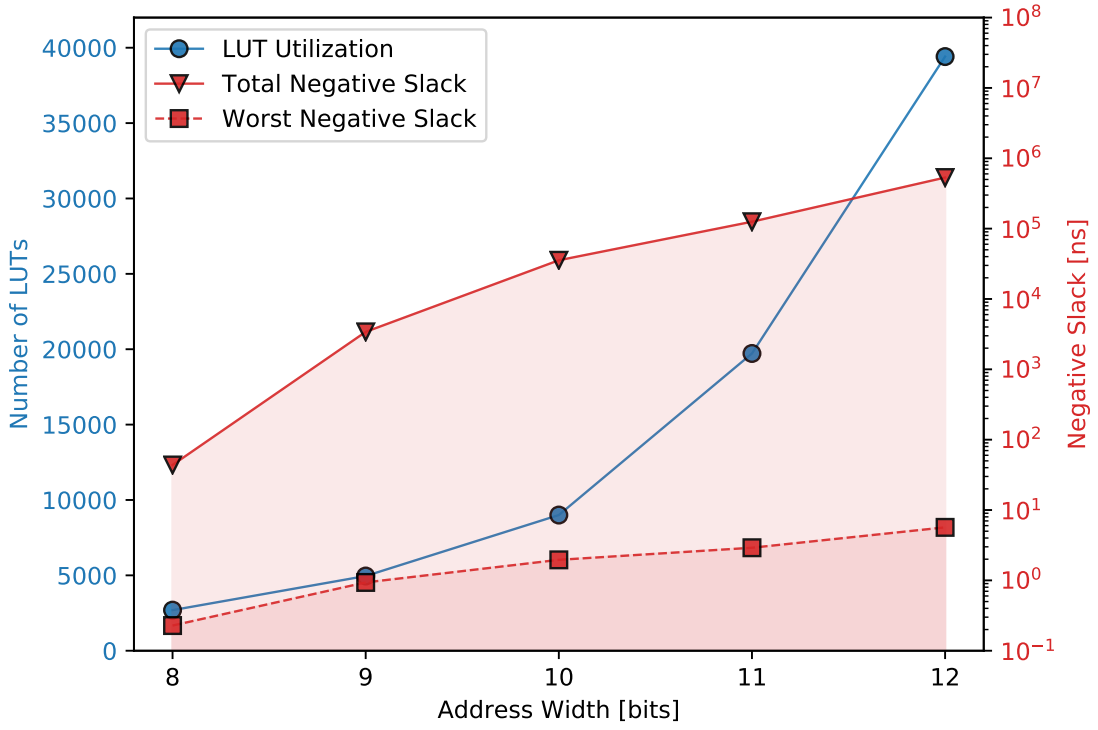


Figure 8.11: Register File resource utilization and design timing failure quantification for several address bus widths, under a frequency of 320 MHz.

8.3 SCAX Implementation and Testing Procedure

In order to adequately test all of the SCAX's core functionalities, a behavioral testbench environment was developed. This involves a top-level design which includes a standard SCAX implementation that interfaces with a handful of user logic registers, alongside a *Dummy OPC Server* connected directly with the SCAX's back-end interfacing ports. The scheme is shown in Figure 8.12.

The top-level expects a set of frame fields and a flag indicating that the issued fields

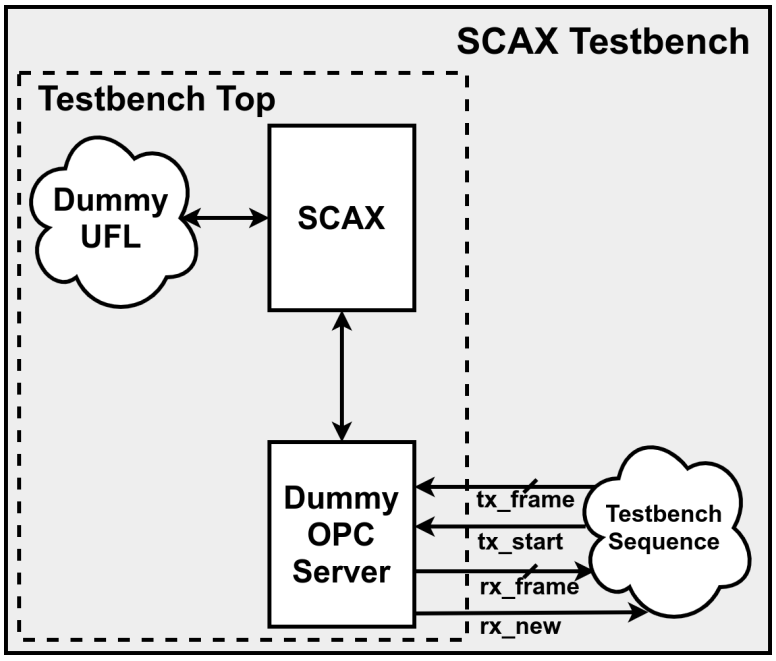


Figure 8.12: Block diagram of the SCAX testbench.

are valid and can be sent (denoted as *tx_frame* and *tx_start* in Figure 8.12 respectively). These fields are the ones dictated by the HDLC/SCA format (see Figure 8.2), and are forwarded to the dummy server, which in essence is just an instantiation of the interfacing modules of the SCAX (see Appendix E for more details). The server sends the message to the SCAX, which replies accordingly. The dummy server dissects the frame, and presents its fields to the associated top-level port, so that the user can inspect its contents.

The actual system that comprises the OPC Server, FELIX and the SCAX, provides the necessary set of tools to inspect the traffic going back and forth between the nodes. Based on the transactions between them, a set of messages that are sent by the dummy server in the testbench is formed, and a set of rules regarding the SCAX's replies is also created. This way, if an error in communication occurs, the user may inspect the SCAX's response in behavioral simulation conditions, provided that the contents of the frame originating from the server is known, thus allowing for an easy way of diagnosing issues in the SCAX's logic and in the way it handles the replies sent to the server.

8.3.1 Field Implementation and Stress-Testing

In order to adequately test the SCAX's functionalities, an extreme situation was chosen, where the module was deployed in a Xilinx® XC7VX690T-2FFG1761C FPGA. The

SCAX had two active I^{2}C Channels, connected with two fully-implemented Register Files interfacing with 1024 32-bit registers each. The two I^{2}C Channels were physically separated to emulate the scenario described above. Also, one I^{2}C Channel was deployed in *CDC Mode*, in order to validate the transactions between the CDC FIFOs of this special version of the component. The implemented design can be inspected in Figure 8.13.

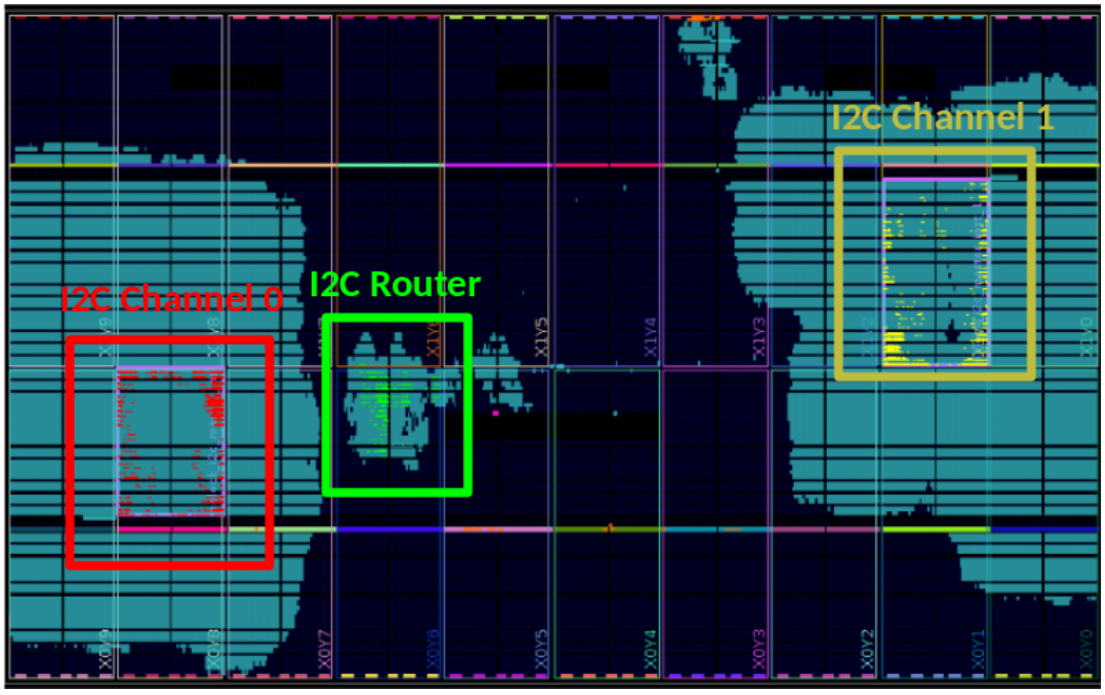


Figure 8.13: SCAX implementation in the FPGA of a Xilinx® VC709. The I^{2}C Router is highlighted in the middle of the Figure, whereas the two I^{2}C Channels are in two different areas of the die, to emulate a physical separation that may have been imposed by the designer of the pre-existing logic or the tool itself. Most of the active cells around the two I^{2}C Channels correspond to the combinatorial logic of the associated `Register File` block and the UFL registers themselves.

Two tests were performed to verify the hardware implementation: first, an OPC transaction stress-test was used to test the back-end interfacing. During this, 400 million frames were sent to the SCAX at a rate of 11 kreq/s. The server evaluated all SCAX reply frames, and no errors were found. Also, in order to test the MCPs between the I^{2}C Router and the two I^{2}C Channels that were mentioned in the previous section, as well as the MCPs between each I^{2}C Channel and the UFL via the combinatorial-logic-heavy Register File, a mass read/write test from/into the registers was used. During this procedure, all 1024 registers of a given Register File were written into, in random order. This randomization was imperative, as the random switching of the address bus

of the module's multiplexer/demultiplexer (see Figure 8.6) ensured that the path would be verified correctly. After writing into all registers, their values were read back, again in nondeterministic order, and were checked off-line by a software routine for integrity. The tests showed no errors over several million transactions. The procedure is depicted in Figure 8.14.

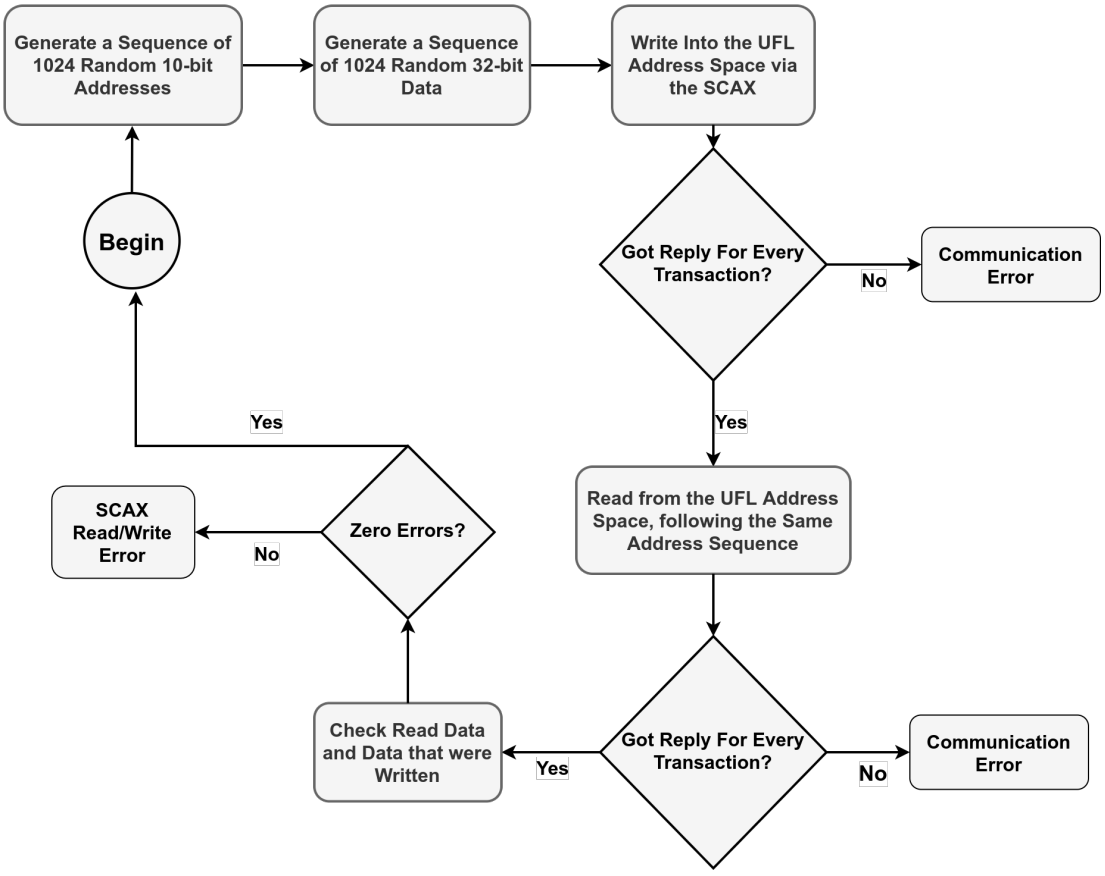


Figure 8.14: Flowchart of the software logic that tests both the low-level communication between the SCAX and the OPC server (by throwing communication errors if no reply was received from the SCAX after a request), and the integrity of the data that are being forwarded to and from the UFL via the SCAX. This software serves as a simple OPC client that connects to the OPC server, which in turn communicates with the SCAX via FELIX, as depicted in Figure 8.3.

Utilization-wise, the SCAX core logic uses a modest 6600 sequential and 3500 combinatorial elements. These correspond to the 0.78 % and 0.76 % of the available resources of a Xilinx® XC7VX690T-2FFG1761C FPGA respectively. Naturally, as the user adds more registers to a Register File, the combinatorial element utilization will increase. For interfacing with 100 registers, an extra 800 LUT overhead is expected, for 400 registers the number increases to 3400 LUTs, while a fully-implemented Register

File utilizes about 10000 LUTs, which still corresponds to 3 % of the resources of the aforementioned package (see also Figure 8.11).

The GBT-FPGA Implementation

As mentioned above, for the SCAX testbench, the core was implemented in a Xilinx® XC7VX690T-2FFG1761C, which is the main FPGA of a VC709 evaluation board. In order to communicate with FELIX, the GBT-FPGA wrapper was deployed, as depicted in Figure 8.3. The process mainly involved migrating the design from the GBT-FPGA repository for the Xilinx® VC707 evaluation board provided by the GBT group [84], into the device that was readily available for the current application [85], and choosing a functional clocking scheme (described below).

The evaluation board in question communicates with FELIX using the transceiver of the GBT-FPGA design, via a bidirectional optical link at a rate of 4.8 Gb/s. Every 25 ns, the GBT frame (see Figure 7.1) is being transmitted from one end to the other, and the receiver implements a Reed-Solomon [86] decoder to check for any errors in the frame via the Forward Error Correction (FEC) field. The total overhead of the header and trailer of the frame results in an effective user bandwidth of 3.36 Gb/s.

The SCAX communicates with the GBT-FPGA core, and is tied to some parts of the frame field in order to send and receive data to and from FELIX. If, for instance, the Elink2FIFO and FIFO2Elink modules of the SCAX are implemented using a data rate of 80 Mb/s, two bits of the received frame will be driven to the Elink2FIFO input, and two bits of the GBT frame that is transmitted to FELIX is tied to the FIFO2Elink output. For a 160 Mb/s implementation, the bus width is four bits, and for a 320 Mb/s, the bus is eight-bit-wide. Finally, the two cores use either 8b10b or HDLC encoding and decoding when sending or receiving the data. Naturally, FELIX must have the same configuration when communicating with the SCAX, both in terms of data rate and line encoding.

After implementing the transceiver into the FPGA, the most important step towards a successful communication with the other node (for this case, FELIX), is to ensure that the system is under a common reference clock. In the NSW electronics scheme, the device that will provide the clock to the system is FELIX, as it will be recovering it from the TTC optical link [57]. The GBTx then recovers this clock from the optical link with FELIX. The ASIC uses an on-package crystal oscillator during start-up in order to recover the data and the clock from the link, and uses that recovered clock to send the GBT frame back to FELIX. It also uses the same clock to drive the other devices it is connected to, that transmit their data serially to the GBTx using that same frequency. This results in a fully synchronous (also referred to as *isochronous*) system.

One way to have the FPGA device that hosts the GBT-FPGA and the SCAX be in the same domain with FELIX is using a clock output from a GBTx that is under the same system. This is achieved easily by driving a clock output of the GBTx into a jitter cleaner (imperative, as the line rate requires a high quality reference), which in turn drives the GBT-FPGA's transceiver reference clock input. This scheme is depicted in Figure 8.15.

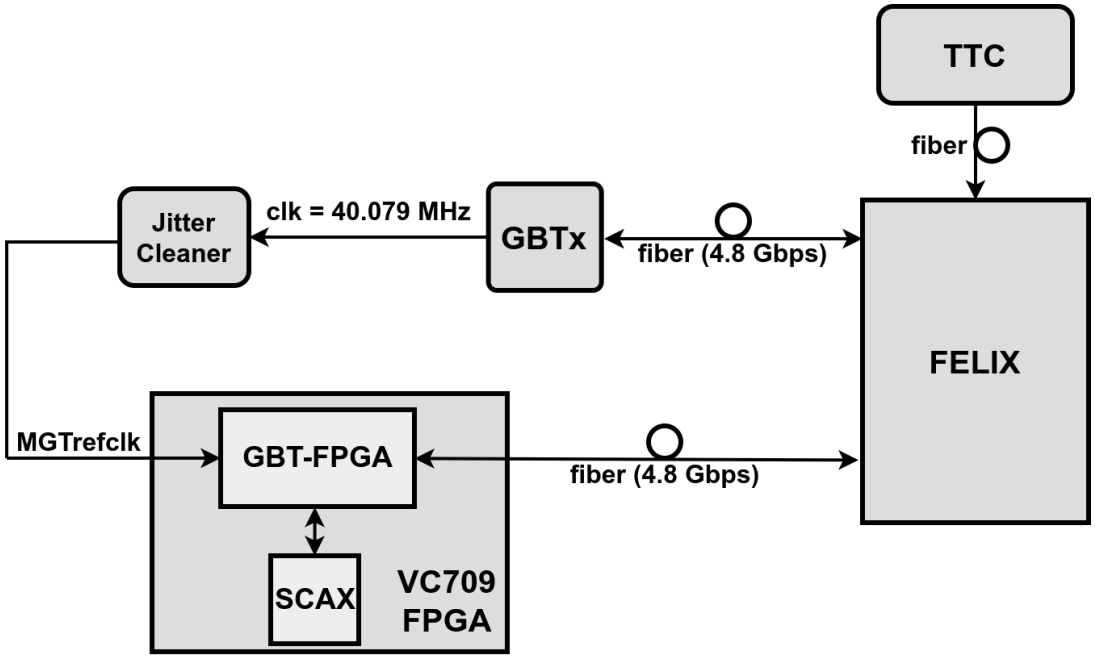


Figure 8.15: Block diagram of the clocking scheme when using the GBTx clock output as an input to the transceiver's reference clock. This connectivity ensures proper communication with FELIX.

This approach emulates that of the NSW electronics scheme, in the sense that the GBTx provides reference clocking to all other devices using its clock recovery capabilities. As mentioned above, the ASIC uses an on-package crystal oscillator as an input to its Clock and Data Recovery (CDR) logic, that recovers clock and data from the fiber connection with FELIX. The aforementioned oscillator's frequency must be very close to that of the reference frequency of the transceiver that the GBTx is connected to. Having that in mind, a different scheme can be used in the setup, that does not use a GBTx device, but mimics its behavior. The setup involves a high-quality frequency generator, that drives the GBT-FPGA's reference clock input *(MGTrefclk)* via a jitter attenuator. That frequency is either equal, or a multiple of the system's frequency of 40.079 MHz[7]. The

[7]This is the clock that FELIX recovers from the TTC input, and it is what is being used in ATLAS as a reference.

FPGA's transceiver uses that reference clock to drive its CDR circuitry, which in turn outputs the recovered clock *(RXuserclk)* into its fabric. The latter is being driven out of the FPGA, into the input of the jitter attenuator, which can automatically switch over to the recovered clock, if it has been configured to do so. After switching, the GBT-FPGA uses the recovered clock as a reference clock input, and is now in full sync with FELIX and the system. This clock-switching method has also been tested thoroughly and was what was being used when running the stress-tests of the SCAX. The scheme can be viewed in Figure 8.16.

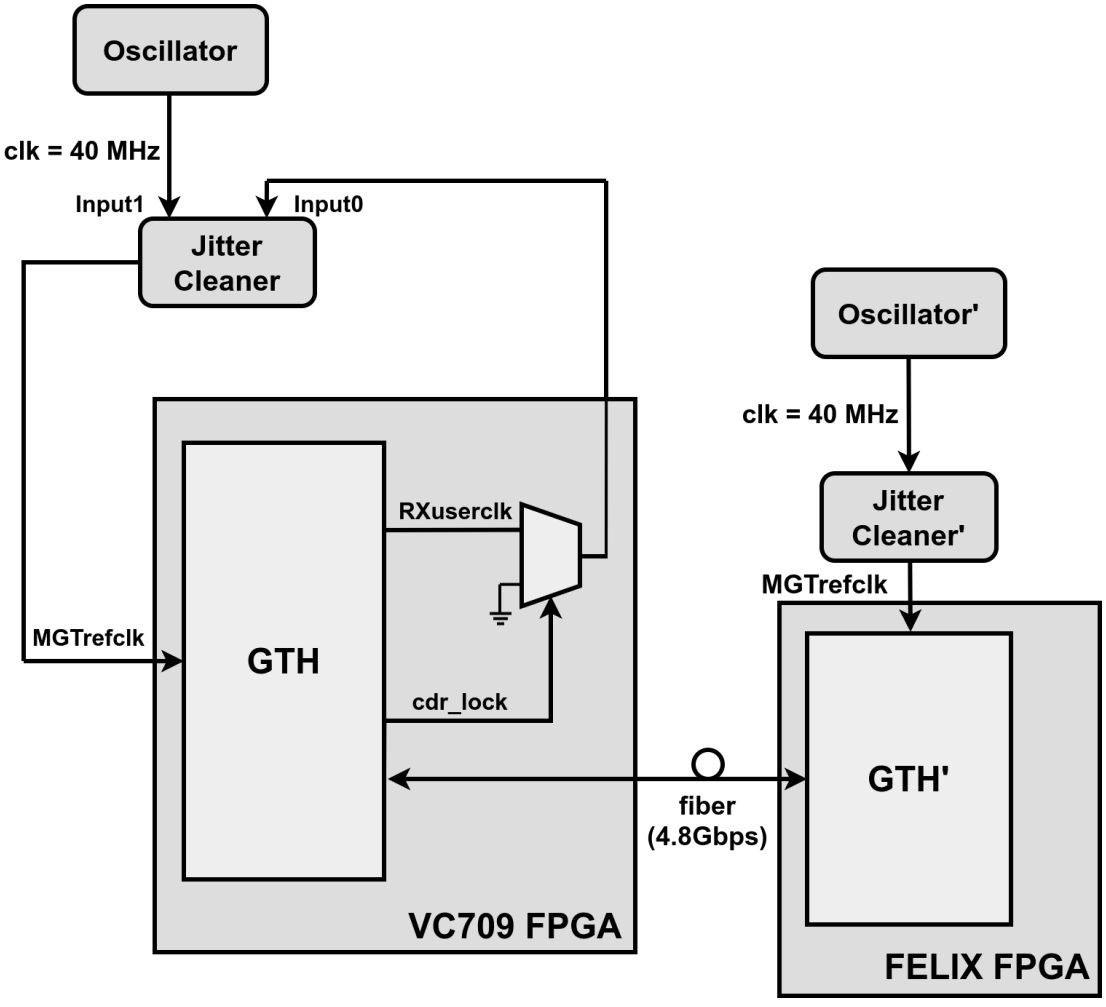


Figure 8.16: Block diagram of the clocking scheme when using an independent high-quality oscillator to recover the clock from FELIX and then feed it back into the transceiver that performed the recovery. The jitter cleaner is configured to automatically switch over to the first input, if a clock of predefined frequency is driven there. The recovered clock is being fed to that input once the CDR circuitry of the transceiver locks to the incoming datastream of FELIX.

Conclusions

In this Chapter, the architecture and functionality of the SCA eXtension was described [82]. Emulating the interface of the SCA ASIC's with the front-end devices and the back-end electronics, the main feature of the SCAX is that it can be used by any FPGA device that communicates with FELIX, and allows the user to access the configuration parameters of the pre-existing logic. It is completely transparent to the OPC server that handles the SCA traffic via FELIX [66], thus allowing the employment of the well-established software infrastructure used for the detector's slow control to monitor the status of the FPGA and write into its register space. A flexible design, SCAX has been developed in such a way as to be transparent to the surrounding logic of the FPGA that it is instantiated in, via MCPs used in the critical paths of the core. All of its functionalities have been tested thoroughly in actual FPGA implementations at a clock frequency of 320 MHz, while the included software package that allows the user to automatically generate the necessary files that vary depending on the actual application, further enhance the package's ease-of-use. These facts, alongside the by-design compatibility with the back-end OPC server, and the module's general adaptability, deem the SCAX ideal for configuration of FPGA-based systems that use FELIX for their communication with the back-end.

For more information, the reader is also referred to an SCAX-related proceeding that was published by the writer on behalf of the ATLAS collaboration [81]. At the time of writing of this dissertation, a journal publication about the subject had just been published as well [80].

Appendix A

Acronyms

ADC Analog to Digital Converter

ADDC ART Data Driver Card

ALICE A Large Ion Collider Experiment

ALTI ATLAS Local Trigger Interface

ALU Arithmetic Logic Unit

ART Address in Real Time

ASIC Application-Specific Integrated Circuit

ATCA Advanced Telecommunications Computing Architecture

ATLAS A Toroidal LHC ApparatuS

BC Bunch-Crossing

BCID Bunch-Crossing Identification

BGA Ball-Grid Array

BW Big Wheel

CA Charge Amplifier

CB Connection Block

CDC Clock Domain Crossing

CDR Clock and Data Recovery

CERN Conseil Europeen pour la Recherche Nucleaire, or European Organization for Nuclear Research

CI Continuous Integration

CKBC Bunch-Crossing Clock

CKDT Data Clock

CKTK Token Clock

CKTP Test-Pulse Clock/Strobe

CLB Configurable Logic Block

CMOS Complementary Metal-Oxide-Semiconductor

CMS Compact Muon Solenoid

CMT Clock Management Tile

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CSC Cathode Strip Chamber

CTF Clock and Trigger Fanout

CTP Central Trigger Processor

DAC Digital to Analog Converter

DAQ Data AcQuisition

DCS Detector Control System

DDF Delayed Dissipative Feedback

DMA Direct Memory Access

DRP Dynamic Reconfiguration Port

DUT Device Under Test

EDA Electronics Design Automation

EMAC Ethernet Media Access Controller

ENC Equivalent Noise Charge

EOF End of Frame

FCS Frame Check Sequence

FEC Forward Error Correction

FELIX Front-End LInk eXchange

FIFO First-In First-Out

FMC FPGA Mezzanine Board

FPGA Field-Programmable Gate Array

FSM Finite State Machine

(S)GMII Serial Gigabit Media Independent Interface

GBT Giga-Bit Transceiver

GUI Graphical User Interface

HDL Hardware Description Language

HDMI High-Definition Multimedia Interface

HDLC High-Level Data Link Control

HLT High-Level Trigger

ICMP Internet Control Message Protocol

IEEE Institute of Electrical and Electronics Engineers

I/O Input/Output

IP Intellectual Property / Internet Protocol

IPv4 Internet Protocol version 4

JTAG Joint Test Action Group

LHC Large Hadron Collider

L0(A) Level-0 (Accept)

L1(A) Level-1 (Accept)

LAN Local Area Network

L1DDC Level-1 Data Driver Card

LM Large Module

LS2/3 Long Shutdown 2/3

LSB Least Significant Bit (Byte)

LUT Look-Up Table

LVDS Low Voltage Differential Signaling

miniSAS mini Serial Attached SCSI

MO Monitor Output

MDT Monitored Drift Tube

Micromegas (MM) Micro-mesh Gaseous Structure

MAC Media Access Control

MAN Metropolitan Area Network

MCP Multi-Cycle Path

MMCM Mixed-Mode Clock Manager

MMFE Micromegas Front-End

MSB Most Significant Bit (Byte)

MSPS Mega Sample Per Second

NIM Nuclear Instrumentation Modules

NSW New Small Wheel

OPC Open Platform Communications

OSI Open Systems Interconnect

PCIe Peripheral Component Interconnect Express

PCS/PMA Physical Coding Sublayer/Physical Medium Attachment

PDO Peak Detector Output

PLL Phase-Locked Loop

PMT Photo Multiplier Tube

PS Proton Synchrotron

R&D Research and Development

RAM Random Access Memory

RMS Research Mean Square

ROC Read-Out Controller

ROM Read-Only Memory

RPC Resistive Plate Chamber

SB Switch Box

SCA Slow Control Adapter

SCAX Slow Control Adapter eXtension

SCSI Small Computer System Interface

SCT Semi Conductor Tracker

SET Single Event Transient

SEU Single Event Upset

SL Sector Logic

SM Small Module

SMA SubMiniature version A

SMC SCAX Memory Controller

SNR Signal to Noise Ratio

SOF Start of Frame

SPS Super Proton Synchrotron

SREJ Selective Reject

SSTL Stub Series Terminated Logic

TAC Time to Amplitude Converter

TCP Transmission Control Protocol

TDO Time Detector Output

ToT Time-over-Threshold

SET Single-Event Transient

SEU Single-Event Upset

SPI Serial Peripheral Interface

SSH Secure Shell

sTGC small-strip Thin Gap Chamber

SW Small Wheel

swROD Software Read-Out Driver

TGC Thin Gap Chamber

TMR Triple Modular Redundancy

TP Trigger Processor

μTPC micro Time Projection Chamber

TRT Transition Radiation Tracker

TTC Trigger Timing and Control

TTL Transistor-Transistor Logic

UDP User Datagram Protocol

UFL User FPGA Logic

USA15 Underground Service Area 15

VCO Voltage-Controlled Oscillator

VERSO VMM Readout System Ethernet Readout Software

VHDL Very High Speed Integrated Circuit Hardware Description Language

VME Versa Module Europa

VRS VMM Readout System

VSB VMM Readout System Supervisory Board

XML Extensible Markup Language

Appendix B

Communication Protocols

B.1 Ethernet - UDP Protocol

Ethernet is a communication protocol used locally (Local Area Networks - LAN), or at a larger scale (Metropolitan Area Networks - MAN). It first appeared in the mid 70s [12] and quickly emerged into a prominent position, overshadowing other types of technology developed for the same applications, due to its simplicity and high throughput capabilities. Ethernet is part of the second layer of the Open Systems Interconnection (OSI) model[1] which is called *Data Link Layer*. The logic implementing the said protocol puts together the Ethernet frame, and forwards it one layer down, to the *Physical Layer*, which is on the lowest level. The general structure of the Ethernet frame can be viewed in Figure B.1. The *Network* and *Transport* Layers exist on top of the Link Layer, which completes the scheme that is of interest for this dissertation: Transport → Network → Link → Physical. The Transport Layer protocol that was implemented in the FPGAs of this work is the *User Datagram Protocol (UDP)*.

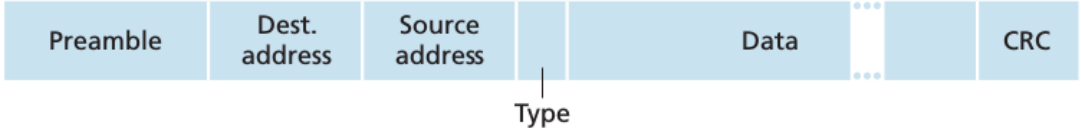


Figure B.1: Structure of the Ethernet frame [12]. The *Data* field, also called *Payload*, contains data that are part of higher layers of the OSI model, like IPv4 for example (see Figure B.2).

[1] The OSI model describes the different functions that a computational system must apply in order for it to interface with other systems. The OSI model dictates a certain hierarchy, thus dividing the different functionalities into distinct *layers*.

The Destination Address (denoted as *Dest. Address* in Figure B.1), is the so-called Media Access Control (MAC) Address of the LAN node the packet is directed to. It is six bytes long and is more or less tied with the hardware (e.g. a network card[2]). Similarly, the *Source Address* is the address of the node from where the frame was transmitted. The *Type* field denotes the so-called *EtherType* of a frame. One commonly used type is called IPv4, associated with the value 0x0800 in that field. The *Data* can be up to 1500 bytes in length, and contains the fields of the frame (or datagram) assembled on the Network Layer. As mentioned before, this is usually an IPv4 datagram (at least for the applications discussed here.). Finally, the Cyclic Redundancy Check (CRC) field is four bytes long, and allows for detection of corrupted data within the entire frame as received on the receiver side. The standard states that the CRC[3] value is computed as a function of the protected MAC frame fields: source and destination address, length/type field, MAC client data and padding (that is, all fields except the FCS) using the left shifting CRC32 BZIP2.

The Ethernet frame described above is part of the Link Layer. On the Network Layer, the IPv4 datagram is assembled, and is encapsulated inside the *Data/Payload* field of the Ethernet frame. On the Transport Layer, the UDP datagram is assembled, and is encapsulated inside the *Data/Payload* field of the IPv4 datagram. The reader can inspect the fields of the IPv4 and UDP datagrams in Figure B.2.

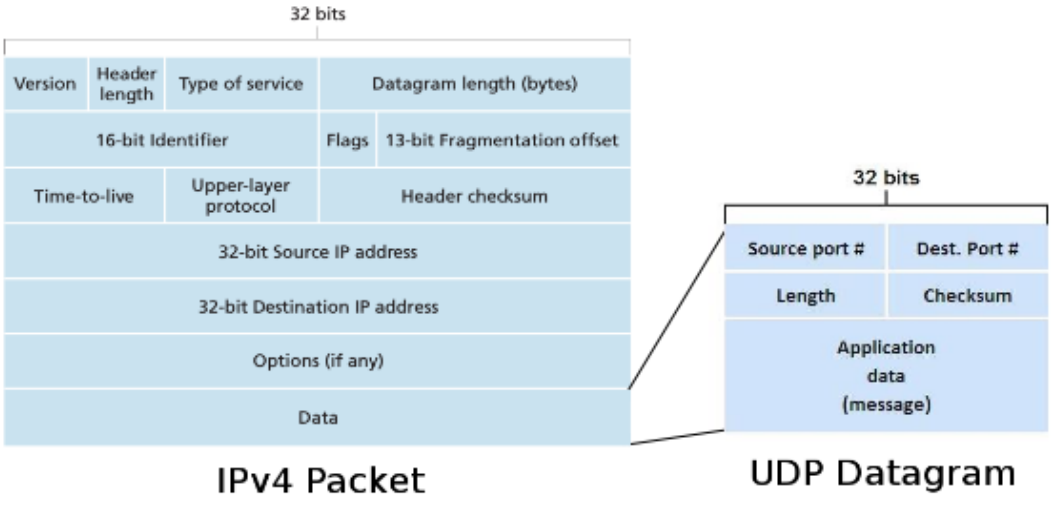


Figure B.2: Structure of an IPv4 datagram (encapsulated inside the *Data* Field of an Ethernet frame) and of a UDP Datagram. The UDP Datagram's contents are inside the IPv4 payload.

[2] As opposed to the *IP Address*, which is part of the Internet Protocol version 4 (IPv4) datagram - see Figure B.2

[3] Also called Frame Check Sequence (FCS).

The IPv4 header has several fields which define the behavior and the packet type. The *Datagram Length* for instance, is sixteen bytes long, which implies that a packet can be 65535 bytes long. All hosts are required to be able to reassemble datagrams of size up to 576 bytes, but most modern hosts handle much larger packets. Sometimes links impose further restrictions on the packet size, in which case datagrams must be fragmented. Fragmentation in IPv4 is handled in either the host or in routers. A receiver can determine if a packet is part of a fragmented group, by using the *Identifier* field, and deduce which part of the fragment the received frame is from, by the contents of the *Fragmentation Offset* field. Another interesting field is the *Time-to-live* (TTL). This field limits a datagram's lifetime. It is specified in seconds, but time intervals less than 1 second are rounded up to 1. In practice, the field is a hop count - when the datagram arrives at a router, the router decrements the TTL field by one. When the TTL field hits zero, the router discards the packet. The *Upper Layer Protocol* denotes which protocol the *Data* field of the IPv4 datagram is part of. Two possible values can be 0x01 for the Internet Control Message Protocol (ICMP), or 0x11 for UDP. Finally, the 32-bit addresses for the source and the destination of the datagram, have a similar functionality as the associated fields of the Ethernet frame, with the difference being that they are easier to manipulate at the level of the Operating System, as opposed to the MAC Address, which is tied with the hardware.

The UDP datagram which is encapsulated in the IPv4 data field, is a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking functionalities, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection. UDP is suitable for purposes where error-checking and correction are either not necessary or are performed in the application. For the needs of the FPGA-related projects of this project, UDP was deemed sufficient, as error-checking is being performed on the software side, which uses fields inside the UDP payload that denote the packet sequence number to report any dropped messages in the network.

B.2 8b10b Encoding

8b10b is a line code that maps 8-bit words to 10-bit symbols to achieve DC-balance and bounded disparity, and yet provide enough state changes to allow reasonable clock recovery, at the expense of bandwidth overhead. This means that the difference between the counts of ones and zeros in a string of at least 20 bits is no more than two, and that there are not more than five ones or zeros in a row. The encoding is widely

used in most serial communication schemes (e.g. FireWire, SATA/SAS, Ethernet, HDMI). The method was first described in 1983 by two IBM researchers in a dedicated publication [33]. The basic philosophy of the scheme is based on the fragmentation of transmitted data into bytes, which are subsequently encoded into 10-bit words (see Figure B.3. This transformation results in a serial stream where the amount of high and low bits is more or less balanced [34]. The encoding algorithm ensures that no more than five subsequent high or low bits are transmitted, and the total number of ones and zeros in a stream of twenty bits in a row will differ by two at most. This difference is known as the *Running Disparity* (RD). Each byte has actually *two* 10-bit words that it can be mapped into, and these two 10-bit words are complementary with respect to each other. One of them has an RD of $+1$, while the other one has $RD = -1$. If, for instance the transmitter wants to send two bytes, then the first byte will be mapped into a word with positive disparity, while the next byte will be encoded into a word with negative disparity. This scheme ensures that the long-term ratio of ones and zeros transmitted is exactly 50%, thus resulting in a *DC-free* transmission line. This is a highly desirable feature, since it allows the datastream to be transmitted through a channel with a high-pass filtering characteristic. This is also called *DC Balancing*, and it is used in high-rate communication schemes.

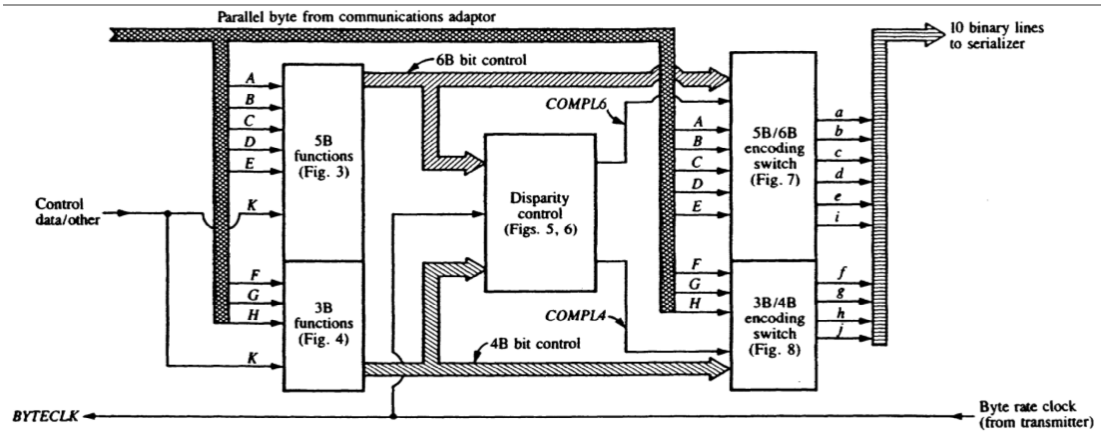


Figure B.3: Diagram of an 8b10b encoder. The encoder receives an 8-bit input (ABCDEFGH), and outputs a 10-bit parallel signal (abcdeifghj). This signal is forwarded to a serializer, which transmits the word bit-by-bit on the transmission line [33].

The other reason behind 8b10b encoding is that the continuous transitions in the voltage levels, make the process of *Clock Recovery* easier for the receiver. In most modern communication schemes, the receiving end accepts only the datastream, and uses that to derive (or *recover*) a digital clock, that will synchronize the register that samples that same stream of data. Finally, to further ease the communication between transmitter and receiver, the 8b10b standard has several special characters defined

(called *K Characters* or *Comma Characters*), that are used for control functionalities (e.g. encode commands such as `ABORT`, `RESET`) and for word boundary alignment. For the latter case, when the transmitter is not sending any user data, it sends a constant stream of `IDLE` characters instead. This is the 0xbc character in the byte space, but in the 10-bit space, it is represented by 0011111010 and 1100000101[4]. The receiver deserializes the stream, and upon recognizing the pre-defined `IDLE` character, knows where the word boundaries are, and when exactly to pass 10 of the received bits into its decoder. There are also two additional and equally important characters defined by the scheme, called `SOP` and `EOP`, that define the start and end of a user payload packet respectively.

All these functionalities have deemed 8b10b ideal for data communication schemes, and it is the reason why it is a recurring theme in modern systems and in this dissertation as well. The only drawback of using 8b10b encoding, is the fact that it adds a 20% overhead in the effective user bandwidth. The solution is to use other line code schemes, with similar characteristics, but with a better useful/"useless" bit ratio, such as 64/66b encoding (approximately 3% bandwidth efficiency), or use a *scrambling/descrambling* scheme between two communicating nodes, that rearrange the bits in such as a way as to achieve DC balancing.

[4]Note that there are two words, complementary to each other - they have negative and positive disparity respectively. They are also denoted as `K28.5`.

Appendix C

The MMFE1 Board

The MMFE1 board [8] houses a single VMM and an Artix®-7 FPGA by Xilinx®. It is being used in several setups throughout this dissertation, including the VMM Front-End readout firmware (see Chapter 4), both in the digital and in the analog readout scheme, and in the VMM Readout System (see Chapter 6) [44].

The board is shown in Figure C.1. Communication with the software host (see Appendix D) is established using the FPGA and a commercial network Ethernet physical interface (PHY) at speeds of 10 or 100 or 1000 Mb/s. A power distribution circuit utilizes step-down regulators with an input voltage ranging from 3.4 up to 42 V and an output ripple of less than 10 mV peak-to-peak. The four power rails of the VMM (*Vddp*, *Vdd*, *Vddad* and *Vddd*) are independently powered by different Low Drop-Out (LDO) regulators to mitigate any ripple effects. Three LEMO connectors provide access to the monitoring, peak-detector and time-detector outputs (MO, PDO, TDO respectively) of the on-board VMM. The board has the ability to output the ART signal, and receive an external trigger signal.

The architecture of the MMFE1 board is shown in Figure C.2. The FPGA is accessible through a standard JTAG connector for programming and monitoring. The configuration file is stored in a flash memory which is accessible over the Serial Peripheral Interface (SPI) protocol. A second memory (EEPROM) is used to store the Media Access Control (MAC) and network addresses of the board, each of which are configurable. The EEPROM is accessed by the FPGA via the I^2C protocol, thus allowing the on-board FPGA to dynamically reconfigure its network address, a crucial aspect for the scalability of the system. A μHDMI connector provides access to

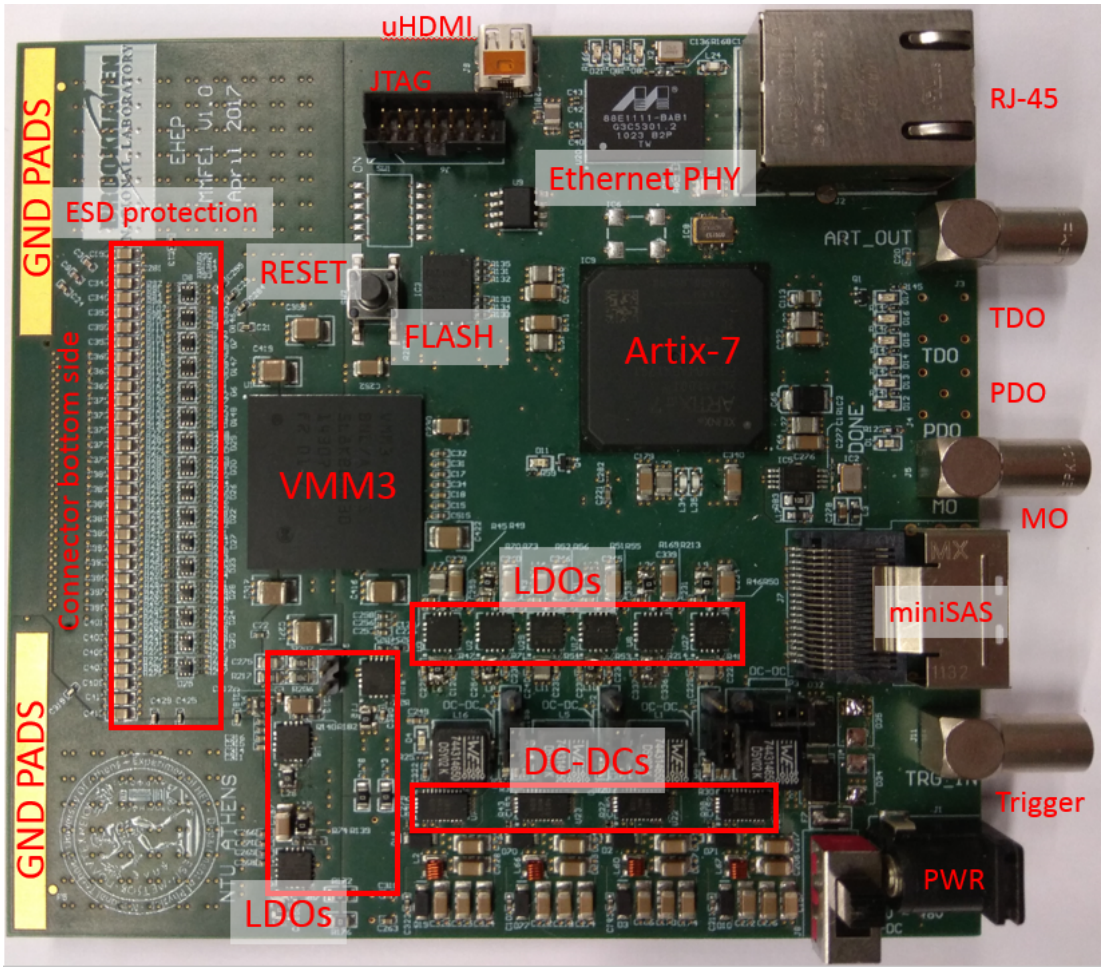


Figure C.1: The MMFE1 board.

additional external signals such as a reference clock, trigger and reset. A miniSAS[1] connector can be used to interface externally with other boards, such as the Level-1 Data Driver Card (L1DDC) [8], that has been developed for the NSW upgrade [7].

The input protection of the VMM is implemented using steering diodes in a rail-to-rail configuration along with series resistors. A number of front-end boards based on the same architecture as the one described here have been fabricated to interface with various types of detectors. The MMFE1 board, for example, was fabricated to validate the functionality of the VMM ASIC using $10 \times 10\,\text{cm}^2$ Micromegas [30] prototype chambers.

[1] Commercial mini Serial Attached Small Computer System Interface.

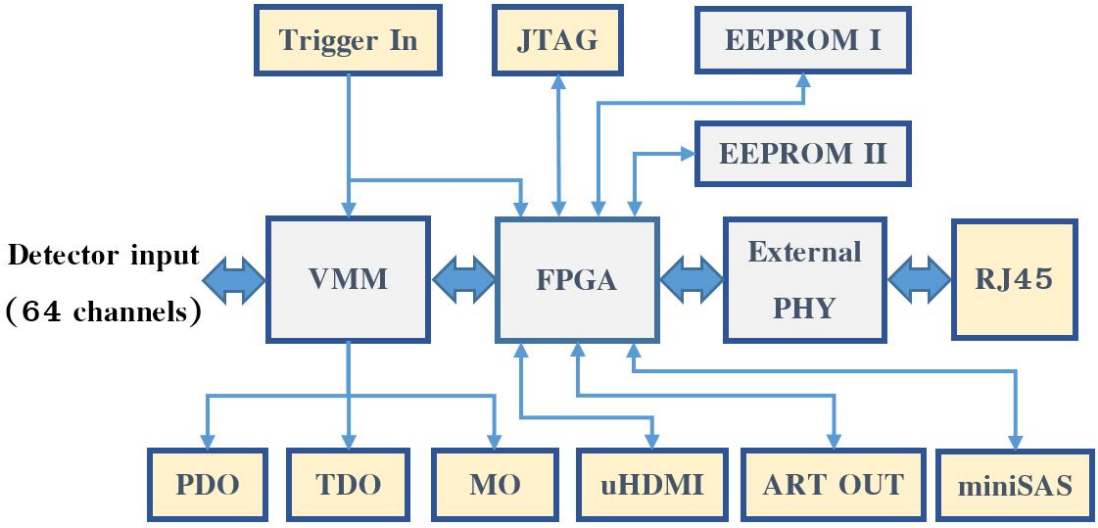


Figure C.2: Block diagram illustrating the architecture of the MMFE1 front-end board. Blocks with yellow color indicate physical connectors while those in gray color indicate on-board components. The VMM is connected to the FPGA via differential lines, while its monitoring outputs drive three LEMO connectors. The trigger signal is propagated to the FPGA and to one of the VMM channels, so that the ASIC can perform precise timing measurements to the trigger signal itself. The FPGA can be accessed via JTAG, while two flash memories store the FPGA's configuration and network access information. Ethernet connection is achieved via a standard RJ45 and External PHY pair that the FPGA can interface with. Finally, the FPGA can interface with off-board devices via a μHDMI and a miniSAS connector.

Appendix D

The VMM Front-End Readout Software

User control of the VRS Front-End FPGA boards is provided by a high-level software tool referred to as 'VERSO': VMM Ethernet Readout SOftware. VERSO is developed entirely in C++ and uses the Qt framework [87] for developing the graphical user interface (GUI) which is shown in Figure D.1. VERSO has two main responsibilities: orchestrating configuration processes and data-acquisition (DAQ). Both functionalities are performed using a custom-made protocol implemented within the UDP standard. VERSO also implements a suite of automated functionalities for performing calibration of the front-end electronics or for data-taking with a fixed number of triggers.

The DAQ functionality provided by VERSO can handle the readout of several front-end boards. The event building is based on a trigger counter that is transmitted with each data fragment upon the reception of a trigger signal at the front-end. An 'event', therefore, is the collection of such data fragments that have the same trigger number. The trigger signal can either be an internal one configured by VERSO and generated by the firmware (e.g. the test-pulse clock, CKTP, described in Section 4.3), or an external signal.

As the UDP protocol cannot ensure a fixed latency, events based on the same trigger number are not guaranteed to be captured by VERSO within the same UDP packet. The event building of VERSO reflects this asynchronous readout by employing a single-producer/single-consumer threading model wherein a reading thread is tasked with the reception of the data-fragments from the front-end(s) and a second processing thread is tasked with processing all those data-fragments associated with each unique trigger number received. The data fragments received by the reading thread are indexed by

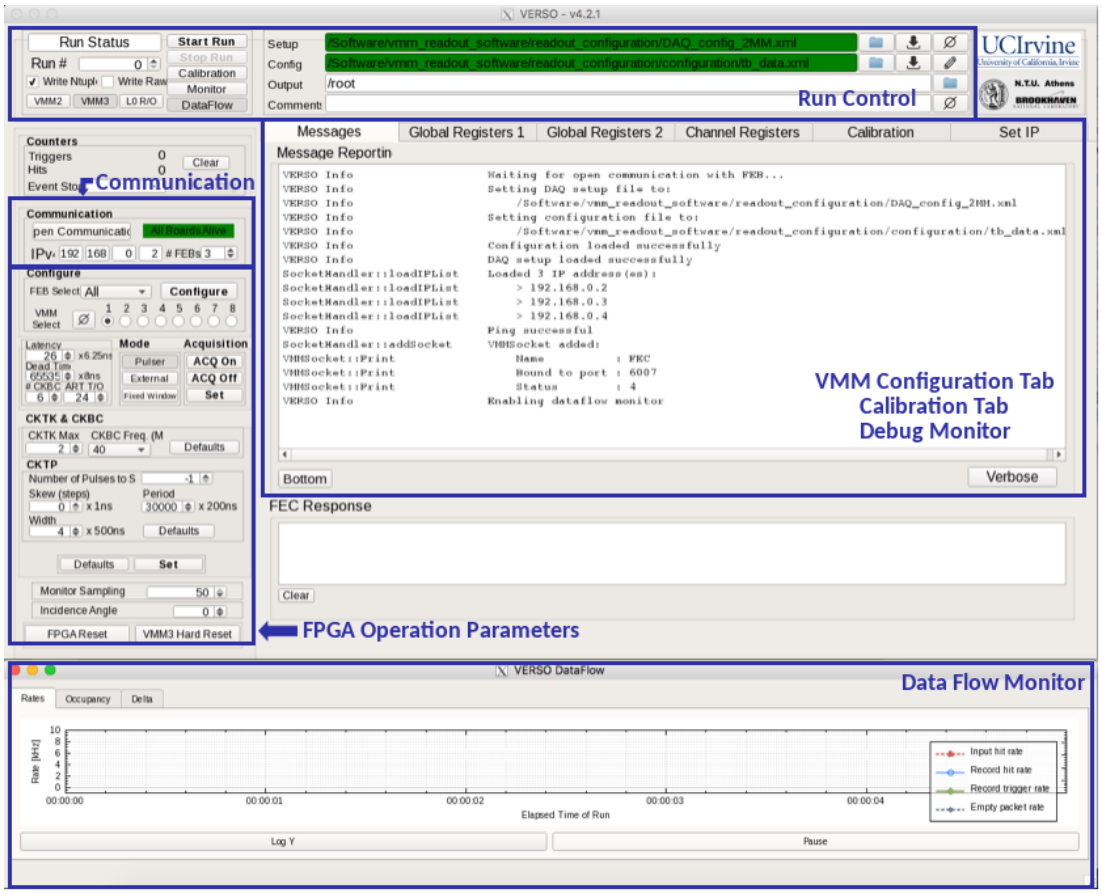


Figure D.1: VERSO user interface.

their associated trigger-number and are buffered in a queue for eventual removal by the processing thread. The queue is implemented as a lock-free concurrent queue [88] which ensures that the two threads never prevent access to the queue for adding or removing data-fragments. This makes for fully-efficient readout even at the maximal data-taking rates observed during test-beam scenarios.

The processing of an event consists of the decoding of the raw data fragments transmitted by the front-end(s) and building high-level data structures representing each event. These structures are continuously stored on disk throughout a given data-taking period in the `.ROOT` data format to be used for offline data analysis.

Appendix E

E-link Wrapper

The E-link Wrapper is an FPGA module that was used extensively in several designs in this dissertation.

E.1 Preface

The E-link Wrapper [89] is a module that can be used as an interfacing agent between two FPGAs, or an FPGA and an ASIC. It supports the HDLC and 8b10b protocols, and bandwidths of 80, 160 and $320\,\mathrm{Mb/s}$. The module deploys two clock-domain-crossing (CDC) FIFOs (one for each direction), that act as an interface between the user logic and the blocks that implement the low-level protocol handling (i.e. HDLC bit unstuffing/stuffing, 8b10b aligning and decoding/encoding, and the actual data serializing and deserializing). As already stated, apart from acting as a bridge between two FPGAs that deploy the said logic, the wrapper can also be used as an interfacing mean between an FPGA and an ASIC that supports the aforementioned data rates and protocols.

The wrapper contains components that originate from the FELIX firmware repository [83]. These pieces of logic are used to connect the rest of the FELIX firmware's logic with front-end devices, that can either be an FPGA which implements the said wrapper (or an equivalent module), or an ASIC. Therefore, the E-link Wrapper is an ideal solution for interfacing an FPGA with FELIX.

One common way to implement this connection, is via a Gigabit Transceiver (GBTx) ASIC [63], that (de)/multiplexes FELIX's fiber (input)/output (into)/from separate serial E-links. This use-case is depicted in Figure E.1. However, this is not the only

way to establish a connection between an FPGA and FELIX. FELIX deploys the so-called *GBT-FPGA* component in order to connect to the GBTx via optical fiber, which makes use of the FELIX FPGA's transceivers. Hence, another way to implement the interfacing between an FPGA and FELIX, is via a direct optical fiber link, provided that the user's FPGA has an equivalent transceiver implementation with FELIX. This scenario is examined later on in this Appendix.

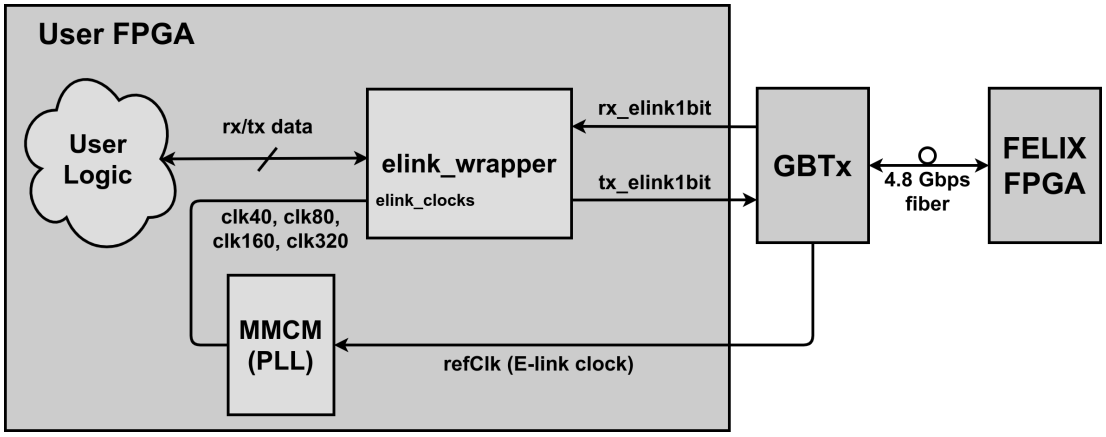


Figure E.1: Example where the E-link Wrapper is used as an interfacing agent between *FELIX* and the FPGA that instantiates the wrapper's logic. Here, FELIX is connected to the FPGA via a *GBTx*. The GBTx is connected to FELIX via optical fiber, and to the user FPGA via two serial E-links. The GBTx is recovering a clock from the optical link, and sends it to the user FPGA via a separate (differential) connection. This reference frequency is used to create the necessary clocking for the *elink_wrapper* logic. Note that the *User Logic* may not be in this domain, since the wrapper deploys two CDC FIFOs that act as a bridge between the wrapper and any surrounding logic.

E.2 E-link Wrapper General Description

The wrapper instantiates the *Elink2FIFO* and *FIFO2Elink* components, that originate from the FELIX firmware repository [83]. These two sub-modules, contain the *Central Router* key logic blocks, that implement the 8b10b aligning and decoding/HDLC unstuffing (RX side - Elink2FIFO), and the 8b10b encoding/HDLC stuffing (TX side - FIFO2Elink). Depending on the chosen bandwidth, these blocks also handle the rate at which the data are received and sent from/to the wrapper. Each direction has *four* available input/output ports. One is single-bit (supporting all data rates), another is two-bit ($80\,\mathrm{Mb/s}$), another is four-bit ($160\,\mathrm{Mb/s}$), and finally one is eight-bit ($320\,\mathrm{Mb/s}$). The single-bit bus is used for serial implementations between two FPGAs, or when the GBTx is involved (see Figure E.1), whereas the multi-bit buses are generally used for

cases where the GBT-FPGA is instantiated within the same FPGA as the wrapper (see Figure E.2).

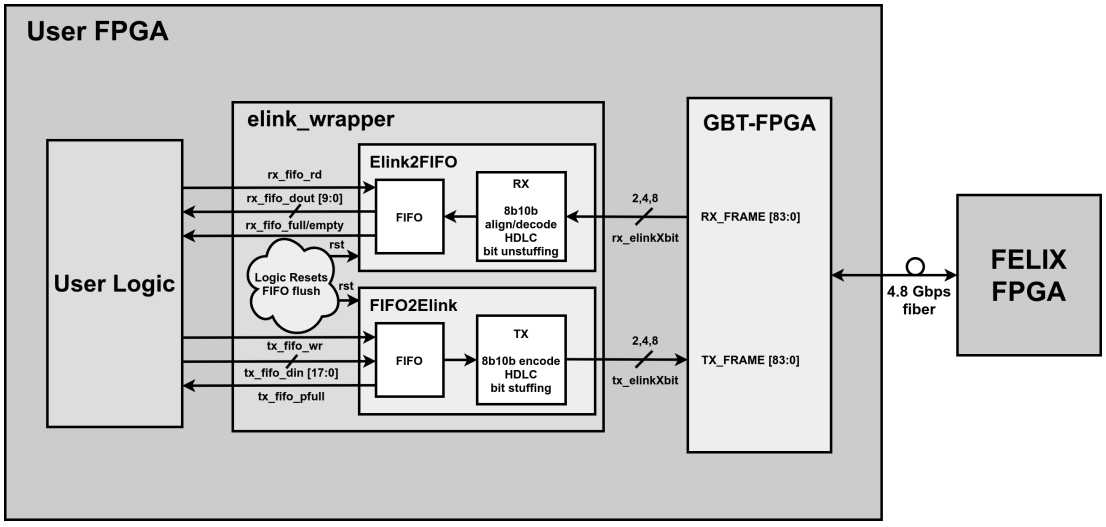


Figure E.2: Example where the E-link Wrapper is used as an interfacing agent between *FELIX* and the FPGA that instantiates the wrapper's logic. Here, FELIX is connected to the FPGA via a *GBT-FPGA* instantiation. Every ~ 25 ns, the GBT-FPGA receives one 84-bit word *(TX_FRAME)*, that serializes to the other endpoint over the fiber. In a similar manner, every ~ 25 ns, the GBT-FPGA presents one newly received 84-bit word *(RX_FRAME)* to the user logic. The *elink_wrapper* is connected to a part of both of these buses. Depending on the chosen data rate, the corresponding number of bits are connected between the two modules (e.g. for $320\,\mathrm{Mb/s}$ eight bits must be parsed between the two components). It is implied that the E-link clocks of the wrapper belong on the same domain as the GBT-FPGA clocking (similarly to Figure E.1, where the GBTx provides the reference clock that generates all of the wrapper's E-link clocks). The *User Logic* reads the RX-FIFO, and writes into the TX-FIFO accordingly.

E.3 Generic and Port Description

The wrapper supports the HDLC and 8b10b protocols. Also, 80, 160 and $320\,\mathrm{Mb/s}$ datarates are available. Finally, the core supports two types of interfacing. There is either a *serial* interfacing scheme, or a *parallel* one. The latter is usually used when a GBT-FPGA instantiation is present within the same FPGA as the wrapper (see Figure E.2), and the former when a direct serial FPGA-to-FPGA connection is to be established, or when the FPGA is communicating with a FELIX endpoint via a GBTx ASIC (see Figure E.1).

The following table provides a full description of the generics section of the E-link Wrapper, that define all aforementioned options:

Generic Name	Possible Values	Description
TXRX_elinkEncoding	2'b01, 2'b10	2'b01 will implement 8b10b communication protocol, and 2'b10 will implement HDLC
TX_dataRate	80, 160, 320	The wrapper's TX Data rate in Mb/s
RX_dataRate	80, 160, 320	The wrapper's RX Data rate in Mb/s
serialized_input	true, false	Setting this to false implements a *parallel* connection of the wrapper, ideal if the logic is implemented on the same FPGA with a GBT-FPGA instance (see Figure E.2). If set to true, the module will accept E-link data in its serial input, intended for applications where a GBT-FPGA is deployed in another FPGA that has a serial connection with the FPGA that hosts the wrapper, or if the wrapper interfaces with FELIX via a GBTx (see Figure E.1, or if a direct FPGA-to-FPGA serial connection has been chosen)

A complete description of the E-link Wrapper's ports section is given in the following table:

Port Name	Direction	Description
clk_usr	in	User clock. The TX-FIFO of FIFO2Elink and the RX-FIFO of the Elink2FIFO submodules use this clock in their write and read domain ports respectively. Therefore, the user logic that handles these ports, must belong on the same domain
clk_40	in	40 MHz E-link clock. Should be related to the GBT-FPGA reference clock, or the GBTx's reference clock. Usually this is the BC clock

`clk_80`	`in`	80 MHz E-link clock. Should be related to the GBT-FPGA reference clock, or the GBTx's reference clock. Usually this is twice the frequency of the BC clock
`clk_160`	`in`	160 MHz E-link clock. Should be related to the GBT-FPGA reference clock, or the GBTx's reference clock. Usually this is four times the frequency of the BC clock
`clk_320`	`in`	320 MHz E-link clock. Should be related to the GBT-FPGA reference clock, or the GBTx's reference clock. Usually this is eight times the frequency of the BC clock
`rst`	`in`	Resets the core. It is strongly recommended to initialize the core upon the system's startup by keeping this signal high and then pulling it low (e.g. by tying it to an inverted `pll_lock` signal)
`reverse_rx`	`in`	Reverses the endianness of the inbound datastream's 8b10b $10-$bit word before being decoded
`reverse_tx`	`in`	Reverses the endianness of the outbound 8b10b $10-$bit word before being sent
`drdy_dbg`	`out`	Clocked at the 40 MHz E-link clock. While high, the word of the `din_dbg` port is being written to the Elink2FIFO buffer. This is a debug input that can be used with a logic analyzer to evaluate the status of the RX link

din_dbg	out	Clocked at the 40 MHz E-link clock. A 10−bit bus indicating which word is written to the Elink2FIFO buffer (the drdy_dbg must be high for the word to be written). This is a debug input that can be used with a logic analyzer to evaluate the status of the RX link. For instance, when using the 8b10b protocol, this port must present the $3bc$ word, indicating the RX link is comma-aligned
rx_elink	in	Serial single-bit input to the wrapper. Should be used if the serialized_input generic is set to true. Should be driven by the serial input from the GBTx, or external GBT-FPGA implementation, or other FPGA's serial TX port
rx_elink2bit	in	Parallel two-bit input to the wrapper. Should be used if the serialized_input generic is set to false and if the RX_datarate is set to 80. Should be driven by two bits of the RX GBT Frame of the GBT-FPGA that connects with FELIX
rx_elink4bit	in	Parallel four-bit input to the wrapper. Should be used if the serialized_input generic is set to false and if the RX_datarate is set to 160. Should be driven by four bits of the RX GBT Frame of the GBT-FPGA that connects with FELIX
rx_elink8bit	in	Parallel eight-bit input to the wrapper. Should be used if the serialized_input generic is set to false and if the RX_datarate is set to 320. Should be driven by eight bits of the RX GBT Frame of the GBT-FPGA that connects with FELIX

`rx_swap`	`in`	Swaps the inbound datastream from the remote endpoint per 2-bits for 80 Mb/s speed, per 4-bits for 160 Mb/s speed, and per 8-bits for 320 Mb/s speed. An equivalent command is also provided on the FELIX side by its software, called *fereverse*
`rx_fifo_rd`	`in`	Toggle high to read one word from the RX-FIFO. Must be in sync with the `clk_usr` domain
`rx_fifo_empty`	`out`	If high, the RX-FIFO yields data that can be presented to the user
`rx_fifo_full`	`out`	If high, the RX-FIFO is full and cannot store any more data coming from the remote endpoint
`rx_fifo_dout`	`out`	After toggling the monrx_fifo_rd port, the RX-FIFO will present the decoded byte plus its two-bit flag in this port
`tx_elink`	`out`	Serial single-bit output from the wrapper. Should be used if the `serialized_input` generic is set to `true`. Should drive the serial output to the GBTx, or external GBT-FPGA implementation, or RX serial input pin of another FPGA
`tx_elink2bit`	`out`	Parallel two-bit output from the wrapper. Should be used if the `serialized_input` generic is set to `false` and if the `TX_datarate` is set to `80`. Should drive two bits of the TX GBT Frame of the GBT-FPGA that connects with FELIX
`tx_elink4bit`	`out`	Parallel four-bit output from the wrapper. Should be used if the `serialized_input` generic is set to `false` and if the `TX_datarate` is set to `160`. Should drive four bits of the TX GBT Frame of the GBT-FPGA that connects with FELIX

tx_elink8bit	out	Parallel eight-bit output from the wrapper. Should be used if the serialized_input generic is set to false and if the TX_datarate is set to 320. Should drive eight bits of the TX GBT Frame of the GBT-FPGA that connects with FELIX
tx_swap	in	Swaps the outbound datastream to-be-sent to the remote endpoint per 2-bits for 80 Mb/s speed, per 4-bits for 160 Mb/s speed, and per 8-bits for 320 Mb/s speed. An equivalent command is also provided on the FELIX side by its software, called *fereverse*
tx_fifo_wr	in	Toggle high to write one word to the TX-FIFO. Must be in sync with the clk_usr domain
tx_fifo_din	in	18−bit data bus containing the two bytes to-be-sent (on the [15:0] part of the bus), plus a two-bit flag
tx_fifo_pfull	out	If high, the TX-FIFO is full and cannot accept more data coming from the user logic

E.4 Transmitting and Receiving Data Using the E-link Wrapper

Transmitting Data

In order to send data using the wrapper, the user logic must handle the tx_fifo_wr and tx_fifo_din ports. The former is the write-enable signal of the TX-FIFO, and the latter is its 18-bit data input bus. In order to transmit data, the sequence below must be followed:

- 1: Write [17:16]=”10” & [15:0]=0x0000. The first two bits represent the Start Of Packet (SOP). The word in [15:0] does not get transmitted by the wrapper.

- `2`: Write [17:16]=”00” & [15:0]=0xXXXX. The first two bits represent the Middle Of Packet (MOP). These data (i.e. 0xXXXX) get transmitted by the wrapper.
- ...
- `N-1`: Write [17:16]=”00” & [15:0]=0xYYYY. The first two bits represent the Middle Of Packet (MOP). These data (i.e. 0xYYYY) get transmitted by the wrapper.
- `N`: Write [17:16]=”01” & [15:0]=0x0000. The first two bits represent the End Of Packet (EOP). The word in [15:0] does not get transmitted by the wrapper.

The sequence above will send N bytes of the [15:0] part of the `tx_fifo_din` bus that were written in-between step 1 and N. Upon the writing of the EOP word, transmission begins. The user may write any number of MOP 16-bit words in-between the SOP and EOP flags, as long as the `tx_fifo_pfull` is low. In Figure E.3, the user logic sends a packet that is sixteen-byte long.

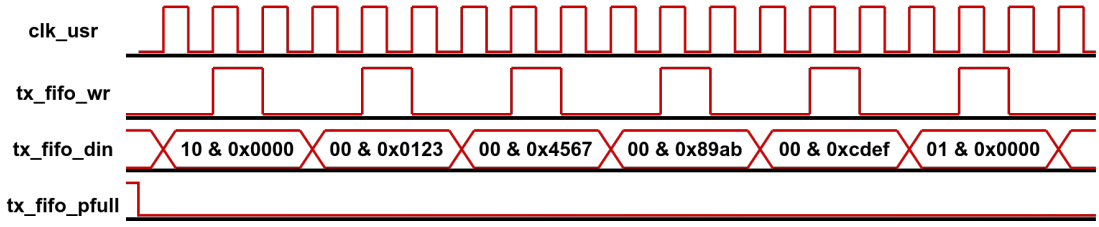


Figure E.3: Timing diagram of the signals driven by the User Logic to the E-link Wrapper’s TX-FIFO (i.e. FIFO2Elink) ports. Here, bytes 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef will be sent.

Receiving Data

In order to read data as received from the wrapper, the user logic must handle the `rx_fifo_rd`, `rx_fifo_dout` and `rx_fifo_empty` ports. By probing the latter signal, the user logic is aware of when a data word has been successfully decoded and buffered by the RX-FIFO (i.e. Elink2FIFO), and ready to be read. By toggling the `rx_fifo_rd` high for one clock cycle, the `rx_fifo_dout` will be updated with the new data word after one clock cycle.

The `rx_fifo_dout` bus is 10-bits wide, the two most significant bits represent the *flag*, i.e. the EOP, SOP or MOP. A value of ”10” indicates this is the SOP, and in the case of the 8b10b encoding, the accompanying byte will be the decoded standard K28.1 character (0x3c). Usually, after the SOP, the main body of the message follows, that is

flagged with ”00” at the first two bits of each decoded byte. The EOP is indicated by a ”01” in the first two bits, and in the case of the 8b10b encoding, the accompanying byte will be the decoded standard K28.6 character (0xdc). See Figure E.4 for the wrapper’s repository testbench result which demonstrates this sequence of events.

E.5 The Repository

The E-link Wrapper may be cloned from the associated repository [89]. The repository contains all the necessary files that the user should deploy in their project to operate the core. The `elink_wrapper` module is instantiated in a top-level design file, called `elink_wrapper_top`, that connects the wrapper’s TX and RX ports in loopback. The TX-FIFO inputs are controlled by the top-level’s ports, while the RX-FIFO is being read by a simple process instantiated in the top-level. The read words are forwarded to the top-level’s ports. A behavioral testbench VHDL source file, named `elink_wrapper_top_tb`, writes a sixteen-byte message to the TX ports, in a similar fashion with Figure E.3. The data are being transmitted back into the wrapper’s RX logic, and the Elink2FIFO presents the data to the user logic via its buffer, which is being read. The result is shown in Figure E.4.

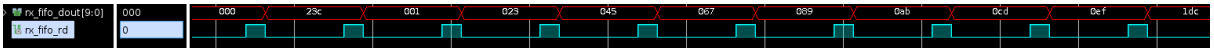


Figure E.4: Behavioral simulation result of the wrapper when instantiated in loopback, as taken from its repository. The data that are written to the TX ports follow the timing diagram of FIgure E.3. The read data are the same as the ones transmitted.

References

[1] Grupen, C. and Schwartz, B., *Paricle Detectors, 2nd Revised Edition*. Cambridge University Press, 2008.

[2] Herr, W. and Muratori, B., "Concept of Luminosity," tech. rep., CERN, Geneva. Available online at: `http://cds.cern.ch/record/941318`.

[3] G. Iakovidis, *Research and Development in Micromegas Detector for the ATLAS Upgrade*. PhD thesis, Natl. Tech. U., Athens, Oct 2014. Available online at: `https://cds.cern.ch/record/1955475`.

[4] K. Ntekas, *Performance characterization of the Micromegas detector for the New Small Wheel upgrade and Development and improvement of the Muon Spectrometer Detector Control System in the ATLAS experiment*. PhD thesis, Natl. Tech. U., Athens, 2016. Available online at: `https://cds.cern.ch/record/2143887`.

[5] "ATLAS Detector and Physics Performance : Technical Design Report, 1," Tech. Rep. CERN-LHCC-99-014. ATLAS-TDR-14, CERN, Geneva. Available online at: `https://cds.cern.ch/record/391176`.

[6] Griffiths, D., *Introduction to Elementary Particles, 2nd Revised Edition*. Wiley-VCH, 2008.

[7] CERN, *New Small Wheel Technical Design Report*. No. CERN-LHCC-2013-006. ATLAS-TDR-020, Jun 2013.

[8] P. Gkountoumis, *Design and development of the Level-1 Data Driver Card (L1DDC) for the New Small Wheel upgrade of the ATLAS experiment at CERN*. PhD thesis, Natl. Tech. U., Athens, April 2019. Available online at: `https://cds.cern.ch/record/2674048`.

[9] G. De Geronimo, J. Fried, L. Shaorui, *et al.*, "VMM1 - An ASIC for Micropattern Detectors," *IEEE Transactions on Nuclear Science*, vol. 60, pp. 2314–2321, June 2013.

[10] Bakalis, C., "VMM3a: an ASIC for Tracking Detectors," *Proceedings of Science*, 2019.

[11] G. Iakovidis, "VMM3, an ASIC for Micropattern Detectors," Available online at: `https://cds.cern.ch/record/2309951`.

[12] F. J. Kurose and W. K. Ross, *Computer Networking, A Top-Down Approach, 6th Edition*. Pearson Education, 2013.

[13] Xilinx, *Xilinx User and Product Guides*. Xilinx Corporation.

[14] "LHC Design Report," Tech. Rep. CERN-2004-003-V-1, CERN, Geneva. Available online at: `https://cds.cern.ch/record/782076`.

[15] "CMS Physics: Technical Design Report Volume 1: Detector Performance and Software," Tech. Rep. CERN-LHCC-2006-001. CMS-TDR-8-1, CERN, Geneva. Available online at: `https://cds.cern.ch/record/922757`.

[16] G. Aad *et al.*, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," *Phys. Lett.*, vol. B716, pp. 1–29, 2012.

[17] S. Chatrchyan *et al.*, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," *Phys. Lett.*, vol. B716, pp. 30–61, 2012.

[18] CERN, *ATLAS Central Solenoid: Technical Design Report*. Geneva: CERN, 1997. Available online at: `https://cds.cern.ch/record/331067?ln=enATLAS-TDR-9` ; CERN-LHCC-97-021.

[19] CERN, *ATLAS End-Cap Toroids: Technical Design Report*. Geneva: CERN, 1997. Available online at: `https://cds.cern.ch/record/331066?ln=enATLAS-TDR-8` ; CERN-LHCC-97-020.

[20] J. P. Badiou, J. Beltramelli, J. M. Baze, and J. Belorgey, *ATLAS Barrel Toroid: Technical Design Report*. Geneva: CERN, 1997. Available online at: `https://cds.cern.ch/record/331065?ln=enATLAS-TDR-7` ; CERN-LHCC-97-019.

[21] CERN, *ATLAS inner detector: Technical Design Report, 1*. Geneva: CERN, 1997. Available online at: `https://cds.cern.ch/record/331064?ln=enATLAS-TDR-5` ; CERN-LHCC-97-017.

[22] CERN, *ATLAS Calorimeter Performance: Technical Design Report*. Geneva: CERN, 1996. Available online at: `https://cds.cern.ch/record/331059?ln=en`ATLAS-TDR-1 ; CERN-LHCC-96-040.

[23] CERN, *ATLAS Muon Spectrometer: Technical Design Report*. Geneva: CERN, 1997. Available online at: `https://cds.cern.ch/record/331068?ln=en`ATLAS-TDR-10 ; CERN-LHCC-97-022.

[24] Y. Kataoka, S. Leontsinis, and K. Ntekas, "Performance studies of a micromegas chamber in the atlas environment," *Journal of Instrumentation*, vol. 9, no. 03, pp. C03016, 2014.

[25] T. Alexopoulos, J. Burnens, R. de Oliveira, *et al.*, "A Spark-Resistant Bulk-Micromegas Chamber for High-Rate Applications," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators Spectrometers, Detectors and Associated Equipment*, vol. A640, pp. 110–118, 2011.

[26] G. De Geronimo and L. Shaorui, "Shaper Design in CMOS for High Dynamic Range," *IEEE Transactions on Nuclear Science*, vol. 58, pp. 2382–2390, Oct 2011.

[27] G. De Geronimo, J. Fried, G. Smith, *et al.*, "ASIC for Small Angle Neutron Scattering Experiments at the SNS," *IEEE Transactions on Nuclear Science*, vol. 54, pp. 541–548, June 2007.

[28] Y. Giomataris, P. Rebourgeard, J. Robert, and G. Charpak, "MICROMEGAS: a High-Granularity Position-Sensitive Gaseous Detector for High Particle-Flux Environments," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 376, pp. 29 – 35, 1996.

[29] Y. Giomataris, "Development and Prospects of the New Gaseous Detector Micromegas," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 419, no. 23, pp. 239 – 250, 1998.

[30] T. Alexopoulos, J. Burnens, R. de Oliveira, *et al.*, "A Spark-Resistant Bulk-Micromegas Chamber for High-Rate Applications," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 640, no. 1, pp. 110 – 118, 2011.

[31] Wotschack, Joerg, "The Development of Large-Area Micromegas Detectors for the ATLAS Upgrade," *Modern Physics Letters A*, vol. 28, no. 13, p. 1340020, 2013.

[32] J. Bortfeldt, "Construction and Test of Full-Size Micromegas Modules for the ATLAS New Small Wheel Upgrade," Tech. Rep. ATL-MUON-PROC-2015-017, CERN, Geneva, Nov 2015. Available online at: `https://cds.cern.ch/record/2104572`.

[33] A. X. Widmer and P. Franaszek, "A DC-Balanced, Partitioned Block, 8B/10B Transmission Code," *IBM Journal of research and development*, vol. 27, no. 5, 1983.

[34] P. Horowitz and W. Hill, *The Art of Electronics, 3rd Edition*. Cambridge University Press, 2015.

[35] S. Hauck and A. DeHon, *Reconfigurable Computing. The Theory and Practice of FPGA-Based Computation*. Elsevier, 2008.

[36] I. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," *Foundations and Trends in Electronics Design Automation*, vol. 2, no. 2, pp. 135–253, 2007.

[37] H. Pavrez and H. Mehrez, *Application-Specific, Mesh-Based, Heterogenous FPGA Architectures*. Springer, 2011.

[38] M. M. Mano and D. M. Ciletti, *Digital Design: With an Introduction to the Verilog HDL, 5th Edition*. Pearson, 2012.

[39] G. Rizzoni and J. Kearns, *Principles and Applications of Electrical Engineering, 6th Edition*. McGraw-Hill Education, 2015.

[40] Athavaleand, A. and Christensen, C., *High-Speed Serial I/O Made Simple - A Designers' Guide, with FPGA Applications*. Xilinx Corporation. Available online at: `https://www.xilinx.com/publications/archives/books/serialio.pdf`.

[41] V. Betz, J. Rose, and A. Marquadt, *Architecture and CAD for Deep-Submicron FPGAs*. Springer, 1999.

[42] R.-M. Coliban, S. Popa, T. Tulbure, *et al.*, "The read out controller for the ATLAS new small wheel," *Journal of Instrumentation*, vol. 11, pp. C02069–C02069, feb 2016.

[43]

[44] T. Alexopoulos, D. Antrim, C. Bakalis, *et al.*, "The VMM Readout System," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators Spectrometers, Detectors and Associated Equipment*, vol. 955, 2020.

[45] "VMM Readout System Front-End FPGA Firmware Repository." URL: https://gitlab.cern.ch/NSWelectronics/vmm_boards_firmware.

[46] "Opencores website."

[47] Xilinx, *Xilinx 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide for HDL Designs*. Xilinx Corporation, 2013.

[48] Maxim Integrated, "NRZ Bandwidth - LF Cutoff and Baseline Wander," tech. rep., Maxim Integrated. Available online at: https://pdfserv.maximintegrated.com/en/an/AN1738.pdf.

[49] IEEE Computer Society, "IEEE Standard for Ethernet, IEEE Std 802.3-2018," tech. rep., Institute of Electrical and Electronics Engineers, Inc., 2018. Available online at: https://ieeexplore.ieee.org/document/8457469f.

[50] De Geronimo, P. and Iakovidis, G. and Polychronakos, V., "ATLAS NSW Electronics Specifications - VMM3a," tech. rep., New Small Wheel - Brookhaven National Laboratory, 2018. Available online at: https://espace.cern.ch/ATLAS-NSW-ELX/Shared%20Documents/VMM/vmm3a.pdf.

[51] "Two-Phase Analog Readout Software Repository." URL: https://gitlab.cern.ch/cbakalis/twophaseanalog_readout_software.

[52] T. Alexopoulos, C. Bakalis, G. De Geronimo, and P. V., "Characterization of the VMM front-end ASIC for High-Resolution Applications," *2020 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, IEEE/NSS Boston 2020 Conference Proceeding.

[53] Analog Devices, *2V, 6A Synchronous Step-Down Regulator with 3uA Quiescent Current*. Linear Technology.

[54] "VMM Readout System Supervisory Board FPGA Firmware Repository." URL: https://gitlab.cern.ch/cbakalis/vsb_daq.

[55] A. Koulouris, *Performance characterization of the NSW Micromegas detector, services design and development of the Slow Control Adapter for the ATLAS upgrade*. PhD thesis, Natl. Tech. U., Athens, May 2019. Available online at: https://cds.cern.ch/record/2675805.

[56] Iakovidis, G., "VMM3 Testbeam Analysis Results." Private Communication, October 2017.

[57] S. Ask, D. Berge, P. Borrego-Amaral, *et al.*, "The ATLAS Central Level-1 Trigger Logic and TTC System," *Journal of Instrumentation*, vol. 3, pp. P08002–P08002, aug 2008.

[58] "Micromegas Analysis Repository." URL: `https://gitlab.cern.ch/aikoulou/koulVMM3/tree/cbakalis`.

[59] CERN, *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*. No. CERN-LHCC-2017-020. ATLAS-TDR-029, Sep 2017.

[60] A. T. Collaboration, "FELIX: developing a new detector interface for ATLAS trigger and readout," Tech. Rep. ATL-DAQ-PROC-2018-026, CERN, Geneva, Oct 2018. Available online at: `http://cds.cern.ch/record/2644047`.

[61] Trovato, M., "FELIX: The New Readout System for the ATLAS Detector," 2019. Available online at: `https://cds.cern.ch/record/2704279`.

[62] Anderson, J. and others, "FELIX: The New Approach for Interfacing to Front-End Electronics for the ATLAS Experiment," *Journal of Instrumentation*, vol. 11, 2016.

[63] P. Moreira, A. Marchioro, and K. Kloukinas, "The GBT: A Proposed Architecure for multi-Gb/s Data Transmission in High Energy Physics," 2007. Available online at: `http://cds.cern.ch/record/1091474`.

[64] P. Moreira, R. Ballabriga, S. Baron, *et al.*, "The GBT Project," 2009.

[65] Baron, S. and Cachemiche, J. P. and Marin, F. and others, "Implementing the GBT Data Transmission Protocol in FPGAs," *CERN Yellow Reports: Conference Proceedings*, 2009.

[66] Moschovakos, P. and Nikiel, P. and Schlenker, S. and Boterenbrood, H. and Koulouris, A., "A Software Suite for the Radiation Tolerant Giga-bit Transceiver - Slow Control Adapter," *17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems*, 2019.

[67] "FELIX User Manual." Available online at: `https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/felix-user-manual/versions/Latest/`.

[68] White, D., "Considerations Surrounding Single Event Effects in FPGAs, ASICs, and Processors," tech. rep., Xilinx Inc., 2012. Available online at: `https://www.xilinx.com/support/documentation/white_papers/wp402_SEE_Considerations.pdf`.

[69] Hussein, J. and Swift, G., "Mitigating Single-Event Upsets," tech. rep., Xilinx Inc., 2015. Available online at: https://www.xilinx.com/support/documentation/white_papers/wp395-Mitigating-SEUs.pdf.

[70] Wang, J. and Guan, L. and Chapman, J. W. and Zhou, B. and Zhu, J., "Design of a Trigger Data Serializer ASIC for the Upgrade of the ATLAS Forward Muon Spectrometer," *IEEE Transactions on Nuclear Science*, vol. 64, no. 12, 2017.

[71] G. Mazza, A. Rivetti, P. Moreira, *et al.*, "A Radiation Tolerant 5 Gb/s Laser Driver in 130 nm CMOS Technology," *Journal of Instrumentation*, vol. 3, pp. P08002–P08002, January 2012.

[72] Menuni, M. and Gui, P. and Moreira, P., "The GBTIA, a 5 Gbit/s Radiation-Hard Optical Receiver for the SLHC Upgrades," Jan 2010.

[73] "GBTx User Manual." Available online at: https://espace.cern.ch/GBT-Project/GBTX/Manuals/gbtxManual.pdf.

[74] K. Poltorak, F. Tavernier, and P. Moreira, "A radiation-hard PLL for frequency multiplication with programmable input clock and phase-selectable output signals in 130 nm CMOS," *Journal of Instrumentation*, vol. 7, pp. C12014–C12014, dec 2012.

[75] Schumacher, J. and Plessl, C. and Vandelli, W., "High-throughput and low-latency network communication with NetIO," Tech. Rep. ATL-DAQ-PROC-2017-010. 8, CERN, Geneva, Apr 2017. Available online at: http://cds.cern.ch/record/2260396.

[76] "IC-Over-NetIO Software Repository." URL: https://gitlab.cern.ch/cbakalis/ic-over-netio.

[77] "FELIX Low-Level Tools Software Repository." URL: https://gitlab.cern.ch/atlas-tdaq-felix/ftools.

[78] "NetIO Traffic Analyzer Software Repository." URL: https://gitlab.cern.ch/cbakalis/netio_traffic_analyzer.

[79] Bakalis, C., "Front-End and Back-End Electronics for the ATLAS New Small Wheel Upgrade," *Proceedings of Science*, 2018.

[80] C. Bakalis, T. Alexopoulos, I. Grayzman, L. Lee, L. Levinson, and V. Polychronakos, "Accessing register spaces in FPGAs within the ATLAS DAQ scheme via the SCA eXtension," *Journal of Instrumentation*, vol. 16, p. P06041, Jun 2021.

[81] T. Alexopoulos, C. Bakalis, L. Levinson, *et al.*, "SCA eXtension: a Design for FPGA Parameter Configuration within the ATLAS DAQ Scheme," *2019 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, 2020.

[82] "SCA eXtension GitLab Repository." URL: `https://gitlab.cern.ch/cbakalis/sca-extension`.

[83] "FELIX Firmware Repository." URL: `https://gitlab.cern.ch/atlas-tdaq-felix/firmware`.

[84] "GBT-FPGA - GBT Group GitLab Repository." URL: `https://gitlab.cern.ch/gbt-fpga/gbt-fpga`.

[85] "GBT-FPGA VC709 GitLab Repository." URL: `https://gitlab.cern.ch/cbakalis/gbt-fpga_vc709`.

[86] I. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, pp. 300–304, Jun 1960.

[87] "Qt." URL: `https://www.qt.io/`.

[88] "A Single-Producer, Single-Consumer Lock-Free Queue for C++." URL: `https://github.com/cameron314/readerwriterqueue`.

[89] "E-link Wrapper Firmware Repository." URL: `https://gitlab.cern.ch/cbakalis/elink_wrapper`.