ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

Αναβαθμίσεις του Συστήματος Σκανδαλισμού Μιονίων στην Κυλινδρική Περιοχή του Πειράματος CMS στο CERN για τον LHC και τον HL-LHC

Σταύρος Μάλλιος

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

ΙΩΑΝΝΙΝΑ 2019

UNIVERSITY OF IOANNINA
SCHOOL OF SCIENCES
DEPARTMENT OF PHYSICS

The CMS Barrel Muon Track Finder and Upgrades for HL-LHC

Stavros Mallios

DOCTORAL THESIS

IOANNINA 2019

Three-member Advisory Committee

1. Ioannis Papadopoulos, Assistant Professor, Department of Physics, University of Ioannina, Greece (Thesis Supervisor).

2. Konstantinos Fountas, Professor, Department of Physics, University of Ioannina, Greece.

3. Ioannis Evangelou, Associate Professor, Department of Physics, University of Ioannina, Greece.

Seven-member Assessment Committee

1. Ioannis Papadopoulos, Assistant Professor, Department of Physics, University of Ioannina, Greece (Thesis Supervisor).

2. Konstantinos Fountas, Professor, Department of Physics, University of Ioannina, Greece.

3. Ioannis Evangelou, Associate Professor, Department of Physics, University of Ioannina, Greece.

4. Nikolaos Manthos, Associate Professor, Department of Physics, University of Ioannina, Greece.

5. Vasilios Christofilakis, Assistant Professor, Department of Physics, University of Ioannina, Greece.

6. Dimitrios Loukas, Director of Research, Democritus Institute of Nuclear and Particle Physics (INPP).

7. Michalis Bachtis, Assistant Professor, Department of Physics & Astronomy, UCLA, USA.

Acknowledgments

Firstly, I would like to thank all the members of High-Energy Physics Laboratory (HEPLAB) and especially my supervisor Assistant Professor I. Papadopoulos. His guidance and support helped me during the time of research and writing of this thesis. I would also like to express my sincere gratitude to my advisor and mentor Professor K. Fountas for his patience, motivation, and immense knowledge. I could not have imagined having better mentor for my Ph.D study.

Special thanks to M. Bachtis for our amazing collaboration without which, this thesis would not be possible, and K. Velidis for his immense support and excellent life and career advises.

I thank my fellow lab-mates and friends G. Flouris, V. Paradas, P. Katsoulis and G. Karathanasis for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years.

I gratefully thank Greg Iles for enlightening and advising me, and all the UK-CMS trigger group for their support in the use of the MP7 and Serenity cards. I am grateful to my colleagues and friends S. A. Thayil, Antonis Agapitos and Sarah Freed, for all the psychological support and for their valuable contribution in the process of writing and editing the present dissertation.

Last but not least, I would like to thank my family and especially my mother Athina, for supporting me throughout writing this thesis and my life in general.

Dedicated to my mother

Abstract

The *Compact Muon Solenoid (CMS)* is a general purpose experiment measuring proton-proton and heavy-ion collisions at the *Large Hadron Collider (LHC)* at CERN. The LHC is a large particle accelerator at the CERN research laboratory, designed to provide particle physics experiments with collisions at unprecedented centre-of-mass energies. To extend the sensitivity for new physics searches, a major upgrade of the LHC has been decided and is being prepared, the *High Luminosity LHC (HL-LHC)*. In its "ultimate" configuration, the HL-LHC would reach a peak instantaneous luminosity of $7 - 7.5 \times 10^{34}$ cm^{-2}s^{-1} increasing the "pile-up" or the average number of proton-proton collisions per bunch crossing to around 200. In order to maintain the excellent performance, the CMS detector has planned a complete replacement of the trigger electronics and a full redesign of the system's architecture.

In this dissertation, the contributions to the Phase-2 CMS detector upgrade are presented, comprised of the design and testing of a new high-speed link protocol for the Phase-2 trigger, the integration of a new muon tracking algorithm for the Barrel Muon Trigger in the Barrel Muon Trigger boards and the contribution in the design and testing of a demonstrator board for the Layer-1 Barrel Muon Trigger. The new links takes advantage of all the latest advances in technology to increase the bandwidth of the trigger channels to 16 Gbps and 25 Gbps. The new muon tracking algorithm is featuring a Kalman filter and extends the trigger capabilities in finding displaced muons originating from very long lived particles. The efficiency of tracking such muons, originating 40 cm to 100 cm away from the interaction vertex, is increased by a factor of 2.5. Finally, the new demonstrator board, features a 20 nm technology Field Programmable Gate Array (FPGA) and it is capable of running sixteen optical links at 16 Gbps, with excellent Bit Error Ratio (BER) of less than 10^{-15}.

Contents

Εκτεταμένη σύνοψη στα Ελληνικά

Εισαγωγή

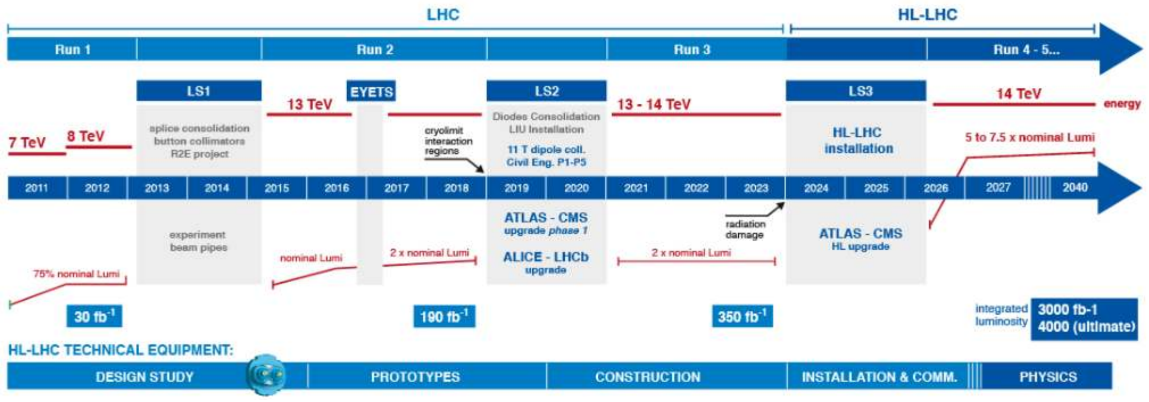
Ο Μεγάλος Επιταχυντής Συγκρουόμενων Δεσμών Πρωτονίων (LHC), βρίσκεται στο ευρωπαϊκό κέντρο πυρηνικών ερευνών CERN, και αποτελεί τον μεγαλύτερο και ισχυρότερο επιταχυντή σωματιδίων στον κόσμο. Αποτελείται από μία διάταξη υπεραγώγιμων μαγνητών μήκους 27 χιλιομέτρων. Ο LHC επιταχύνει δέσμες πρωτονίων ή ιόντων που κινούνται μέσα στον δακτύλιο σε αντίθετες κατευθύνσεις, με ταχύτητες που πλησιάζουν την ταχύτητα του φωτός (99.9999991% c). Οι δέσμες συγκρούονται σε τέσσερα σημεία του δακτυλίου. Γύρω από κάθε σημείο σύγκρουσης, έχουν εγκατασταθεί διατάξεις ανιχνευτών, που αποτελούν τα τέσσερα κύρια πειράματα: CMS, ATLAS, LHCb και ALICE.

Το πείραμα CMS, στο οποίο εκπονήθηκε η παρούσα διατριβή, προκύπτει ως αρκτικόλεξο του “Compact Muon Solenoid”, που μεταφράζεται ως Συμπαγές Μιονικό Σωληνοειδές, και αποτελεί έναν από τους δύο μεγάλους ανιχνευτές γενικού σκοπού που λειτουργούν στον LHC.

Οι κυριότεροι στόχοι του CMS είναι η επιβεβαίωση θεωρητικών προβλέψεων του Καθιερωμένου Προτύπου, όπως η ανακάλυψη του μποζονίου Higgs (στόχος που επιτεύχθηκε το 2012), η εξερεύνηση της φυσικής στην κλίμακα των TeV, καθώς και η αναζήτηση φυσικής πέρα από το Καθιερωμένο Πρότυπο, όπως π.χ. η υπερσυμμετρία. Επίσης, το CMS μελετά το πλάσμα κουάρκ και γλουονίων μέσω συγκρούσεων βαρέων ιόντων (Pb σε Pb). Μεγαλύτερη έμφαση έχει δοθεί στη μελέτη των ιδιοτήτων του μποζονίου Higgs, λόγω του βασικού του ρόλου στην κατανόηση της φύσης του σπασίματος της ηλεκτρασθενούς συμμετρίας.

Κατά τη διάρκεια της πρώτης Φάσης (Phase-1) του προγράμματος του LHC (2011-2023), η φωτεινότητα αλληλεπιδράσεων έφτασε τα $2 \times 10^{34}\ cm^{-2}s^{-1}$. Για κάθε αλληλεπίδραση, οι επιμέρους ανιχνευτές του πειράματος CMS παράγουν τεράστιο όγκο πληροφορίας, που υπερβαίνει τα 27 Petabits ανά δευτερόλεπτο, όγκος που είναι αδύνατον να αποθηκευτεί. Μεγάλο μέρος, όμως, της πληροφορίας αυτής, είτε είναι θόρυβος ή δεν προσφέρει κάτι νέο ή χρήσιμο στην Φυσική. Για αυτό τον λόγο, έχει κατασκευαστεί ένα εξελιγμένο ηλεκτρονικό σύστημα επιλογής (σύστημα σκανδαλισμού), υπεύθυνο για τη μείωση του παραγόμενου όγκου δεδομένων κατά 40000 φορές. Η μείωση πραγματοποιείται σε δύο στάδια. Το 1ο επίπεδο σκανδαλισμού, ή Level-1 Trigger (L1T), και το δεύτερο επίπεδο σκανδαλισμού ή High Level Trigger (HLT). Το L1T, αποτελείται από ειδικά σχεδιασμένες ηλεκτρονικές διατάξεις, οι οποίες πρέπει να αποφασίσουν, σε χρόνο μικρότερο των 4 μs, αν θα απορρίψουν την πληροφορία που συλλέχτηκε ή αν θα την αποδώσουν στο δεύτερο επίπεδο σκανδαλισμού για περαιτέρω επεξεργασία. Λαμβάνει δεδομένα από τα καλορίμετρα και από το σύστημα μιονίων, με συχνότητα 40 MHz (εκατομμυρίων γεγονότων ανά δευτερόλεπτο), και επιλέγει 100000 από αυτά (μείωση του ρυθμού στα 100 kHz). Το HLT από την άλλη πλευρά, διαβάζει και επεξεργάζεται την πληροφορία που προέρχεται από το L1T, στην συστοιχία των υπολογιστών του, και επιλέγει τα γεγονότα που ικανοποιούν συγκεκριμένους κανόνες σκανδαλισμού που κάθε φορά τίθενται, ανάλογα με το εκάστοτε μελετώμενο κανάλι φυσικής. Το HLT επιλέγει τις 1000 πιο ενδιαφέρουσες αλληλεπιδράσεις ανά δευτερόλεπτο, οδηγώντας σε τελική μείωση του ρυθμού στο 1 kHz.

Το 2023, ο επιταχυντής LHC πρόκειται να αναβαθμιστεί, και θα εισέλθει στην δεύτερη Φάση του (Phase-2). Ο αναβαθμισμένος LHC, ή High Luminosity LHC (HL-LHC) όπως ονομάζεται, έχει στόχο την διεύρυνση των δυνατοτήτων του επιταχυντή με την φωτεινότητα (luminosity) να φτάνει τα $7.5 \times 10^{34}\ cm^{-2}s^{-1}$, 7.5 φορές μεγαλύτερη από τον αρχικό σχεδιασμό του επιταχυντή (εικόνα 1). Η αύξηση της φωτεινότητας θα οδηγήσει σε αύξηση τον αριθμό των ταυτοχρόνων συγκρούσεων (pile-up), κατά την σύγκρουση δύο πακέτων πρωτονίων (bunches), από ~ 45 (Φάση I) σε ~ 200. Το νέο περιβάλλον συγκρούσεων θα απαιτεί μεγαλύτερη αποδοτικότητα των ηλεκτρονικών συστημάτων (efficiency), καλύτερη αξιοπιστία του υλικού και μείωση του ρυθμού σκανδαλισμού (trigger rate).

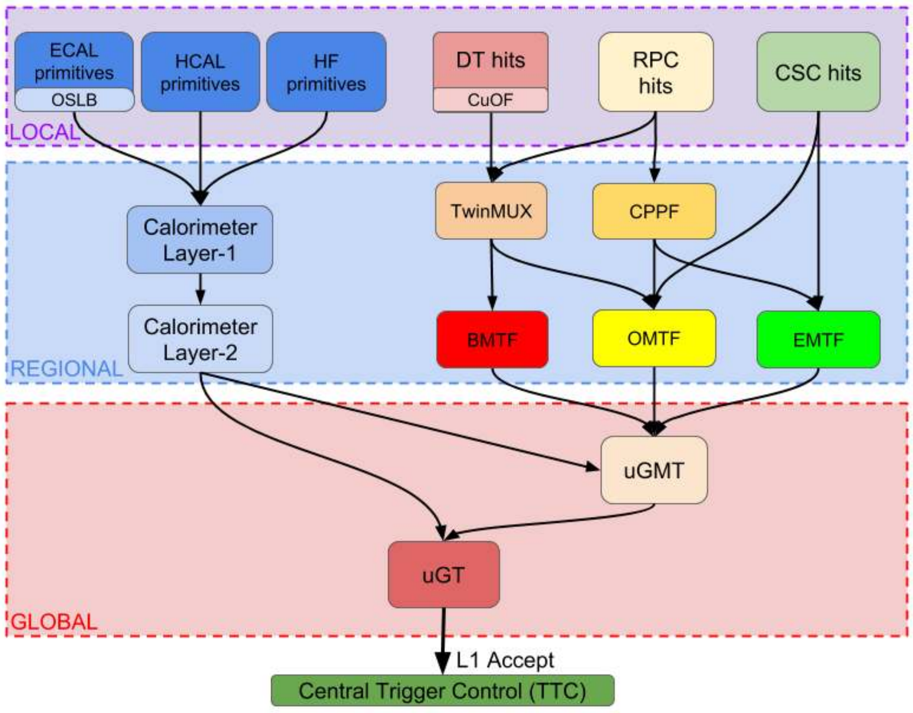


Εικόνα 1: Το χρονοδιάγραμμα της αναβάθμισης του LHC.

Το πείραμα CMS θα προχωρήσει στην αναβάθμιση τόσο των επιμέρους ανιχνευτών, όσο και του συστήματος σκανδαλισμού, ώστε να αντεπεξέλθει στις νέες υψηλότερων απαιτήσεων συνθήκες. Ο χρόνος απόφασης του L1T θα αυξηθεί από 4 σε 12.5 μs, και ο ρυθμός επιλογής από 100 kHz σε 750 kHz. Όλα τα ηλεκτρονικά του συστήματος σκανδαλισμού θα αντικατασταθούν με νέα, που θα αντικατοπτρίζουν και την αντίστοιχη πρόοδο της τεχνολογίας. Οι σημαντικότερες αναβαθμίσεις περιλαμβάνουν την αντικατάσταση των καλοριμέτρων από νέα, υψηλής διακριτικής ικανότητας, καθώς και την πρόσβαση του συστήματος σκανδαλισμού των μιονίων στα δεδομένα του ανιχνευτή τροχιών (tracker).

Το σύστημα σκανδαλισμού μιονίων

Το πρώτο επίπεδο σκανδαλισμού του CMS χωρίζεται σε δύο ομάδες υποσυστημάτων σκανδαλισμού: τον σκανδαλισμό του καλοριμέτρου (calorimeter trigger) και τον σκανδαλισμό των μιονίων (muon trigger). Το σύστημα σκανδαλισμού μιονίων απαρτίζεται χωρικά από ανιχνευτές τροχιών μιονίων, οι οποίοι λαμβάνουν δεδομένα που παράγονται σε τρεις τύπους ανιχνευτών: τους ανιχνευτές θαλάμων ολίσθησης (Drift Tubes, DT) στην περιοχή του βαρελιού, τους ανιχνευτές θαλάμων καθοδικών λωρίδων (Cathode Strip Chambers, CSC) στην περιοχή των άκρων, και τους θαλάμους επιφανειών υψηλής αντιστάσεως (Resistive Plate Chambers, RPC) που καλύπτουν και τις δύο περιοχές. Στην περιοχή του βαρελιού, το σύστημα σκανδαλισμού αποτελείται από 60 ηλεκτρονικές κάρτες (TwinMUX), που συλλέγουν τα δεδομένα από τους θαλάμους DT και τους θαλάμους RPC, και από 12 ευρετές τροχιών (Barrel Muon Track Finders - BMTF). Η αρχιτεκτονική του συστήματος σκανδαλισμού φαίνεται στην εικόνα 2.

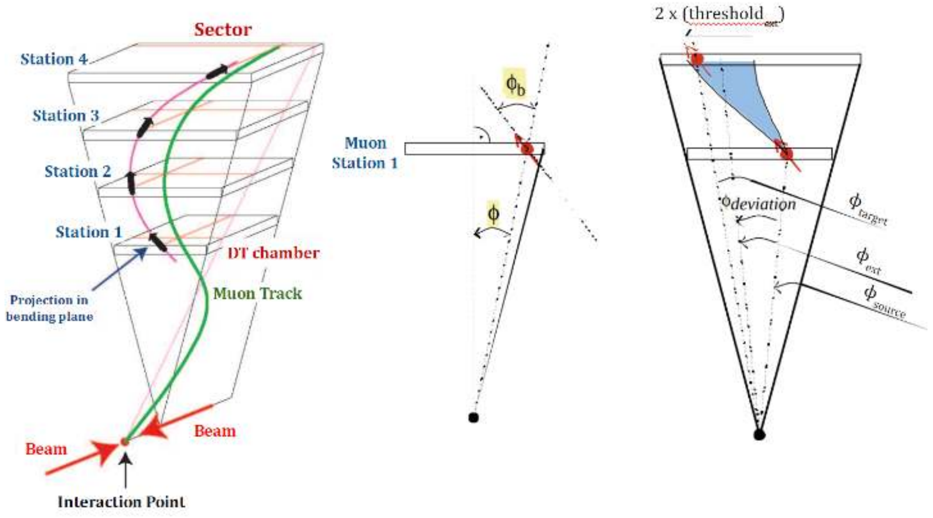


Εικόνα 2: Η αρχιτεκτονική του πρώτου επιπέδου του συστήματος σκανδαλισμού του CMS κατά την διάρκεια του Run 2.

Το σύστημα σκανδαλισμού μιονίων στην περιοχή του Βαρελιού

Το σύστημα σκανδαλισμού των μιονίων του CMS στην περιοχή του Βαρελιού, καλύπτει την περιοχή με ψευδοωκύτητα $0.83 < |\eta|$. Οι ανιχνευτές DT και RPC ομαδοποιούνται σε 12 τμήματα (wedges) ανοίγματος 30 μοιρών στο ϕ. Κάθε τμήμα χωρίζεται σε 5 τομείς (sectors), και κάθε sector αποτελείται από 4 DTs και 3 RPCs. Τα αναλογικά σήματα των ανιχνευτών, από κάθε sector, μετατρέπονται σε ψηφιακά και αποστέλλονται σε μια κάρτα πολυπλεξίας και διανομής, την TwinMUX. Οι κάρτες αυτές συνδυάζουν την πληροφορία από τα DTs και τα RPCs, κατασκευάζοντας τα λεγόμενα super-primitives, τα οποία αποστέλλονται στους 12 ευρετές τροχιών μιονίων (BMTF). Κάθε ευρετής τροχιών λαμβάνει πληροφορία από τους 5 sectors, που ανήκουν στο ίδιο Wedge, και από τους 5 sectors των δυο γειτονικών wedges.

Ο BMTF συνδυάζει τα super-primitives και ανακατασκευάζει τις τροχιές των μιονίων, ξεκινώντας από τα ίχνη που βρίσκονται πλησιέστερα στο σημείο αλληλεπίδρασης. Για τον συνδυασμό των ιχνών, χρησιμοποιεί τις χωρικές συντεταγμένες τους (ϕ-hits, η-hits), τη γωνία διέλευσης του μιονίου (ϕ_b), και προσπαθεί να τα αντιστοιχίσει με ίχνη σε συγκεκριμένες περιοχές των πιο απομακρυσμένων σταθμών ανίχνευσης. Τα όρια αυτά είναι προϋπολογισμένα και αποθηκευμένα στο FPGA σε πίνακες αντιστοίχησης (LUTs). Η μονάδα συναρμολόγησης της τροχιάς (Track Assembler Unit) συνδυάζει όλα τα πιθανά ζεύγη ιχνών, και συνθέτει μία πλήρη τροχιά. Για την ανακατασκευή μιας τροχιάς, απαιτούνται τουλάχιστον δύο ίχνη. Τέλος, η ηλεκτρονική μονάδα αντιστοίχησης (Assignment Unit), χρησιμοποιώντας επίσης LUTs, αντιστοιχεί σε κάθε τροχιά τις αντίστοιχες φυσικές παραμέτρους (εγκάρσια ορμή p_T, διεύθυνση ϕ και ψευδοωκύτητα η) (εικόνα 3).



Εικόνα 3: Ο αλγόριθμος εύρεσης και ανακατασκευής τροχιών μιονίων του συστήματος σκανδαλισμού του CMS, στην περιοχή του βαρελιού, κατά την διάρκεια του Run 2.

Η λογική προ-υπολογισμού και αποθήκευσης των παραμέτρων των μιονίων σε LUTs, καθιστά απαγορευτική τη χρήση περισσότερων από δύο ιχνών για τον υπολογισμό του p_T, καθώς ο αριθμός συνδυασμών ξεπερνά τις δυνατότητες αποθήκευσης του FPGA. Για την αύξηση της ακρίβειας της μέτρησης του p_T, ο αλγόριθμος υποθέτει ότι τα μιόνια προέρχονται από το σημείο αλληλεπίδρασης, προσθέτοντας ένα τρίτο σημείο στην τροχιά του. Με αυτή τη μέθοδο βελτιώνει την ποιότητα μέτρησης του p_T, αλλά καθιστά απαγορευτική την ανακατασκευή τροχιών “μετατοπισμένων” μιονίων. Τα μετατοπισμένα μιόνια είναι δυνατόν να προέλθουν από μακρόβια, ουδέτερα σωματίδια, τα οποία μπορούν να διασπαστούν μακριά από το αρχικό σημείο αλληλεπίδρασης των πρωτονίων. Για παράδειγμα θα μπορούσαν να προέλθουν από τη διάσπαση υπερσυμμετρικών σωματίων, όπως το προβλεπόμενο Higgsino.

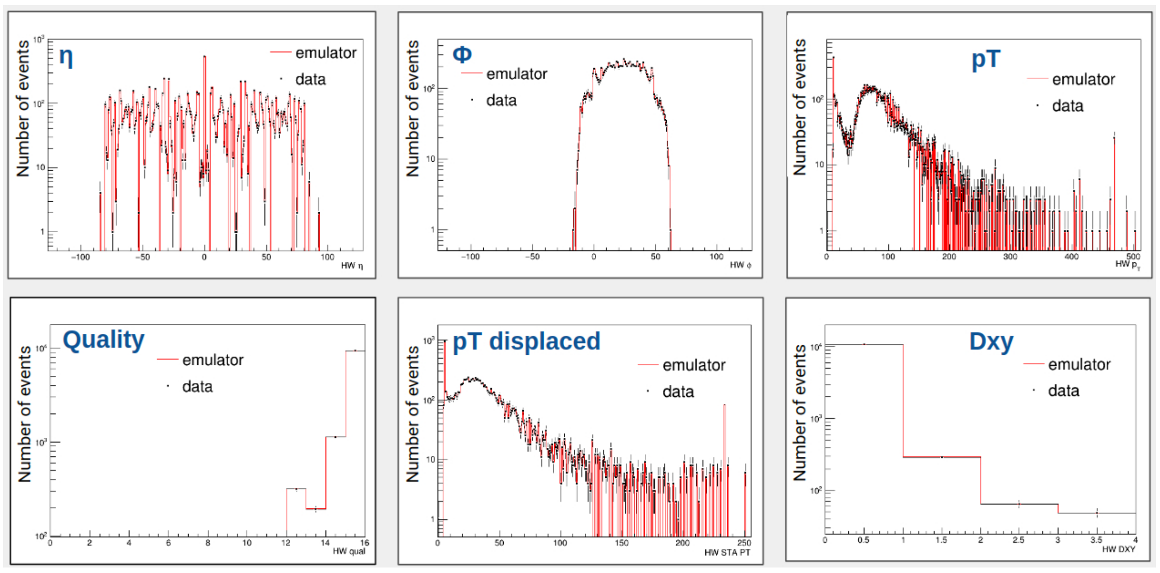
Ο νέος αλγόριθμος σκανδαλισμού μιονίων με χρήση ενός φίλτρου Kalman

Ένας νέος αλγόριθμος, με χρήση φίλτρου Kalman, προτάθηκε στις αρχές του 2018 για να επιλύσει το πρόβλημα ανίχνευσης μετατοπισμένων μιονίων. Το φίλτρο Kalman είναι ένας αλγόριθμος που εφαρμόζεται σε συστήματα που δέχονται εξωτερικές φυσικές διαταραχές (θορύβους), με σκοπό τον "καθαρισμό" των μετρήσεων, και τη δημιουργία μιας νέας εκτίμησης της κατάστασης του συστήματος, απογυμνωμένη από διαταραχές.

Ξεκινώντας από τον εξωτερικό σταθμό ανίχνευσης μιονίων, οι παράμετροι της τροχιάς $(p_T, quality, \phi, \phi_b)$ δίνονται από ένα διάνυσμα κατάστασης χ_n. Ο αλγόριθμος χρησιμοποιεί έναν πίνακα μετάβασης κατάστασης *F*, που εξαρτάται από τη γεωμετρία του ανιχνευτή και το μαγνητικό πεδίο *B*, ώστε να προβλέψει τη θέση στον επόμενο σταθμό ανίχνευσης. Αφού υπολογιστεί η επόμενη κατάσταση, χρησιμοποιούνται οι παράμετροι της πλησιέστερης μέτρησης, που δίνονται με την μορφή ενός διανύσματος μετρήσεων z_n, και οι παράμετροι του διανύσματος κατάστασης ενημερώνονται. Η διαδικασία επαναλαμβάνεται, μέχρι τον τελευταίο (εσώτερο) σταθμό ανίχνευσης. Σε αυτό το στάδιο, η μέτρηση της εγκάρσιας ορμής (p_T) του μιονίου αποθηκεύεται, χωρίς την απαίτηση προέλευσης του μιονίου από το σημείο αλληλεπίδρασης, επιτρέποντας και μετατοπισμένα μιόνια. Στη συνέχεια, μια δεύτερη τιμή της p_T υπολογίζεται, αυτή τη φορά με τον περιορισμό διέλευσης της τροχιάς από το σημείο αλληλεπίδρασης, παρομοίως με τον παλιό αλγόριθμο.

Ο νέος αλγόριθμος σχεδιάστηκε αρχικά για το Phase-2, αλλά ενσωματώθηκε στο σύστημα σκανδαλισμού του Phase-1, παράλληλα με τον παλιό αλγόριθμο. Αυτό κατέστη εφικτό λόγω της εκτεταμένης χρήσης DSPs (σε αντίθεση με τον παλιό αλγόριθμο), της ευελιξίας στο συνολικό διαθέσιμο latency του BMTF, και της περιορισμένης χρήσης πόρων του FPGA από τον παλιό αλγόριθμο. Ο αλγόριθμος Kalman αναπτύχθηκε σε γλώσσα C++, και στην συνέχεια μεταγλωττίστηκε σε VHDL, με τη χρήση του λογισμικού *Vivado High Level Synthesis* της Xilinx. Για την ενσωμάτωση του νέου αλγορίθμου, αναπτύχθηκε κώδικας που υλοποιεί την διεπαφή του με το firmware της BMTF. Οι παράμετροι εισόδου τροποποιήθηκαν κατάλληλα, λαμβάνοντας υπόψιν τις ιδιαιτερότητες του νέου αλγορίθμου. Επιπλέον, πραγματοποιήθηκε βελτιστοποίηση του firmware, με την προσθήκη επιπλέον καταχωρητών και επιλογή ειδικών στρατηγικών, ώστε όλα τα στοιχεία του νέου αλγόριθμου να είναι συγχρονισμένα. Η ορθή λειτουργία του νέου firmware, ελέγχθηκε εκτεταμένα στο εργαστήριο, με την ανάπτυξη λογισμικού και firmware (test benches). Τέλος το νέο firmware ελέγχθηκε με πραγματικά δεδομένα στο CMS.

Στην εικόνα 4 μπορούμε να δούμε μια σύγκριση των παραμέτρων των μιονίων, μεταξύ των πραγματικών δεδομένων με αυτά του εξομοιωτή (emulator). Παρατηρείται πολύ καλή συμφωνία, της τάξης του 99.6%. Το όριο για να γίνει αποδεκτός ένας νέος αλγόριθμος από το CMS είναι 99%, το οποίο και επιτεύχθηκε.

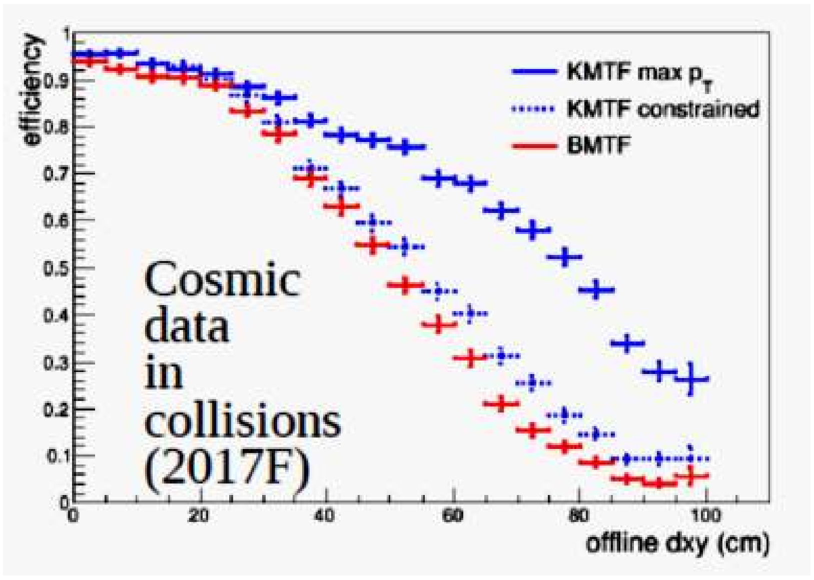


Εικόνα 4: Σύγκριση των παραμέτρων των μιονίων, μεταξύ των πραγματικών δεδομένων με αυτά του εξομοιωτή (emulator).

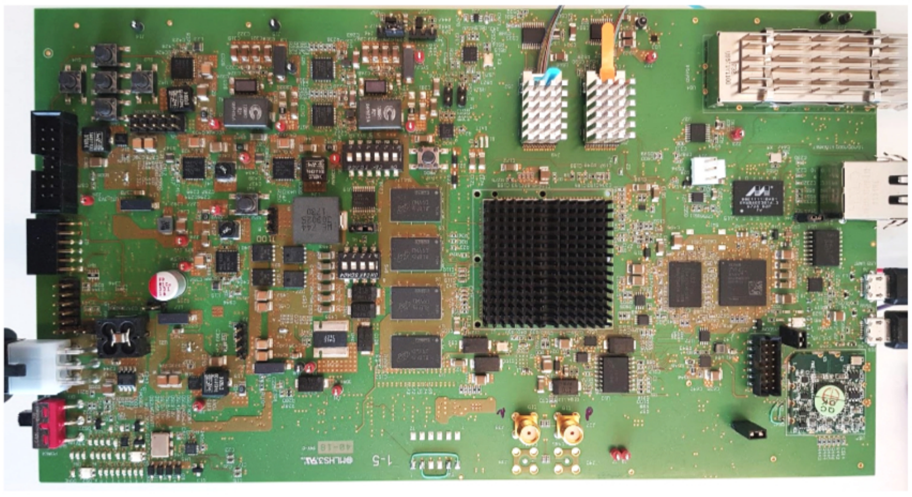
Στην εικόνα 5, φαίνεται η απόδοση ανίχνευσης μετατοπισμένων μιονίων με τη χρήση του παλιού αλγορίθμου (με κόκκινο) και με τη χρήση του νέου αλγορίθμου, με τον περιορισμό του σημείου αλληλεπίδρασης (με γαλάζιο) και χωρίς (με μπλε). Είναι εμφανής η βελτίωση για μιόνια με προέλευση από 20 έως 100 cm μακριά από το σημείο αλληλεπίδρασης.

Η δοκιμαστική ηλεκτρονική κάρτα L1-BMT

Στο πλαίσιο της έρευνας και ανάπτυξης νέου υλισμικού για την περίοδο Phase-2 του LHC, σχεδιάστηκε και κατασκευάστηκε μια νέα αναπτυξιακή ηλεκτρονική κάρτα, η L1-BMT, για το σύστημα σκανδαλισμού του CMS στην περιοχή του βαρελιού (εικόνα 6). Η κάρτα αυτή, παρέχει ένα περιβάλλον για την ανάπτυξη και την αξιολόγηση νέων αλγορίθμων και τεχνολογιών. Στην καρδιά της κάρτας βρίσκεται το XCKU040, ένα Kintex Ultrascale FPGA τελευταίας γενιάς από την Xilinx, που παρέχει, μεταξύ άλλων, 20 Multi-Gigabit transceivers (MGT) με μέγιστη ταχύτητα 16 Gbps. Δώδεκα MGTs είναι συνδεδεμένοι με τα οπτικά στοιχεία Fireflys, της Samtec, υπεύθυνες για τη μετατροπή του ηλεκτρικού σήματος σε οπτικό. Επιπλέον, τέσσερις MGTs συνδέθηκαν με ένα QSFP. Τα Fireflys έχουν επιλεγεί από το πείραμα CMS για να χρησιμοποιηθούν στις ηλεκτρονικές κάρτες του συστήματος σκανδαλισμού του Phase-2. Η κάρτα, επίσης, παρέχει δύο προγραμματιζόμενες γεννήτριες συχνοτήτων υψηλής ακρίβειας, και έναν διαφορικό SMA connector, για το χρονισμό της λογικής και των transceivers του FPGA. Επιπλέον, όλα τα προγραμματιζόμενα στοιχεία της κάρτας (π.χ γεννήτριες συχνοτήτων, Fireflys, QSFP κτλ) ελέγχονται από ένα System-on-Chip FPGA (Zynq XC7Z010). Τέλος, το Kintex FPGA και το Zynq μπορούν να προγραματιστούν μέσω JTAG.



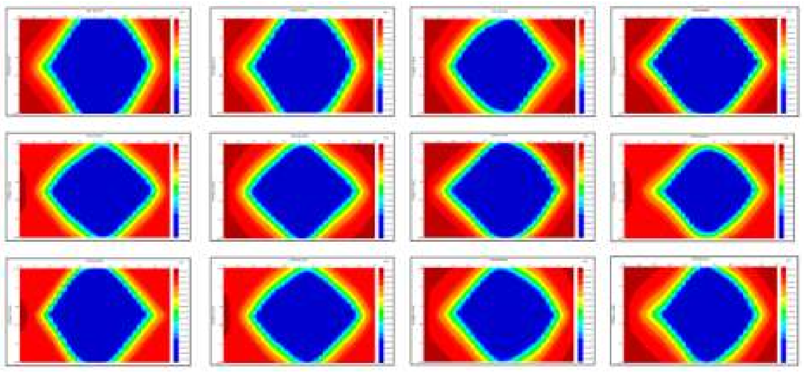
Εικόνα 5: Απόδοση ανίχνευσης μετατοπισμένων μιονίων του παλιού αλγορίθμου (με κόκκινο), και του νέου αλγορίθμου με τον περιορισμό της διέλευσης από το σημείο αλληλεπίδρασης (με γαλάζιο) και χωρίς (με μπλέ).



Εικόνα 6: Η αναπτυξιακή ηλεκτρονική κάρτα L1-BMT, για το σύστημα σκανδαλισμού του CMS στην περιοχή του βαρελιού.

Η λειτουργία των οπτικών links της κάρτας ελέγχθηκε χρησιμοποιώντας το ενσωματωμένο Bit Error Rate Test (iBERT) της Xilinx. Το iBERT μας δίνει τη δυνατότητα υπολογισμού του ρυθμού εισαγωγής σφαλμάτων κατά τη μετάδοση δεδομένων, καθώς και την ανοχή του καναλιού σε θόρυβο, με τη χρήση των λεγόμενων eyescans. Τα eyescans είναι δισδιάστατα ιστογράμματα τα οποία κατασκευάζονται προσθέτοντας ένα offset στον χρόνο (οριζόντιος άξονας) και στην τάση (κατακόρυφος άξονας) της λαμβανόμενης ψηφιακής πληροφορίας, και μετρώντας κάθε φορά το BER.

Στην εικόνα 7, φαίνονται τα eyescans για τα 12 Firefly links στα 16 Gbps. Παρατηρούμε ότι η μπλε περιοχή, που αντιστοιχεί σε μηδενικό BER, είναι ευρεία και βρίσκεται στο κέντρο του διαγράμματος, πράγμα που υποδεικνύει πολύ καλή ποιότητα καναλιού με ανοχή στον θόρυβο.



Εικόνα 7: Eyescans για τα 12 Firefly links στα 16 Gbps.

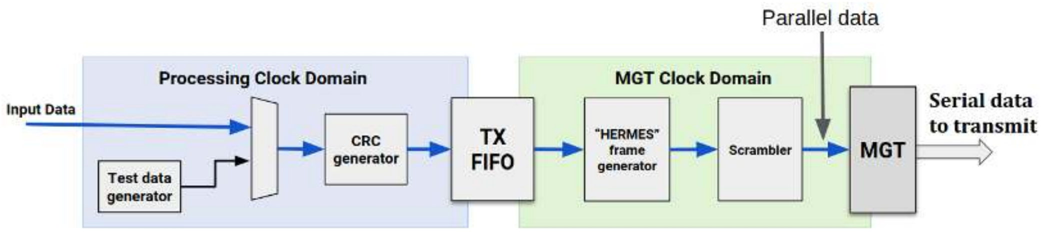
Το πρωτόκολλο επικοινωνίας "Hermes".

Ακολουθώντας τα υπόλοιπα υποσυστήματα του πειράματος, το πρωτόκολλο επικοινωνίας των links του συστήματος σκανδαλισμού του CMS θα πρέπει να αναβαθμιστεί, ώστε να αντεπεξέλθει στις νέες απαιτητικές συνθήκες που θα προκύψουν μετά την αναβάθμιση του LHC. Στο πλαίσιο της αναβάθμισης αυτής, σχεδιάστηκε το πρωτόκολλο επικοινωνίας "Hermes", βασιζόμενο στα τελευταίας γενιάς Ultrascale και Ultrascale+ FPGAs, με ενσωματωμένους transceivers GTH/GTY, που φτάνουν σε ταχύτητες 32 Gbps. Το νέο πρωτόκολλο ενσωματώνει την κωδικοποίηση 64b66b, η οποία προσθέτει μόνο 3.125% επιπλέον πληροφορίας (overhead), έναντι 20% της παλιάς κωδικοποίησης 8b10b. Για να καλύψει τις ανάγκες του νέου συστήματος σκανδαλισμού, το πρωτόκολλο θα πρέπει να έχει τα εξής βασικά χαρακτηριστικά :

1. Χαμηλό latency, λόγω του περιορισμένου συνολικού διαθέσιμου latency του συστήματος σκανδαλισμού.
2. Περιορισμένη χρήση πόρων στο FPGA, που θα αφήνει περισσότερο χώρο για τη λογική των αλγορίθμων.
3. Ασύγχρονη λειτουργία με το ρολόι του LHC, το οποίο παρουσιάζει διακυμάνσεις στη φάση του, ικανές να απο-συγχρονίσουν τους transceivers.
4. Συγχρονισμός και αυτόματη ανάκτηση της σύνδεσης, ώστε να εξοικονομηθεί πολύτιμος χρόνος, σε περίπτωση προσωρινής διακοπής στην αποστολή δεδομένων.

5. Ανίχνευση σφαλμάτων, η οποία θα επιτρέπει τον έλεγχο της ορθότητας των δεδομένων.

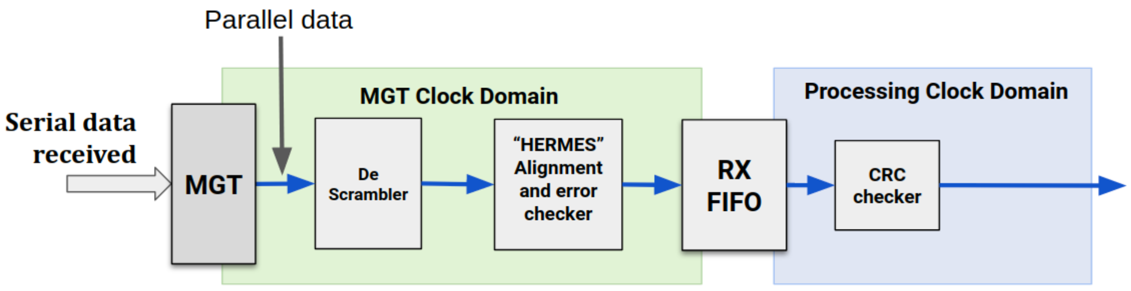
Στην εικόνα 8, φαίνεται ένα απλοποιημένο σχηματικό διάγραμμα του πομπού. Σε κανονική λειτουργία, τα δεδομένα προς μετάδοση προέρχονται από τους αλγόριθμους. Σε λειτουργία ελέγχου, μπορούμε να επιλέξουμε να στείλουμε δεδομένα από μια γεννήτρια ψευδοτυχαίων αριθμών ή από έναν μετρητή 16-bit. Τα δεδομένα εισέρχονται στη γεννήτρια CRC, η οποία παράγει ένα checksum, που αποστέλλεται μετά το τέλος της ροής δεδομένων. Στη συνέχεια, τα δεδομένα, μέσω μιας μνήμης FIFO, περνούν από το ρολόι του LHC στο πιο σταθερό, και ελαφρώς πιο γρήγορο, ρολόι του transceiver. Όταν η μνήμη αδειάσει, διακόπτουμε προσωρινά την ανάγνωση, εισάγοντας ψευδο-δεδομένα (filler blocks). Στη συνέχεια, τα δεδομένα κωδικοποιούνται σύμφωνα με το πρωτόκολλο “Hermes”. Κωδικοποιημένες λέξεις προστίθενται για να δηλώσουν την αρχή (Start of Frame - SoF) και το τέλος (End of Frame - EoF) της ροής δεδομένων, ενώ όλα τα δεδομένα που δεν θα χρησιμοποιηθούν από τους αλγόριθμους, αντικαθίστανται από κωδικοποιημένες λέξεις (IDLEs). Στη συνέχεια, τα δεδομένα εισέρχονται σε έναν περιπλέκτη (scrambler), ο οποίος τυχαιοποιεί την αλληλουχία των bits, δίνοντάς τους τα κατάλληλα ηλεκτρικά χαρακτηριστικά. Τέλος, τα δεδομένα εισέρχονται στον transceiver, ο οποίος τα διαμορφώνει κατάλληλα, τα μετατρέπει σε παράλληλα, και τα αποστέλλει.



Εικόνα 8: Απλοποιημένο σχηματικό διάγραμμα του πομπού.

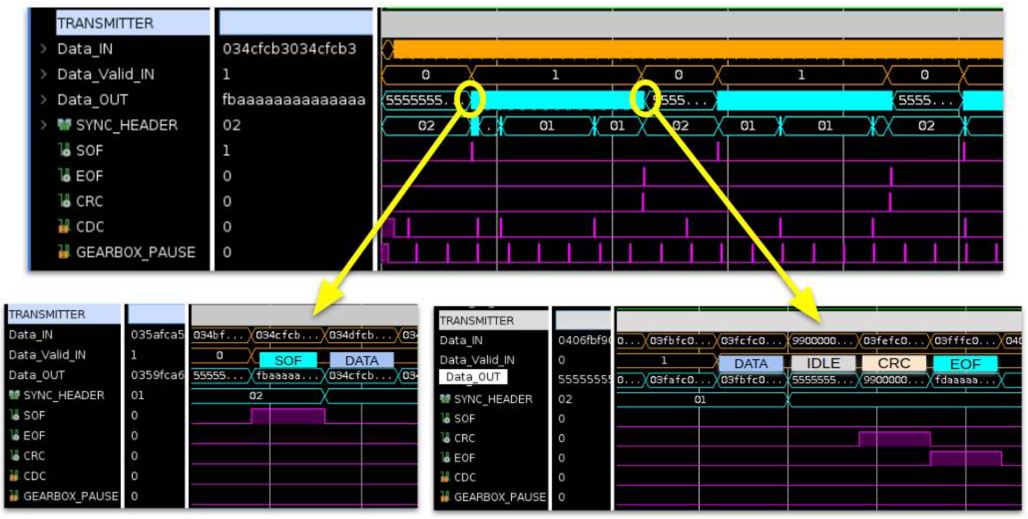
Στην εικόνα 9, φαίνεται ένα απλοποιημένο σχηματικό διάγραμμα του δέκτη. Τα δεδομένα λαμβάνονται από τον δέκτη του transceiver, μετατρέπονται σε παράλληλα, και στέλνονται στον απο-περιπλέκτη (de-scrambler). Στη συνέχεια, οι κωδικοποιημένες λέξεις του πρωτοκόλλου και ο 2-bit header της 64b66b κωδικοποίησης, ελέγχονται για σφάλματα. Στο στάδιο αυτό, πραγματοποιείται και η ευθυγράμμιση του link. Για κάθε 16 συνεχόμενα σφάλματα, ο δείκτης που καθορίζει το πρώτο bit των παράλληλων λέξεων στον transceiver, μετακινείται κατά μια θέση. Με την ορθή λήψη των δεδομένων, η ευθυγράμμιση έχει ολοκληρωθεί. Η διαδικασία επαναλαμβάνεται αυτόματα, σε περίπτωση λήψεως μεγάλου αριθμού σφαλμάτων, που υποδεικνύουν απώλεια του συγχρονισμού. Μετά τον έλεγχο, τα filler blocks αφαιρούνται, και με την χρήση μίας μνήμης BRAM μεταβαίνουν από το ρολόι του transceiver στο ρολόι του LHC. Τέλος, μια γεννήτρια CRC αναπαράγει το checksum, χρησιμοποιώντας την λαμβανόμενη ροή δεδομένων, και το συγκρίνει με το αντίστοιχο checksum που υπολογίστηκε στον πομπό. Σε περίπτωση διαφωνίας, αυξάνει κατά μία μονάδα ένας καταχωρητής καταμέτρησης σφαλμάτων.

Ο έλεγχος ορθής λειτουργίας του κωδικοποιητή πραγματοποιήθηκε με προσομοίωση του firmware, χρησιμοποιώντας το λογισμικό *Vivado* της Xilinx. Στην εικόνα 10, παρουσιάζεται η διαμόρφωση των δεδομένων, με την προσθήκη των κωδικοποιημένων λέξεων στην αρχή (SoF) και



Εικόνα 9: Απλοποιημένο σχηματικό διάγραμμα του δέκτη.

στο τέλος (CRC, EoF) της ροής δεδομένων. Ο έλεγχος της κωδικοποίησης πραγματοποιήθηκε και σε επίπεδο hardware, αποθηκεύοντας τα λαμβανόμενα δεδομένα σε μια προσωρινή μνήμη στον δέκτη, η οποία διαβάζεται με ειδικό λογισμικό που αναπτύχθηκε.



Εικόνα 10: Προσομοίωση του κωδικοποιητή. Προσθήκη των κωδικοποιημένων λέξεων SoF (κάτω αριστερά) και CRC, EoF (κατω δεξιά).

Για τη μέτρηση του latency των links, χρησιμοποιήθηκε ο ενσωματωμένος λογικός αναλυτής της Xilinx, και ελήφθησαν μετρήσεις, για ταχύτητες 16 Gbps και 25 Gbps. Η μέθοδος που χρησιμοποιήθηκε ήταν η αποστολή μιας δοκιμαστικής λέξης, και η μέτρηση των κύκλων ρολογιού μέχρι αυτή να εμφανιστεί στον πομπό. Στα 16 Gbps, με συχνότητα του παράλληλου ρολογιού ίση με 240 MHz, μετρήθηκαν 23 κύκλοι (ή ισοδύναμα 3.8 BXs[1]). Αντίστοιχα, στα 25 Gbps, με συχνότητα του παράλληλου ρολογιού ίση με 360 MHz, μετρήθηκαν 27 κύκλοι (ή ισοδύναμα 3 BXs).

Τέλος, για την αξιολόγηση της σταθερότητα των links στα 16 Gbps, χρησιμοποιήθηκαν

[1]Ένα BX ή Bunch Crossing, είναι ίσο με την περίοδο του LHC clock, δηλαδή 25 ns.

διάφορες αναπτυξιακές κάρτες (KCU105, L1-BMT), στέλνοντας δεδομένα σε εσωτερικό loopback, αλλά και μέσω οπτικής ίνας μήκους 20 m. Και στις δύο περιπτώσεις, στάλθηκαν δεδομένα επί 72 ώρες, χωρίς να παρουσιαστεί σφάλμα. Αυτό μεταφράζεται σε $BER < 2.41 \times 10^{-16}$. Για τον έλεγχο των links στα 25 Gbps, χρησιμοποιήθηκε η κάρτα Serenity, η οποία σχεδιάστηκε και κατασκευάστηκε από το Imperial College για το Phase-2. Επίσης στάλθηκαν δεδομένα για 72 ώρες, χωρίς να παρουσιαστεί σφάλμα, γεγονός που μεταφράζεται σε $BER < 1.54 \times 10^{-16}$.

Chapter 1

Introduction

The Large Hadron Collider (LHC) is the world's largest and most powerful particle accelerator at the CERN research laboratory. It consists of a 27-kilometre ring of superconducting magnets, with a number of accelerating structures to boost the energy of the particles. LHC's main purpose is to accelerate protons or ions in the form of two high-energy beams travelling in opposite directions at a speed close to the speed of light. The beams collide at four points, where the two rings of the machine intersect and at each point a detector was built around the collision point.

The Compact Muon Solenoid (CMS) detector is one of the two large multi-purpose detectors operating at the LHC. Its physics program ranges from studying the Standard Model, including the Higgs boson, to searching for extra dimensions and particles that could make up dark matter. Great emphasis, however, has been given to the discovery of the predicted Higgs boson, due to its key role in understanding the nature of the electroweak symmetry breaking. The experimental study of the Higgs mechanism, can also shed light on the mathematical consistency of the Standard Model at energy scales above about 1 TeV.

During Phase-1 of the LHC program (2011-2023) the instantaneous luminosity reached $2 \times 10^{34}\ cm^{-1}s^{-1}$, with a pile-up of about 50. This number of events produces an enormous amount of raw data that exceeds 40 Petabytes per second. In order to deal with that amount of data, a sophisticated electronics system was built, responsible of reducing this rate by at least a factor of 10^6, through a physics event selection process. The rate reduction is performed in two steps; the Level-1 Trigger (L1T) and the High Level Trigger (HLT). The L1T is a high-bandwidth, fixed latency system based on Field-Programmable Gate Arrays (FPGAs) that receives data from the calorimeters and the muon system and reduces the rate to 100 kHz.

After the end of Run 3 (2023) the LHC will undergo a major upgrade. The new configuration, known as High Luminosity LHC (HL-LHC), will rely on a number of key innovations that push accelerator technology beyond its present limits. The main target of the HL-LHC is to increase the peak instantaneous luminosity to $5 \times 10^{34}\ cm^{-2}s^{-1}$. This will allow an integrated luminosity of 250 fb^{-1} per year, with the goal of 3000 fb^{-1} in over 10 to 12 years after the upgrade. The CMS collaboration will have to upgrade the experiment, in order to cope effectively with the higher pile up and maintain data quality, by limiting the degradation due to aging and radiation damage. Furthermore, the trigger system must increase its selectivity and the capacity of the readout system. CMS has decided to increase the L1T latency to $12.5\ \mu s$ from $4\ \mu s$ for the legacy system, while the read out data rate at a higher rate limit will increase to 750 kHz compared to 100 kHz of the legacy trigger.

In chapter 2, a short description of the LHC is given, followed by the presentation of the CMS detector and its subsystems in chapter 3. A small summary of the technologies being used by the various electronics systems of the CMS trigger follows in chapter 4 and a detailed presentation of the L1T system is presented in chapter 5.

Contributions and thesis overview

The main contribution of this dissertation was the improvement of the Level-1 Trigger Muon Track Finder algorithms in the Central Region of the CMS detector, as well as related R&D projects for the HL-LHC.

Firstly, a low latency asynchronous protocol for high speed links was designed, implemented and tested. The protocol was designed to meet the needs of the Phase-2 CMS trigger systems and will run at rates of 16 - 28 Gbps, improving the performance of the current 10 Gbps links up to 400%. The firmware was tested in various CMS Phase-2 board demonstrators (*Serenity* [1], *L1-BMT*, *Ocean* [2]) , using Xilinx Ultrascale and Ultrascale+ FPGAs. An extensive description of the firmware, the link functionality and the test results are presented in chapter 7.

At the end of 2017, an upgrade of the Level 1 muon tracking algorithm was presented by the CMS UCLA group, featuring an FPGA embedded Kalman Filter, implemented using High Level Synthesis (HLS) [3] tools. The possibility of testing the algorithm was investigated with real data during Run 2 (2017 - 2018). The firmware was integrated in the CMS data taking, running in parallel with the current algorithm, by implementing both algorithms into the same FPGA. This required tackling many challenges like latency optimization, area optimization, meeting timing constraints and running an HLS algorithm in parallel with the old algorithm written in VHDL. The new algorithm will be the default standalone algorithm for Run 3 (2021-2023). This is considered a breakthrough, since it was the he first time an HLS based algorithm was used at a CMS trigger subsystem. The new algorithm is optimised to identify displaced muons, creating new opportunities for the physics groups searching for long-lived particles. An overview of the algorithm and the integration, testing and commissioning of the new firmware is presented in chapter 6.

Finally, within the scope of Phase-2 R&D, a new L1T processor card was designed by the Greek CMS Trigger team, to provide a hardware environment for developing and evaluating the new L1T muon algorithms and links. The board comes with state-of-the-art fiber optics technologies, using micro footprint optical interconnects. An overview of the board along with the testing and debugging of the high-speed optical links and the integration of the *Hermes* link protocol is presented in chapter 6.

Chapter 2

The Large Hadron Collider

The Large Hadron Collider is the world's largest and most powerful particle accelerator and its main purpose is to accelerate protons or ions. It consists of a 27-kilometre ring of superconducting magnets, with a number of accelerating structures to boost the energy of the particles. The particles form two high-energy beams travelling in opposite directions at a speed close to the speed of light. They collide at four points where the two rings of the machine intersect. Thousands of magnets of different varieties and sizes are used to direct the beams around the accelerator. These include 1232 dipole magnets, 15 metres in length, that bend the beams, and 392 quadrupole magnets, each 5–7 metres long, which focus the beams. Just prior to collision, another type of magnet is used to "squeeze" the particles closer together to increase the chances of collisions. The particles are so tiny that the task of making them collide is akin to firing two needles 10 kilometres apart with such precision that they meet halfway.

The electromagnets are built from coils of special electric cable that operates in a superconducting state, efficiently conducting electricity without resistance or loss of energy. This requires chilling the magnets to -271.3℃ - a temperature colder than outer space. For this reason, much of the accelerator is connected to a distribution system of liquid helium, which cools the magnets, as well as to other supply services. The LHC structure is housed 100 metres underground, in a circular tunnel once in use by another machine, Large Electron-Positron Collider (LEP), that was dismantled in 2000. Hadrons are injected into the LHC from a chain of smaller accelerators that drives them up to higher energies.

2.1 The LHC injector chain

Protons are created in an ion source called *duoplasmatron*, from which they are extracted with an energy of 50 KeV. They are then injected to LINAC2, a 35 meter long Linear Accelerator and their energy is increased to 50 MeV. The beam is further increased to 1.4 GeV, in the Proton Synchrotron

Booster (PSB), before they continue their journey to the Proton Synchrotron (PS). Here the particles are grouped into a train of bunches of 10^11 protons, 1.2 m long and 7 m apart. Their energy increases to 26 GeV.

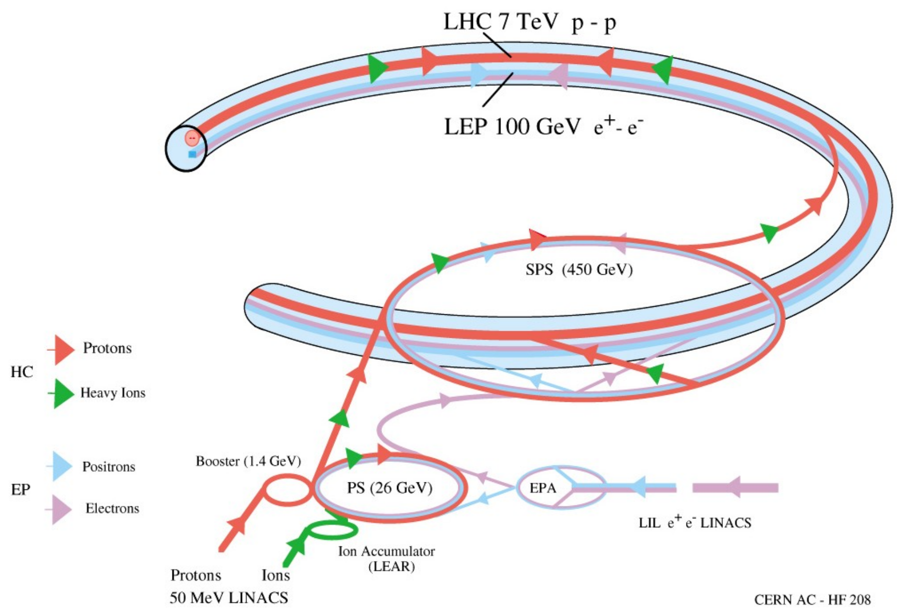


Figure 2.1: Illustration of the CERN accelerator complex. The LHC is supplied with protons or heavy ions by the SPS via two transfer lines, which additionally provide beams to several non-LHC experiments. The SPS is fed with pre-accelerated protons by the PS, CERN's first synchrotron. The first circular accelerator in the proton-proton injection chain is PSB which is filled by the linear accelerator LINAC2. For beams of heavy ions the injection chain begins with LINAC2 which feeds PS via the LEIR.

The next machine in the chain is the Super Proton Synchrotron (SPS), the second-largest machine in CERN's accelerator complex, measuring nearly 7 km in circumference. It accelerates the bunches of protons to 450 GeV and provide beams for the Large Hadron Collider, the NA61/SHINE, the NA62 and the COMPASS experiments. It is a cycling machine, with the cycles having a period of about 10 s and it includes 1317 conventional (room-temperature) electromagnets, including 744 dipoles to bend the beams round the ring. The transverse beam size is largest at injection and decreases with the square root of the beam energy during acceleration. When the SPS operates as LHC injector, up to 288 bunches are accelerated, each bunch with about 1.1×10^{11} protons. The bunch length is $0.5\ ns$ and two neighboring bunches are separated by 25 ns so that the duration of the entire beam is about 7 μs [4].

Ions are generated in the *Lead LINAC* (LINAC3), however, they do not pass via the PSB, but instead are injected, accumulated and accelerated in the Low Energy Ion Ring (LEIR). From this accumulator, the ions are transferred to the PS, and then they follow the proton route [5].

The Cern Control Center (CCC) controls this extensive process by joining all of the accelerator operators with the engineering and cryogenics departments. By coordinating the process of injection, the CCC guarantees a high-quality beam.

2.2 Luminocity of the LHC

Once the proton beams have reached nominal energies and have acquired stabilised orbits inside the LHC ring, they are brought to collision in the four Interaction Points (IPs) at which the detectors are positioned. The rate of a certain event R_{event} for a given interaction with cross-section, σ_{event}, is defined as

$$R_{event} = \mathcal{L}\sigma_{event}, \tag{2.1}$$

where $\mathcal{L}$ is the instantaneous luminosity for two oppositely directed beams. This instantaneous luminosity only depends on the beam parameters and can be written as

$$\mathcal{L} \equiv f_{rev}\frac{N_1 N_2}{4\pi\sigma_x\sigma_y}. \tag{2.2}$$

where $\mathcal{L}$ is the *instantaneous luminosity* for two oppositely directed beams of relativistic particles with *revolution frequency* f_{rev}. The terms N_1 and N_2 are the number of protons within the two beam bunches. The quantity $4\pi\sigma_x\sigma_y$ is the *effective section* of collision that depends on the cross section of the bunch ("effective" because the beam profile doesn't have a sharp edge).

We can also express the Luminosity in terms of ϵ (*emittance*) and β (*amplitude function*) as

$$\mathcal{L} = f_{rev}\frac{N_1 N_2}{4\pi\epsilon\beta^*}. \tag{2.3}$$

The transverse emittance, ϵ , and the amplitude function, β, both express the beam size. A low emittance particle beam is a beam where the particles are confined to a small distance and have nearly the same momentum. The amplitude function at the IP, β^*, is determined by the accelerator's magnet configuration (basically, the quadrupole magnet arrangement) and powering. If β^* is low, the beam is narrower, "squeezed", while if β^* is high, the beam is wide and straight. Clearly it is desired β^* to be as small as possible. How small depends on the capability of the hardware to make a near-focus at the IP. It can be deduced that the LHC peak luminosity is constrained by the machine's ability to make high population bunches of low emittance to collide at high frequency at locations where the beam optics provide as low values of the amplitude functions as possible.

The integral of the delivered luminosity over time is called *Integrated Luminosity*. It is a measurement of the collected data size, and it is an important value to characterize the performance of an accelerator because it directly relates to the number of observed events. The integral (Equation 2.4) is taken over the *sensitive time*, i.e., excluding possible dead time:

$$L = \int \mathcal{L} dt. \tag{2.4}$$

It is expressed in inverse cross section. The next graph shows the integrated luminosity delivered to the ATLAS and CMS experiments during different LHC runs. The 2018 run produced 65 fb^{-1} of data, which is 16 fb^{-1} more than in 2017 (Figure 2.2).

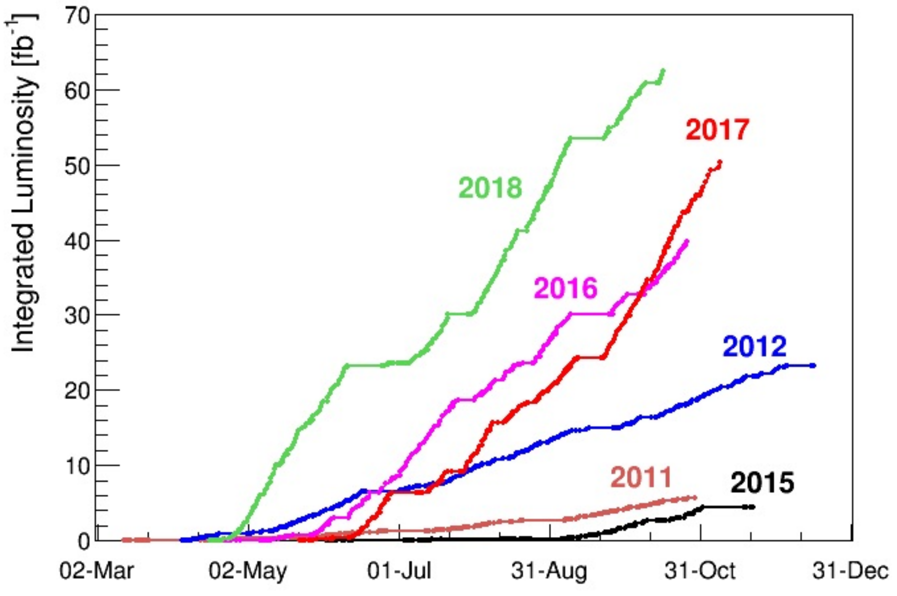


Figure 2.2: Integrated luminosity delivered to the ATLAS and CMS experiments during different LHC runs (Image: CERN).

The luminosity in the LHC is not constant over a physics run, but decays due to the degradation of intensities and emittances of the circulating beams. The main cause of the luminosity decay during nominal LHC operation is the beam loss from collisions. The decay of the beam intensity and luminosity as functions of time is given by:

$$\mathcal{L} = \frac{\mathcal{L}_0}{(1 + \frac{t}{\tau_{nuclear}})^2} \tag{2.5}$$

where $\mathcal{L}_0$ is the *initial luminosity* and $\tau_{nuclear}$ is the *initial decay time* of the bunch intensity due to beam collisions. The time required to reach $1/e$ of the initial luminosity is $\tau_{nuclear,1/e} \approx 29\ h$.

Other contributions to beam losses come from *Touschek scattering* and from particle losses due to a slow emittance blow-up. Emittance blow-up can be caused by the scattering of particles on residual gas, the nonlinear force of the beam-beam interaction, RF noise and IBS scattering effects. Approximating further the decay by an exponential process, the net luminosity lifetime can be estimated as:

$$\frac{1}{\tau_L} = \frac{1}{\tau_{IBS}} + \frac{1}{\tau_{restgas}} + \frac{1}{\tau_{nuclear,1/e}} \tag{2.6}$$

Integrating the luminosity over one run yields:

$$L = L_0 \tau_L [1 - e^{-T_{RUN}/\tau^L}] \tag{2.7}$$

where T_{run} is the *total length* of the luminosity run. The overall collider efficiency depends on the ratio of the length of the run to the average turnaround time [4].

2.3 LHC experiments

Seven experiments, at the Large Hadron Collider, use detectors to analyze particles produced by collisions in the accelerator. CMS, ATLAS, ALICE, LHCb, TOTEM, LHCf and MoEDAL are run by collaborations of scientists from institutes all over the world. Each experiment is distinct, and characterized by its detectors.

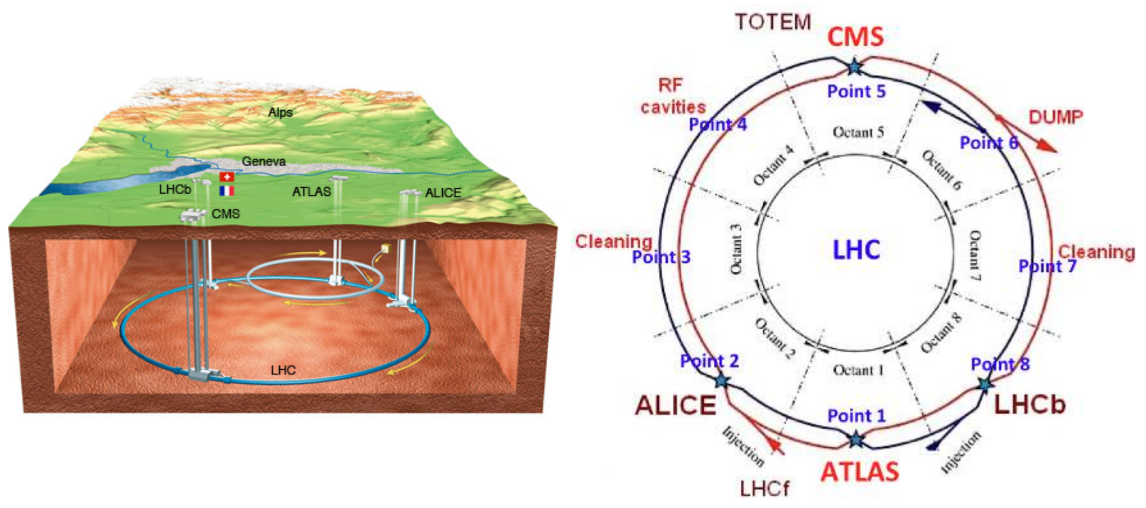


Figure 2.3: LEFT: A map of the LHC with the four bigger experiments. **RIGHT**: Layout of the LHC, showing separation into octants, with beam-1 (beam-2) rotating clockwise (anti-clockwise). The four large experiments, the beam injection points, the RF-cavities location, the beam cleaning and the dump locations are shown (taken from [6]).

- **CMS** is one of the two general-purpose detectors with the purpose of investigating the largest range of physics possible. Its broad physics program ranges from studying the *Standard Model* (including the Higgs boson) to searching for extra dimensions and particles that could make up dark matter. Built around a huge solenoid magnet that generates a field of 4 T, it was constructed in 15 sections at ground level, before being lowered and reassembled into an underground cavern, near the french village of Cessy. The complete detector measures 21 m long, 15 m wide and 15 m high, weighting more than 14,000 ton.

- **ATLAS** is the second general-purpose particle physics experiment and, together with CMS, is designed to exploit the full discovery potential and the huge range of physics opportunities that the LHC provides. As with CMS, ATLAS 's scientific exploration uses precision measurements to test the predictions of the *Standard Model* which encapsulates our current understanding of what the building blocks of matter are and how they interact.

- **ALICE** experiment is optimized to study heavy-ion (Pb-Pb nuclei) collisions at a centre of mass energy of 2.76 TeV per nucleon pair. The high temperature and energy density produce *quark-gluon plasma* in conditions similar to the ones existed a fraction of the second after the *Big Bang*, before quarks and gluons bound together to form hadrons and heavier particles. The existence of the quark-gluon plasma and its properties are key issues in *quantum chromodynamics* for understanding color confinement and chiral symmetry restoration [7].

- **LHCb** is an experiment dedicated to heavy flavour physics. Its primary goal is to look for indirect evidence of new physics in cp violation and rare decays of beauty and charm hadrons. The experiment allows physicists to study cp violation and rare decays of B_d, B_s and *D-mesons* with high statistics and using many different decay modes [8].

- **TOTEM** is an independent experiment, but technically integrated into CMS. Its main purpose is to measure the total p-p cross section, with a method independent of the luminosity, and study elastic diffractive scattering. It consists of two tracking telescopes, installed on each side of pseudo-rapidity region $3.1 \leq |\ \eta\ | \leq 6.5$ and *Roman Pot* stations placed at distances of $\pm$ 147 m and $\pm$ 220 m from the CMS interaction Point (P5) [9].

- **LHCf** is intended to measure the energy and numbers of *neutral pions* (π_0) produced by the collider, in an effort to explain the origin of ultra-high-energy cosmic rays. It consists of two similar calorimeters placed symmetrically, at a distance of 140 m from the ATLAS Interaction Point, covering the very forward region of the LHC [10].

- **MoEDAL** is designed to enhance, in a complementary way, the physics reach of the LHC. The prime motivation of MoEDAL is to directly search for the Magnetic Mono-pole and other highly ionizing Stable (or pseudo-stable) Massive Particles (SMPs) at the LHC. MoEDAL is using a passive plastic track technique which does not require a trigger. It is, thus, optimized to detect slow moving particles that the main LHC experiments cannot detect due to the very short LHC trigger window of 25 ns. [11]

2.4 The High Luminocity LHC upgrade

The commissioning of the Large Hadron Collider marked the beginning of a remarkable era for cosmology, astrophysics and high energy physics. It is since then, at the forefront of attempts to understand the fundamental nature of the universe. The discovery of the Higgs boson in 2012 was a major milestone in the history of science, but beyond this, the LHC has the potential to go on and help answer some key questions; the existence of super-symmetry, the nature of dark matter, the existence of extra dimensions and the properties of the Higgs boson are some of the most important of them. To extend its discovery potential, the world's most powerful accelerator has to be upgraded.

In the 2020s, the LHC will go through major upgrade to extend its operability by another decade and to increase its luminosity (and thus collision rate) by a factor of five, beyond its initial design value. The Integrated Luminosity design goal is an increase by a factor of ten. The necessary developments require about 10 years to prototype, test and realize new equipment. The updated machine configuration, the so-called HL-LHC, will rely on a number of key innovative technologies representing exceptional technological challenges.

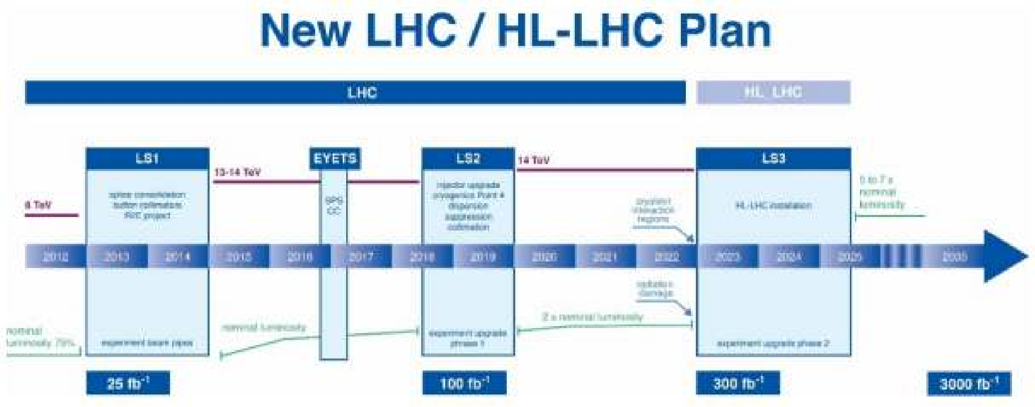


Figure 2.4: LHC baseline plan showing the collisions energy (upper red line) and luminosity (lower green lines). The first long shutdown (LS1) in 2013-2014 allowed the design parameters of beam energy and luminosity to be reached. The second long shutdown (LS2) in 2018-2019, will consolidate luminosity and reliability as well as the upgrading of the LHC injectors. After LS3 (2023-2025), the machine will be in the High Luminosity configuration (HL-LHC)[12].

In spring 2016, the CERN Council approved the proposal, describing the goals of the upgrade, the physics case and the technology challenges of the HL-LHC. This proposal [13] covers all the project period including installation and commissioning by 2026 [12]. The main objective of the HL-LHC design study was to determine a set of beam parameters and the hardware configuration that will enable the LHC to reach the following targets:

- a peak luminosity of 5×10^{34} $cm^{-2}s^{-1}$ with levelling, and

- an integrated luminosity of $250 fb^{-1}$ per year with the goal of $3000 fb^{-1}$ in 10-12 years after the upgrade.

Regarding the hardware, several systems need to be improved, and possibly replaced, to allow the LHC machine to achieve luminosity above the current ultimate peak luminosity of $2 \times 10^{34}\ cm^{-2}s^{-1}$. The main hardware upgrades required are:

- **Inner triplet magnets**. Possible radiation damage to the triplet quadrupoles could cause sudden electric breakdown, entailing serious and long repairs. Replacement of the low-beta triplet is a long intervention, requiring a one- to two-year shutdown and must be coupled with major detector upgrades.
- **Cryogenics**. The installation of a new cryogenics plant that will allow full separation between superconducting rf and magnet cooling is planned. This will increase intervention flexibility and machine availability.
- **Collimation**. The collimation system will require an upgrade that takes into account the need for the lower impedance imposed by the planned increased beam intensities. A new configuration will also be required to protect the new triplets.
- **Electronics boards of the power converter system**. Considerable effort is being made to study how to replace the radiation-sensitive electronics boards of the power converter system with radiation-hard cards.
- **Quench Protection System(QPS)), machine protection and remote manipulation**.
  - **QPS** for the superconducting magnets is based on a design that is almost 20 years old.
  - **Machine protection** will require improvement.
  - **Remote manipulation** While full robotics is difficult to implement, given the conditions on the ground, remote manipulation, enhanced reality and supervision are the keys to minimize the radiation doses sustained during interventions.

2.4.1 The HL-LHC luminocity

As already mentioned, the major upgrade of the LHC will allow us to push the machine peak levelled luminosity to about $5 \times 10^{34}\ cm^{-2}s^{-1}$, while the *pile-up* (number of events per bunch crossing) will increase to around 200. This luminosity level should enable the collection of up to $300 - 350\ fb^-1$ per year, provided that the experiments can digest this pile-up level. The luminosity profile without levelling, however, quickly decreases from the initial peak value due to "*luminosity burn-off*". A further limitation upon peak luminosity is imposed by the consideration of energy deposition by collision debris in the interaction region magnets, and the necessity to limit the peak pile-up in the experimental detector.

These considerations lead to a luminosity profile with optimized run time and levelling at $\approx 5 \times 10^{34}\ cm^{-2}s^{-1}$) (Figure 2.5). The average luminosity achieved with levelling is almost the same as that without levelling in an ideal running configuration without premature fill aborts. In terms of total integrated luminosity, the ultimate performance corresponds to a value of about $4000\ fb^{-1}$.

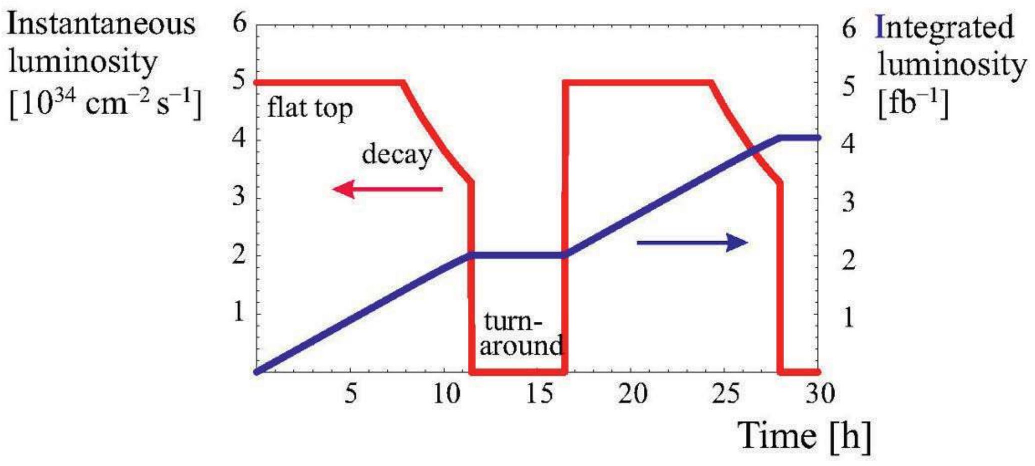


Figure 2.5: Luminosity cycle for HL-LHC with levelling and a short decay (optimized for integrated luminosity)[12].

Chapter 3

The CMS experiment

The Compact Muon Solenoid detector is one of the two large multi-purpose detectors operating at the LHC at CERN. It is installed 100 m underground, at Interaction Point 5 (P5), outside of the French village of Cessy, between Lake Geneva and the Jura mountains. It has a broad physics program, ranging from studying the Standard Model, to searching for extra dimensions and particles that could make up dark matter. Great emphasis, however, has been given to the discovery of the Higgs boson, due to its key role in understanding the nature of the electroweak symmetry breaking. The experimental study of the Higgs mechanism, can also shed light on the mathematical consistency of the Standard Model at energy scales above 1 TeV.

The unique conditions of the LHC require a very careful design of the detectors. At the design luminosity, a mean of about 20 inelastic collisions is superimposed on the event of interest. This implies that around 1000 charged particles emerge from the interaction region every 25 ns. The products of an interaction under study may be mixed with products from other interactions in the same bunch crossing. This problem, clearly, becomes more severe when the response time of a detector element and its electronic signal is longer than 25 ns. This effect is called pile-up and can be reduced by using high-granularity detectors with good time resolution, resulting in low occupancy. This requires the use of millions of detector electronic channels with very good synchronization. CMS detector observes an event rate of approximately 10^9 inelastic events per second, which leads to a number of experimental challenges. The online event selection process (trigger) must reduce the huge rate to about 1000 events per second for storage and subsequent analysis.

The CMS detector, as can be seen in Figure 3.1, houses a high-magnetic field created by a large-bore superconducting solenoid, surrounding a silicon pixel and strip tracker, an electromagnetic calorimeter and a sampling hadron calorimeter. The iron yoke of the flux-return is instrumented with four stations of muon detectors, covering most of the 4π solid angle. Forward sampling calorimeters extend the pseudorapidity coverage to high values ($|\eta| < 5$), assuring good hermeticity and stereoscopic coverage. The overall dimensions of the CMS detector are a length

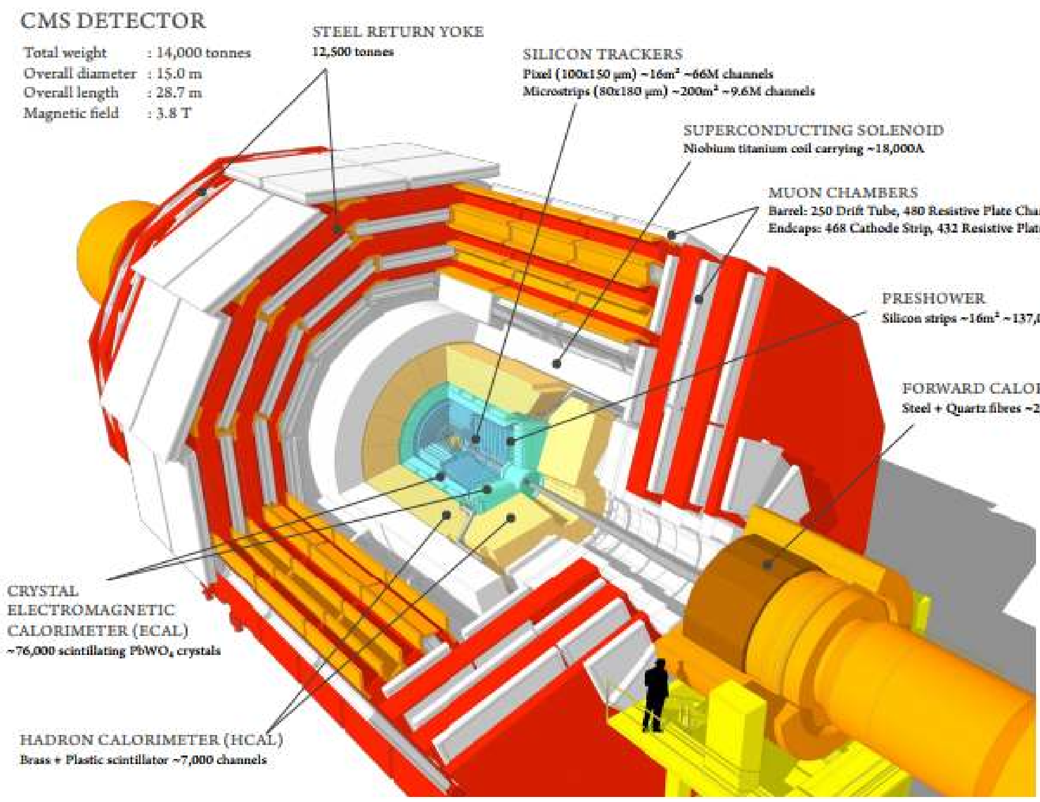

CMS DETECTOR
Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T
STEEL RETURN YOKE
12,500 tonnes
SILICON TRACKERS
Pixel (100x150 μm) ~16m² ~66M channels
Microstrips (80x180 μm) ~200m² ~9.6M channels
SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying ~18,000A
MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Resistive Plate Chambers
Endcaps: 468 Cathode Strip, 432 Resistive Plate Chambers
PRESHOWER
Silicon strips ~16m² ~137,000 channels
FORWARD CALORIMETER
Steel + Quartz fibres ~2,000 Channels
CRYSTAL ELECTROMAGNETIC CALORIMETER (ECAL)
~76,000 scintillating PbWO₄ crystals
HADRON CALORIMETER (HCAL)
Brass + Plastic scintillator ~7,000 channels

Figure 3.1: A cutaway view of the CMS detector [14].

of 21.6 m, a diameter of 14.6 m and a total weight of 12.5 kton.

The CMS detector requirements, in order to meet the goals of the LHC physics program, can be summarised as follows:

- ▶ Good muon identification and momentum resolution over a wide range of momenta and angles, a good di-muon mass resolution ($\sim$1% at 100 GeV), and the ability to determine, unambiguously, the muons charge with $p < 1$ TeV.
- ▶ Good momentum resolution and reconstruction efficiency in the inner tracker for charged particles. Efficient triggering and offline tagging of τ-particles and *b-jets*, which implies the use of pixel detectors close to the interaction region.
- ▶ Good electromagnetic energy resolution, good di-photon and di-electron mass resolution ($\sim$ 1% at 100 GeV), wide geometric coverage, π^0 rejection, and efficient photon and lepton isolation at high luminosity.
- ▶ Good *missing transverse energy* and di-jet mass resolution, which requires the use of hadron calorimeters with large hermetic geometric coverage and with fine lateral segmentation.

3.1 The CMS Coordinate system

In the CMS coordinate system, the origin is centered at the nominal collision point, inside the experiment, the y-axis pointing vertically upwards, and the x-axis pointing radially inwards toward the center of the LHC. Thus, the z-axis points along the beam direction toward the Jura mountains from LHC Point 5. The azimuthal angle ϕ is measured from the x-axis in the x-y plane. The polar angle, θ, is measured from the z-axis. The pseudorapidity, η, is defined as:

$$\eta = -\ln \tan(\frac{\theta}{2}). \tag{3.1}$$

The momentum and energy measured transverse to the beam direction, denoted by p_T and E_T, respectively, are computed from the x and y components. The imbalance of energy measured in the transverse plane equals to the missing transverse energy and it is denoted by E_T^{miss}.

3.2 The superconducting solenoid magnet

As highlighted before, it is crucial for the CMS to identify muons and in general charged particles with good momentum and efficiency. To achieve these goals and measure precisely the momentum of energetic charged particles, large bending power is required. This scale of bending power can only be achieved with the use of superconducting magnets. The superconducting magnet of the CMS detector has been designed to reach a 4 T field in a free bore of 6 m diameter and 12.5 m length with a stored energy of 2.6 GJ at full current. The flux is returned through a 10,000 ton yoke, comprising of 5 wheels for the barrel region and 2 endcaps composed of three disks each (Figure 3.2 and Figure 3.3).

The distinctive feature of the 220 ton cold mass is the 4-layer winding, made from a stabilised, reinforced NbTi conductor. The ratio between stored energy and cold mass is high (11.6 kJ/kg), causing a large mechanical deformation ($\sim$0.15%) during energising, well beyond the values of previous solenoidal detector magnets.

The single most important aspect of the overall detector design is the configuration and parameters of the magnetic field for the measurement of muon momenta. The parameters of the CMS magnet are summarised in Table 3.1. The magnet was assembled and tested in a surface hall (SX5), prior to being lowered 90 m below ground to its final position in the experimental cavern. After provisional connection to its ancillaries, the CMS magnet, was fully and successfully tested and commissioned in SX5 by the end of autumn 2006.

General parameters	
Magnetic length	12.5 m
Cold bore diameter	6.3 m
Central magnetic Induction	4 T
Total Ampere-turns	41.7 MA-turns
Nominal current	19.14 kA
Inductance	14.2 H
Stored energy	2.6 GJ
Cold Mass	
Layout	Five modules mechanically and electrically coupled
Radial thickness of cold mass	312 mm
Radiation thickness of cold mass	Weight of cold mass
Weight of cold mass	220 t
Maximum induction on conductor	4.6 T
Temperature margin wrt operating temperature	1.8 T
Stored energy/unit cold mass	11.6 kJ/kg
Iron yoke	
Outer diameter of the iron yoke	14 m
Lenght of barrel	13 m
Thickness of the iron layers in barrel	300, 630 mm
Mass of iron in barrel	6000 t
Thickness of iron disks in endcaps	250, 600 and 600 mm
Mass of iron in each endcap	2000 t
Total mass of iron in return yoke	10 000 t

Table 3.1: Main parameters of the CMS magnet

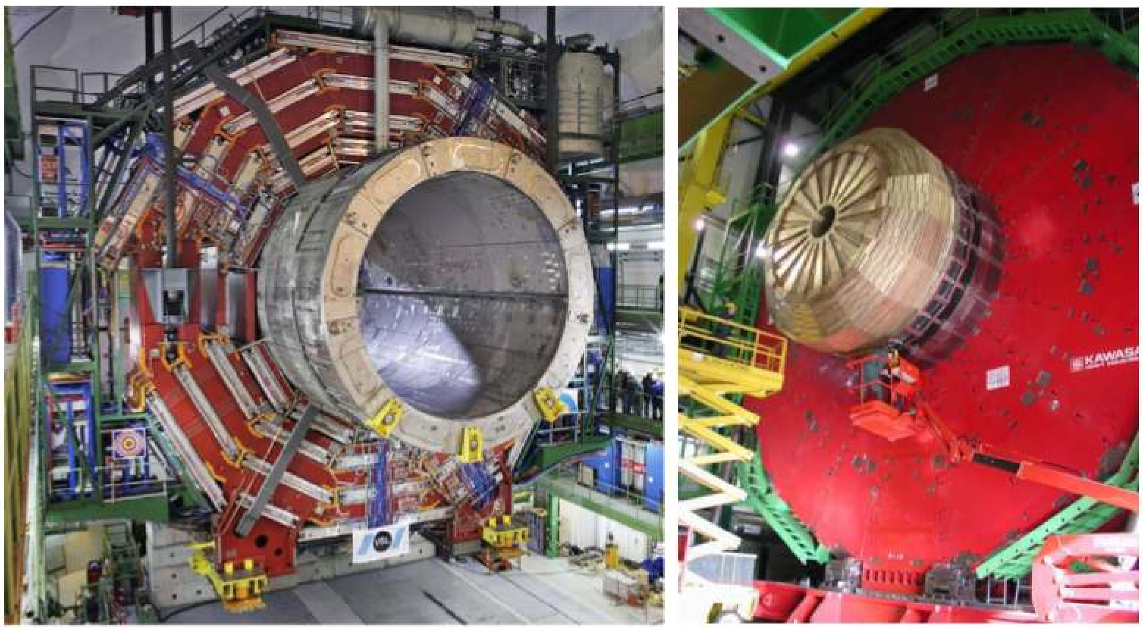


Figure 3.2: **LEFT**: The Compact Muon Solenoid magnet and one of the five barrel yoke wheels. **RIGHT**: View of one endcap disk (out of six), 15 m in diameter. This 600 mm thick disk, weighing 700 ton, is supporting a 300 ton hadronic endcap calorimeter

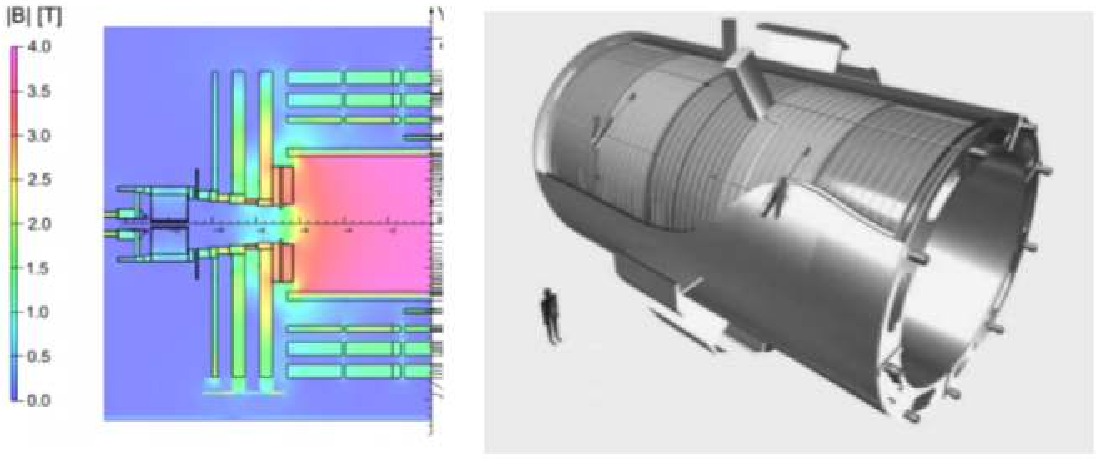


Figure 3.3: **LEFT**: Value of |B| predicted on a longitudinal section of the CMS detector for the underground model at a central magnetic flux density of 3.8 T [15]. **RIGHT**: Perspective view of the CMS coil inside the outer vacuum tank showing the five modules, the tie-bar suspension system, and the thermosyphon cooling circuits outside the mandrels [16].

3.3 The Inner Tracking System Overview

The inner tracking system is the innermost part of the CMS detector. It is designed to provide a precise and efficient measurement of the trajectories of charged particles emerging from the LHC collisions. It is very important, for the tracking system, to distinguish primary and *secondary vertices* [1]. Furthermore, determining precisely the secondary vertices and the impact parameters d_{xy} and d_z, is required for the efficient identification of heavy flavours.

The inner tracking system surrounds the thin beryllium beam-pipe which accommodates the LHC vacuum and is centred at the IP where the two beams collide. Its total tracking volume is given by a cylinder of 5.8 m length and 2.6 m diameter. The CMS solenoid provides a homogeneous magnetic field of 4 T over the full volume of the tracker.

During the development phase of the tracker system, it was calculated that for the LHC design luminosity of $1 \times 10^{34}\ cm^{-2}s^{-1}$, an average of about 1000 charged particles, that originate from more than 20 overlapping proton-proton interactions, would be produced every 25 ns. The inner tracker is the closest system to the interaction point and is subjected to an enormous particle flux. Therefore, a detector technology featuring high granularity and fast response was selected. The high granularity insures that the trajectories can be identified reliably, and the detector's fast response the trajectory's assignment to the correct bunch crossing. The expected tracker hit rate density was 100 MHz/cm^2 in the barrel part, at a radius of 4 cm. This requires high power on-detector electronics and an efficient cooling system. A compromise was made, however, to keeping the amount of tracker material to as little as possible, to minimise phenomena such as multiple scattering, bremsstrahlung, photon conversion and nuclear interactions.

3.3.1 The Silicon Strip Tracker

The CMS Silicon Strip Tracker (SST), together with the pixel detector, provides measurement of the charged particle trajectories up to a pseudorapidity of $|\eta| < 2.5$. The layout of the SST is shown in Figure 3.4. The detector is 5 m long with a diameter of 2.5 m, and it is segmented into 9.6 million single-sided *p-on-n* micro-strip sensors, distributed over 15148 modules. They provide 198 m^2 of active silicon area.

The barrel region consists of 10 layers, with 4 layers in the Tracker Inner Barrel (TIB) and 6 layers in the Tracker Outer Barrel (TOB). In the forward region, the detector consists of two Tracker End-Cap (TEC) composed of 7 rings each, as shown in Figure 3.4. Stereo modules constructed from two silicon modules, glued back-to-back with their strips aligned at an angle of 100 mrad, are used in 4 layers of the barrel and 3 disks of the endcap, to provide the 2D measurements. The APV25 chip was designed for the readout of the silicon microstrip detector. It is a 128-channel analogue pipeline chip, developed to meet the demands of low-noise, low power and radiation hardness. Each channel comprises a low noise amplifier, a 192 cell analogue pipeline

[1] The point of decay of a collision product that was itself produced in the primary vertex

and a de-convolution readout circuit. Output data are transmitted on a single differential output via an analogue multiplexer [17]. They are then converted to optical signals on analog-opto-hybrids (AOH), and transmitted to Front-End Drivers (FEDs), located in the service cavern outside the radiation zone, via optical fibers. Pedestal and common mode subtraction, as well as cluster finding, are performed in the FEDs. Clock and trigger information is distributed by Front-End Controllers (FECs) to Communication and Control Units (CCU) grouped in token ring networks (control rings).

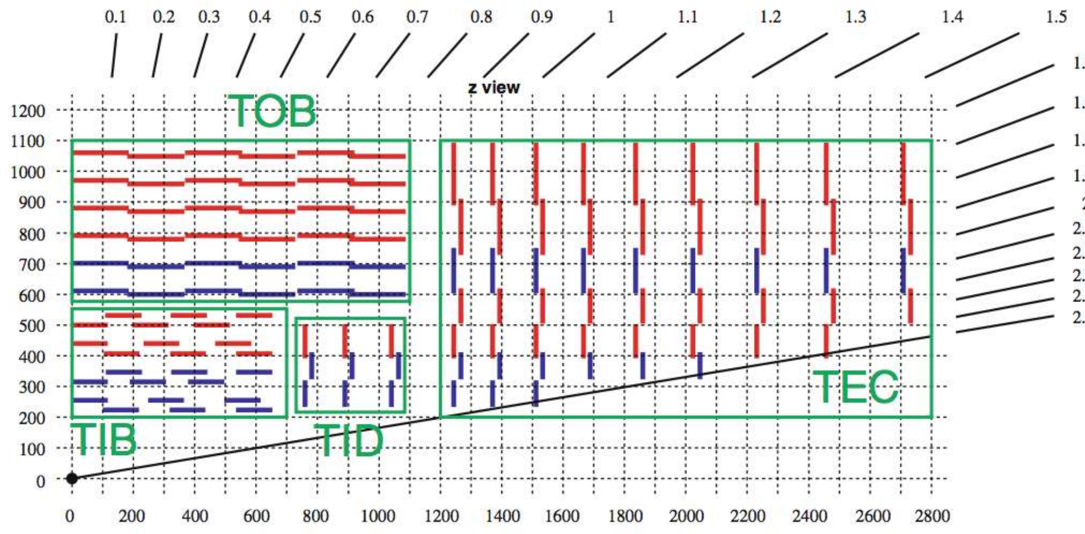


Figure 3.4: The layout of the CMS Silicon Strip Tracker [18].

The CMS Silicon Strip Tracker has shown no significant degradation after 10 years of operation, and it continues to deliver high quality data for physics analyses. At the beginning of 2018 the operational temperature of the detector has been changed from -15°C to -20°C, which helped to decrease the leakage current in regions with degraded cooling. The detector was re-calibrated and commissioned at the new operational temperature.

3.3.2 The Pixel Detector

The pixel detector is the part of the tracking system that is closest to the interaction region. It is essential for the reconstruction of secondary vertices from b and τ decays, and forming seed tracks for the outer track reconstruction and high level triggering. The tracks reconstructed, with the pixel detector alone, is the only available tracking information at the first stage of the High Level Trigger (HLT). In this stage, speed is more important than accuracy or efficiency. Pixel-only tracks can be reconstructed in less than 20 ms (110 ms) per event for regional (global) track finder.

The pixel detector covers a pseudorapidity range of $|\eta| < 2.5$, matching the acceptance of the central tracker. The Barrel Pixel (BPix) consists of three layers and the Forward Pixel (FPix) (endcap) of two disks on each side. The BPix layers are 53 cm long and located at mean radii of 4.4, 7.3 and 10.2 cm respectively, covering an area of 0.78 m^2. The FPix disks are placed on each side of the interaction point at $z = \pm 34.5$ cm and $z = \pm 46.5$ cm, extending from 6 to 15 cm in radius, covering an area of another 0.28 m^2. The total amount of pixels is 66 million, 48 millions contained in the Barrel Pixel and 18 millions in the Forward Pixel, with a pixel unit cell of $100\mu m \times 150\mu m$. This very fine granularity is important to avoid saturation and keep the tracker occupancy at the level of $\sim$1%, due to the high hit-rate density at very small distances around the Interaction Point. The geometric arrangement of the 3 barrel layers and the forward pixel disks (as shown in Figure 3.5) gives 3 tracking points over almost the full η-range. In the high η region the two disk points are combined with the lowest possible radius point from the 4.4 cm barrel layer.

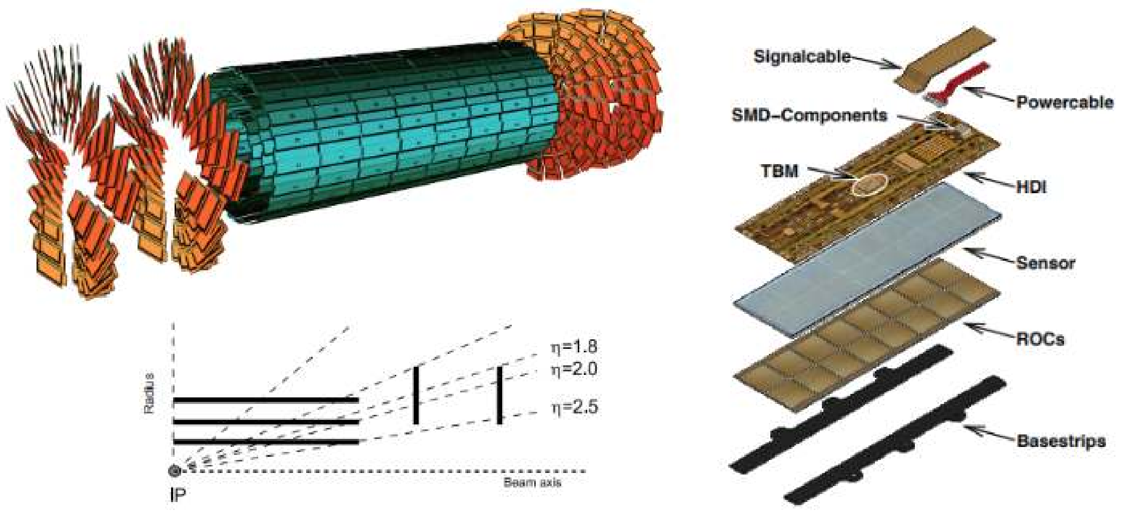


Figure 3.5: **LEFT**: Sketch (top) and geometrical layout (bottom) of the pixel detector. **RIGHT**: Exploded view of a barrel module (taken from [19], [20]).

The barrel layers are made out of two types of sensor modules, half modules and full modules. The three layers comprise of 128, 224 and 320 full modules respectively and 32 half modules each at the edges of the half-shells. The barrel module sensor is a pixelated, high dose, *n* implantation in a lightly *n-doped* bulk material. The backside is *p-doped* forming the *n* junction. The active area of a full (half) module sensor is $64.8mm \times 5.3mm$. It is a double sided processed “*n+ on n*” design. The inter-pixel isolation is done with a moderated *p-spray* technique. For the forward pixel the sensor design is of “*n+ on n-*” type. A *p-stop* isolation technique, in which a high dose *p-implant* surrounds each *n+* type region, provides explicit electrical isolation between neighboring *n+* electrodes.

Sixteen (eight) Read-Out Chips (ROCs) are bump-bonded to each full (half) sensor. The ROCs were fabricated in a commercial 0.25 μm five-metal-layered CMOS process. Their main purposes are amplification and buffering of the charge signal from the sensor, zero suppression in the pixel unit cell, level-1 trigger verification, sending hit information and configuration data out to the Token Bit Manager (TBM) chip and adjusting various voltage levels and offsets in order to compensate for chip-to-chip variations in the CMOS device parameters. The module is read out in a daisy chain scheme. The read out process is controlled by the TBM chip. For each level 1 trigger the TBM generates a token bit which controls 8 or 16 ROCs. One ROC at a time sends its hit information through a Low Voltage Differential Signal (LVDS) cable to the TBM chip where they are amplified and converted into a Low Current Differential Signal (LCDS). ([21], [22]). The fact that charge sharing is enhanced between pixels, due to Lorentz drift in the 4 T magnetic field, leads to increased spatial resolution through analog signal interpolation.

The pixel readout electronics were designed and optimized for the data rates and pixel occupancy expected for the LHC design luminosity of $1 \times 10^{34}\ cm^{-2}s^{-1}$, with 25 ns bunch spacing. For these conditions, there is a dynamic inefficiency of $\sim$ 4% from the current readout chip, in the innermost layer. At the nominal readout rate of 100 kHz, the data loss would have increased to 16% in the innermost layer as the luminosity went up to $2 \times 10^{34}cm^{-2}s^{-1}$. To avoid this degradation and maintain the excellent performance of the pixel detector at higher luminosity and pile-up, the pixel detector was replaced during the year-end technical stop of 2016/2017. The old 3-layer barrel / 2-disk endcap system was replaced with a 4-layer barrel / 3-disk endcap system, providing four hit coverage (Figure 3.6). The addition of the fourth barrel layer at a radius of 16 cm, provides redundancy in pattern recognition and reduces fake rates with high pile-up. Moreover, the mass of the new detector was significantly lowered by relocating much of the passive material out of the tracking volume.

The preliminary results of the detector performance showed that the primary goal for the upgrade, to ensure the acquisition of high-quality data at the increased instantaneous luminosity of the LHC, was met. The sensor efficiency of the new detector is better than 99% up to an instantaneous luminosity twice as high as the working instantaneous luminosity of the old system. Since the sensor design has not changed, the same hit resolution is observed as that of the old detector. The track reconstruction has been improved, by adding the extra layers in the detector, while the hit resolution remained the same. Finally, the initial operating temperature of the pixel detector was lowered from 0℃ to -15℃, after the first Long Shutdown (LS1) and further lowered to -20℃, after the installation of the upgraded detector. [23]

3.4 The Electromagnetic Calorimeter

The CMS Electromagnetic Calorimeter (ECAL) is placed inside the solenoid and surrounds the tracker. Its mission is to identify and measure precisely the energy of the charged particles (i.e photons, electrons and positrons) produced by the collision. ECAL played an essential role in the discovery of the Higgs boson back in 2012. The $H \to \gamma\gamma$ decay mode provided a distinctive signature for its discovery and this channel was the main benchmark while designing the ECAL (see Figure 3.7). This decay process produces a sharp peak above the background for Higgs boson masses of 140 GeV, which leads to a strict constraint of 1 % at 100 GeV on the ECAL energy resolution. Additional requirements are good angular resolution, as well as accurate π^0/photon separation.

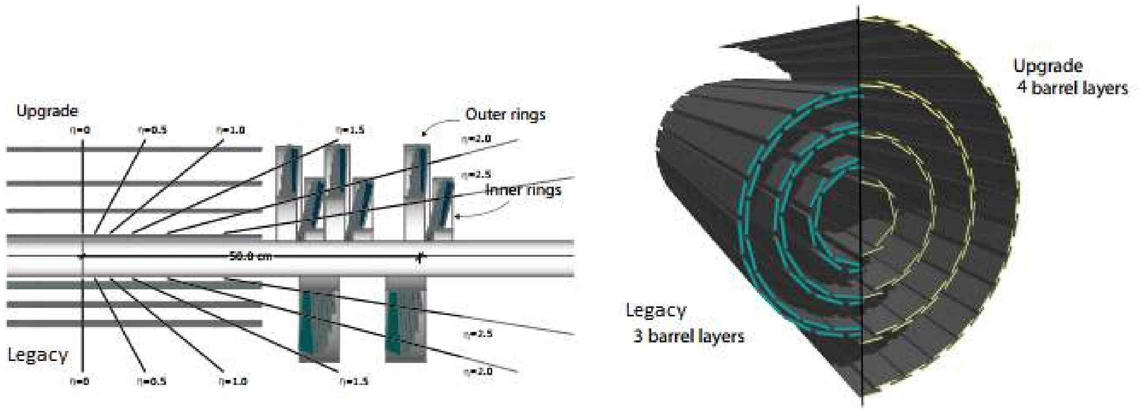


Figure 3.6: **Left**: Layout comparison of the old and current pixel detector. **Right**: Transverse view comparing the pixel barrel layers (taken from [24]).

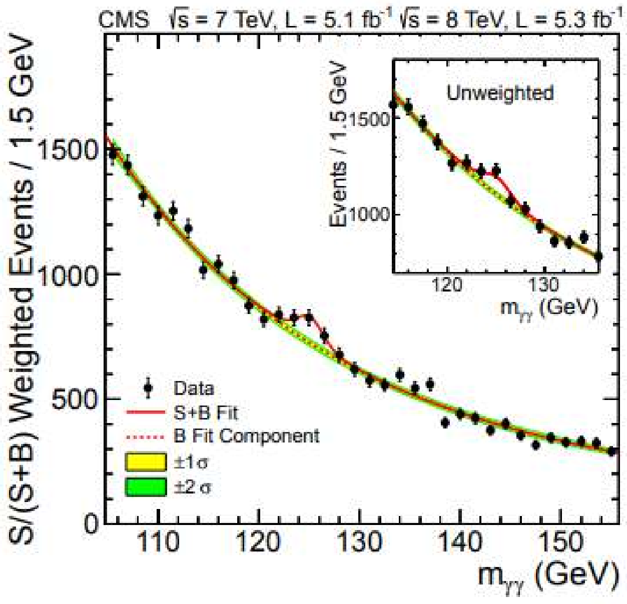


Figure 3.7: The di-photon invariant mass distribution with each event weighted by the S/(S + B) value of its category. The lines represent the fitted background and signal, and the coloured bands represent the ±1 and ±2 standard deviation uncertainties in the background estimate. The inset shows the central part of the unweighted invariant mass distribution [25].

ECAL is composed of 61200 lead tungstate ($PbWO_4$) crystals, installed in the central barrel, and 7324 crystals placed in each one of the endcaps. The ECAL Barrel (EB) covers a pseudorapidity region of $|\eta|$<1.48, which the endcaps extend to $|\eta|$<3. The motivation behind the choice of $PbWO_4$ was its radiation tolerance, its small *Radiation Length* ($X_0 = 0.89cm$) and *Moliere radius* ($r_M = 2.19cm$), and its fast response (99% of the light is collected in 100 ns, which is compatible with the 40 MHz interaction rate of the LHC). The most important drawback of using $PbWO_4$ crystals is their small light yield. In order to mitigate this problem, modern photodiodes are utilized. The layout of the calorimeter is shown in Figure 3.8.

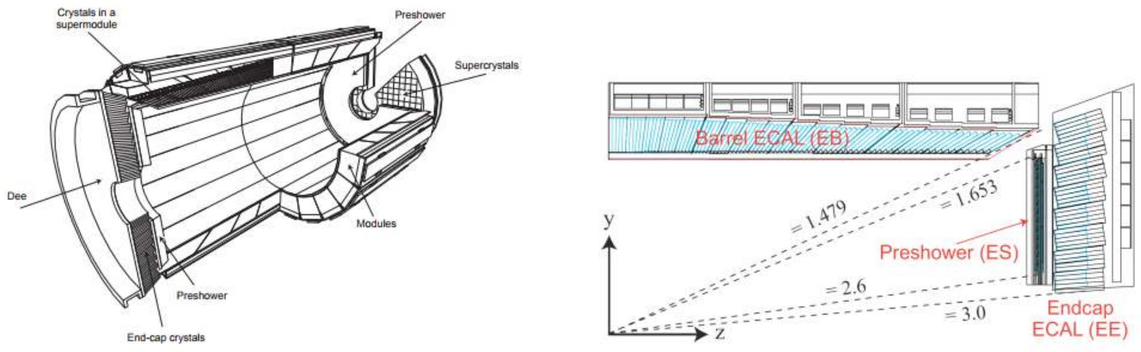


Figure 3.8: **LEFT**: Schematic layout of the CMS electromagnetic calorimeter, presenting the arrangement of barrel supermodules, endcaps and the preshower. **RIGHT**: Longitudinal cut of one quarter of the ECAL (taken from [26]).

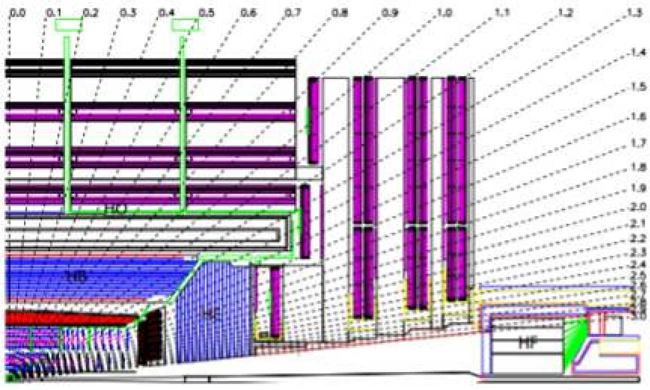
The barrel part is divided in 360 segments in ϕ and 2×85 segments in η. The crystals are mounted with a 3° tilt in both ϕ and η, with respect to the beam axis, to avoid cracks aligning with particle trajectories. The length of each crystal is 23 cm or 25.8 X_0, with a cross section of 22 mm $\times$ 22 mm for the outer face and 26 mm $\times$ 26 mm for the rear face of the crystal.

The endcap consists of identically shaped crystals, grouped in mechanical units of 5×5 crystals (Super-Crystals or SCs). Each endcap is divided into 2 halves, with 3662 crystals each. In each half the crystals and SCs are arranged to point at a focus 1300 mm beyond the Interaction Point, giving different angles, ranging from 2 to 8 degrees with respect to the z-axis. The length of each crystal is 22 cm or 24.7 X_0, with a cross section of 28.6 mm $\times$ 28.6 mm for the front face, and 30 mm $\times$ 30 mm for the rear face of the crystal ([26],[27]).

A *Preshower* detector is installed in front of the endcaps of the ECAL. The *Preshower* detector is a 20 cm thick sampling calorimeter, composed of two layers. One layer of lead radiators that initiate electromagnetic showers from incoming photons/electrons and a second layer of silicon strip sensors, placed after each radiator, to measure the deposited energy and the transverse shower profiles. The main purpose of this fine-grain photon sensitive detector is to identify and reject neutral pions in the endcaps by resolving the two close spaced photons coming from the π^0 decay. *Preshower* also helps in the identification of electrons against minimum ionizing particles and improves the position determination of electrons and photons with high granularity [28].

3.5 The Hadronic Calorimeter

The CMS Hadronic Calorimeter (HCAL) is a brass/scintillator sampling calorimeter with coverage up to $|\eta| < 3$. It is the second calorimeter layer met by a particle that emerges from the tracking system. HCAL is very important for the measurement of hadronic jets, as well as the detection of neutrinos or exotic particles that manifest in the detector as an apparent missing transverse energy. A jet is a bunch of hadrons originating from the same initial quark. The jets are then observed as a superposition of many hadron showers which also contain an electromagnetic component. HCAL is split into two parts, the barrel (HB) and the endcap (HE). Figure 3.9 illustrates an r-Z schematic drawing of a quarter of the CMS detector, showing the location of the Hadronic Calorimeter, as well as the different particle sub-detector penetration.



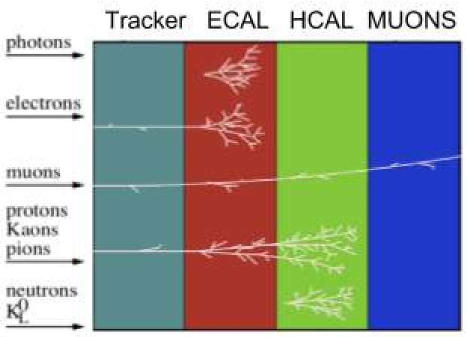


Figure 3.9: **RIGHT**: An r-Z schematic of a quarter of the CMS detector, showing the location of the HB, HE, HO, and HF calorimeters in CMS [26]. **LEFT**: Illustration of the different particle sub-detector penetration.

HCAL is radially restricted between the outer extent of the ECAL (1.77 m) and the inner extent of the magnetic coil (2.95 m). As a result, the amount of material that can be used to absorb the incident particle energy is constrained to these dimensions. This space, however, is not sufficient to completely absorb the most energetic hadrons. For this reason an outer barrel hadronic calorimeter (HO) was added, after the solenoid, using the magnet itself as absorption material. The HO consists of one layer of scintillators, placed immediately before the barrel muon system, ensuring that hadronic showers are sampled with nearly 11 hadronic interaction lengths. Similarly, in the forward regions, the endcap calorimeter is complemented by the HCAL Forward (HF) calorimeters, placed at a distance of $11.2\ m$ from the IP and outside the muon chambers. The HF extends the coverage to $|\eta|$<5, and enables better missing E_T measurements (Figure 3.9 left).

3.6 The muon system

As implied by the experiment's middle name, the detection of muons is of great importance to the Compact Muon Solenoid. For example, the decay of a Higgs boson into ZZ, which in turn decay into 4 leptons, as predicted by the Standard Model, has been characterized as "*gold plated*" for the case in which all the leptons are muons (see Figure 3.10). The muon system has three goals; identifying muons, reconstructing their track, assigning momentum and charge over the entire kinematic range of the LHC, and triggering events based on their presence.

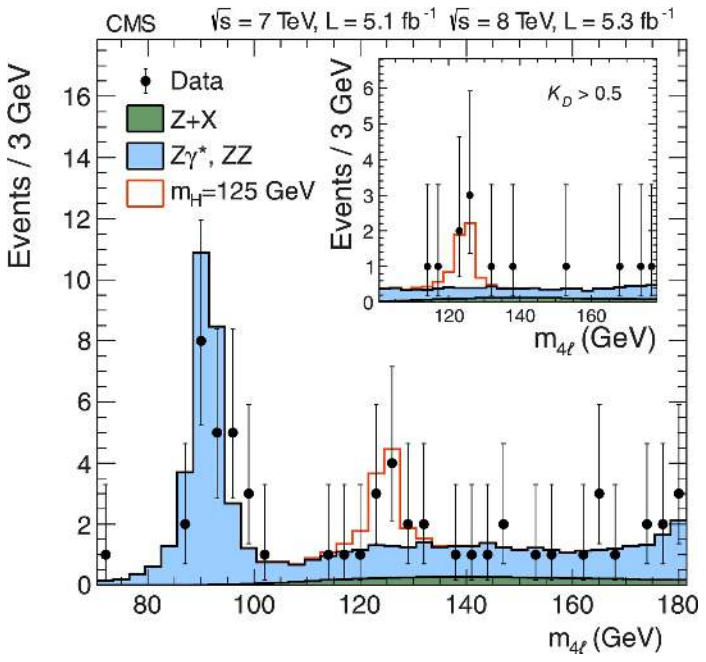
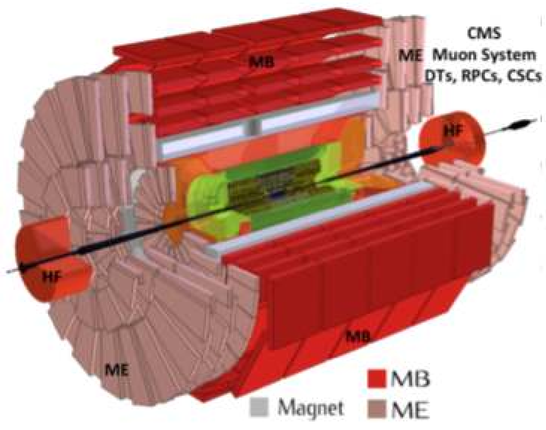


Figure 3.10: Distribution of the four-lepton invariant mass for the $ZZ \rightarrow 4\ell$ analysis. The points represent the data, the filled histograms represent the background, and the open histogram shows the signal expectation for a Higgs boson of mass , added to the background expectation. The inset shows the distribution after selection of events with , as described in the text. [29].

The muon system of CMS is embedded in the return yoke of the superconducting solenoid, and comprises the last sub-detector traversed by a generated particle. The presence of the strong magnetic field (and its flux iron return yoke) absorbs hadrons that might escape HCAL and the magnet, and provides the bending force required for good muon momentum resolution and trigger capability. It consists of one cylindrical barrel section (MB), covering the pseudorapidity region $|\eta| < 1.2$, and two planar endcap sections (ME), covering the pseudorapidity region $0.9 < |\eta| < 2.4$.

CMS uses 3 types of gaseous particle detectors for muon identification, with three different technologies [2]. Drift Tube (DT) chambers in the barrel region, Cathode Strip Chambers (CSC) in the endcap regions, and Resistive Plate Chambers (RPC) in both regions [31]. The muon system layout is shown in Figure 3.11. The lower cost and very high position resolution DTs were chosen



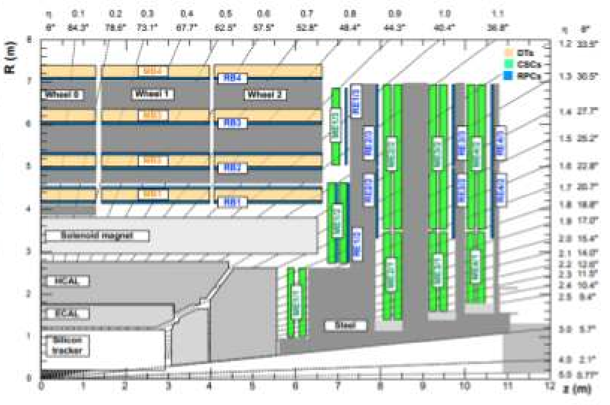


Figure 3.11: **LEFT**: 3D illustration of the CMS Barrel and Endcaps muon system. **RIGHT**: Schematic view of one quadrant of the CMS muon system in the R-z plane, in the Run-2 configuration. The RPC chambers are marked in blue, the DT chambers in yellow and the CSC chambers in green. [30].

in the barrel region, where the muon rate is low, the neutron background is relatively small and the magnetic field is mostly uniform with strength below 0.4 T in-between the yoke segments. On the downside, DTs have low timing accuracy due to drift times of the order of 400 ns. This was the main reason that led the DT system being complemented by RPCs, that have a response time of about 1 ns. In the endcap regions, due to the non-uniform magnetic field and the higher muon rates and background levels, CSCs were installed, since they have fast response time (due to their short drift path), and can be finely segmented (Figure 3.12).

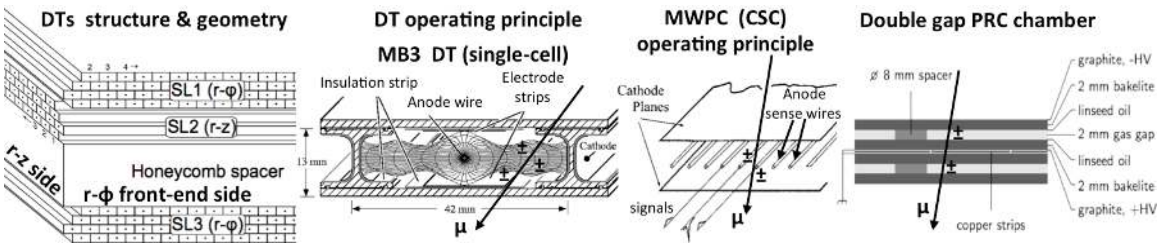


Figure 3.12: **From LEFT to RIGHT**: DTs geometry, DT individual cell, CSC chamber operating principle and RPC geometry ([19]).

3.6.1 The Drift Tube system

As previously mentioned, the use of drift chambers as tracking detectors for the barrel muon system is optimal, due to the low expected rate and the relatively low strength of the local magnetic field, providing a low-cost solution. The CMS barrel muon detector consists of 4 concentric cylinders around the beam line; the 3 inner cylinders have 60 drift chambers each, and the outer cylinder has 70. Each cylinder is segmented in 5 wheels along the z-direction and each wheel is further divided in $12 \times 30°$-sectors in ϕ. Each sector consists of four layers of stations (MB1, MB2, MB3, MB4 in increasing radius), and every station contains both DT and RPC chambers. The layout of the DT system can be seen in Figure 3.13

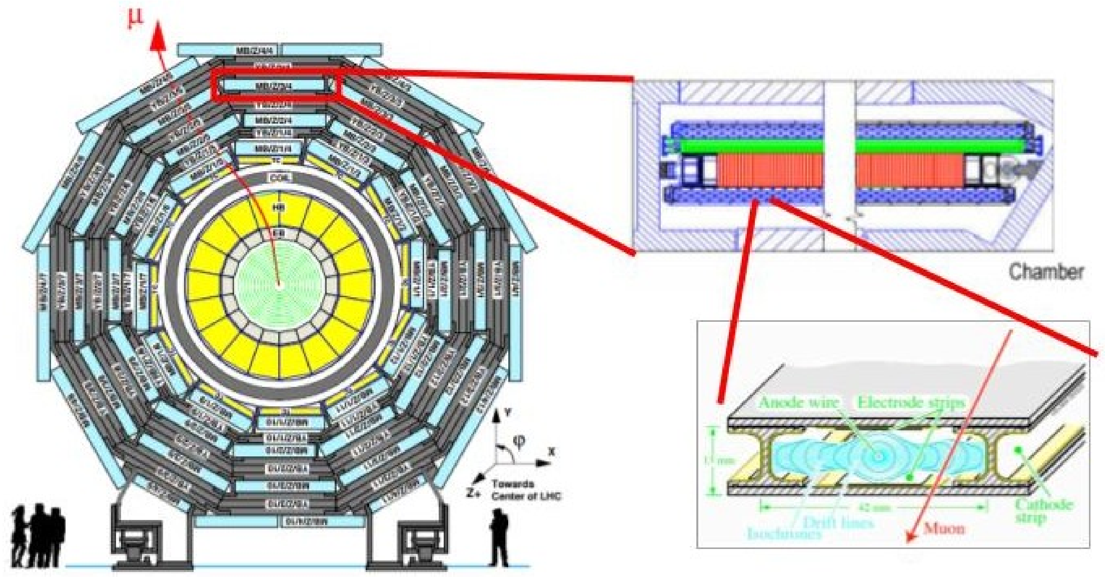


Figure 3.13: Layout of the DT system in the CMS detector. A muon station (or DT chamber) consists of three *superlayers* made of 4 layers of DT cells. A SL measuring the z-coordinate (in green) and a honeycomb plate (red) are positioned between two ϕ-measuring SLs (blue) [19].

The 3 inner drift-tube chambers are made of 3 *superlayers* (SLs), consisting of four planes of drift cells glued together. The outer DT chamber is made of only 2 SLs. The SLs are glued to the outer faces of a very light aluminium spacer, the honeycomb plate. The inner and outer superlayers of the DT stations are used to measure the azimuthal coordinate ϕ in the bending plane transverse to the accelerator beams. The central superlayer measures the pseudorapidity η. Since the fourth muon station has only ϕ-superlayers, it does not provide an η measurement. [32]

The drift cell uses the ionization electrons drift time in the active Ar/CO_2 gas, to measure the spatial position of an ionizing particle. The number of electrons can continue to multiply (avalanche multiplication), before being collected by a $2.4m$-long anode wire to which they drift due to the presence of an electric field. The time of arrival of the electron avalanche at the anode wire gives a measure of the distance of the particle track to the wire. The dimensions of the drift cells are $42mm \times 13mm$ resulting in a maximum drift path of 21 mm. The corresponding drift time is ~400 ns in the gas mixture of 85% Ar + 15% CO_2, at atmospheric pressure (Figure 3.14). Finally, the chamber resolution is approximately $100\ \mu m$ and is achieved by the 8 track points measured in the two ϕ SLs.

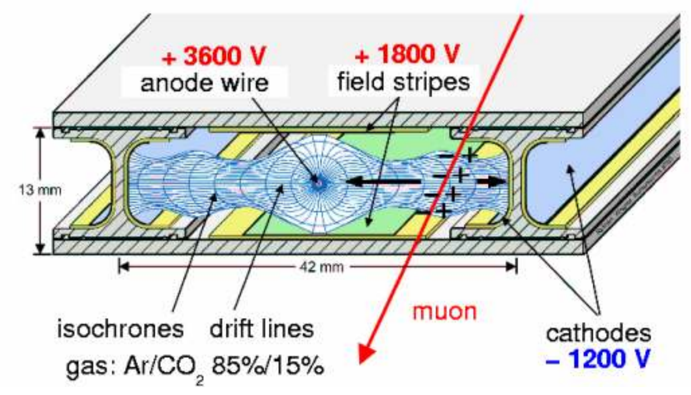


Figure 3.14: Transverse view of a baseline drift tube cell illustrating its operating principle [33].

3.6.2 The Cathode Strip Chambers

In contrast with the barrel region of the CMS detector, in the endcap regions, the particle flux is high and the magnetic field is stronger and non-uniform. The long drift time and the complicated drift path calculations make the use of drift tubes non-optimal. For these reasons, in the CMS endcaps , CSCs were used. CSCs offer fast response times that lead to short drift paths, fine segmentation and tolerance for the non-uniformity of the magnetic field. The CSCs cover the $|\eta|$ region from 0.9 to 2.4.

During Run 1, the CMS Endcap Muon system consisted of 468 Cathode Strip Chambers of six different kinds, spanning from 10° to 20° in ϕ and 1.7 m to 3.5 m in length. During the first Long Shutdown, an additional layer of 72 chambers, along with a layer of yoke disks, were added, increasing the total number of chambers to 540. Similarly to the barrel region, each endcap has 4 stations of chambers (ME1, ME2, ME3, ME4 from innermost to outermost), mounted on the faces of the endcap steel disks, perpendicular to the beam. The CSC system, as part of the muon layout, is illustrated in Figure 3.11.

The CSCs are multi-wire proportional chambers comprised of 6 anode wire planes, interleaved among 7 copper cathode panels. These panels define 6 gas gaps filled with a mixture of 40% Ar + 50% CO_2 + 10% CF_4. The anode wire planes are stretched between two copper cathodes, one continuous and the other segmented in strips, to provide position measurement. The cathode panels run lengthwise, at constant $\Delta\phi$ width, to provide a precise measurements in the r-ϕ plane (by means of interpolating charges induced on strips), while wires run azimuthially and measure the track's radial coordinate (Figure 3.15).

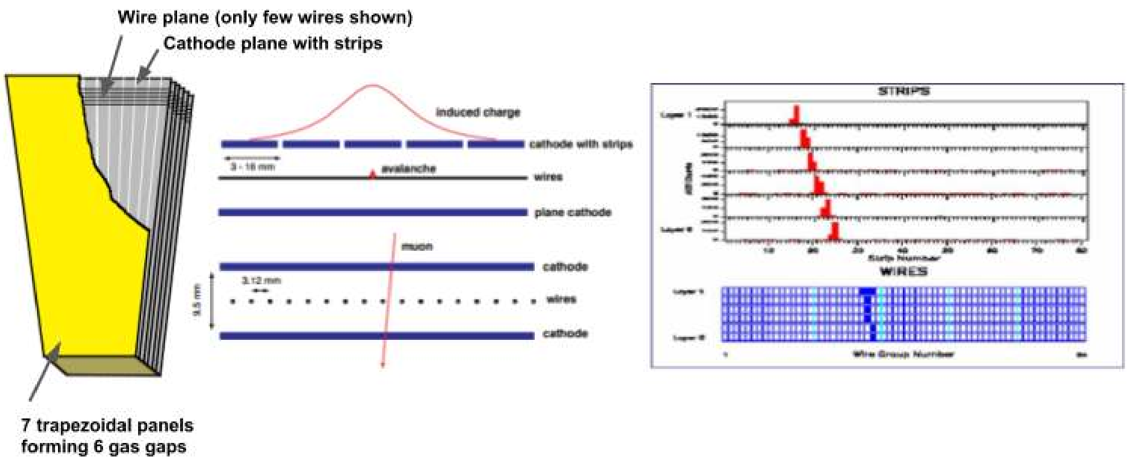


Figure 3.15: LEFT: Layout of a CSC made of 7 trapezoidal panels. The panels form 6 gas gaps with planes of sensitive anode wires. The cut-out in the top panel reveals anode wires (azimuthal direction) and cathode strips (radial direction). **MIDDLE**: A schematic view of a single gap, illustrating the principle of CSC operation [34]. **RIGHT**: A cosmic ray observed with CSC chambers during the commissioning [35].

3.6.3 The Resistive Plate Chambers

The measurement of the correct beam crossing time is crucial for the muon system. For this reason, in both barrel and endcap regions of the muon system, a complementary triggering detector system was included to provide better time resolution. The Resistive Plate Chambers cover the range of $|\eta| < 1.6$ and provide a fast, independent trigger, with a time resolution of 1 ns but with lower position resolution measurement.

Similarly to the DTs and CSCs, the RPCs are arranged in stations. In the barrel region 4 RPC stations (RB1, RB2, RB3 and RB4 from innermost to outermost) were installed in 6 layers adding to a total of 480 chambers. In the two innermost stations, two layers of RPCs were placed, to provide better measurements for the low momentum muons that will not reach the outer stations. In the endcap region, 3 layers of RPC stations (RE1, RE2 and RE3 from innermost to outermost) were installed and used during Run-1. However, during the LS1, one more outer layer (RE4) was added, to support the extra CSC (ME4) layer, giving a total of 576 chambers. The layout of the RPC system can be seen in Figure 3.11.

The RPCs are double-gap chambers that operate in avalanche mode. Each RPC gap consists of two resistive plates that define a $2mm$ gap filled with a mixture of 96% $C_2H_2F_4$, 3.5% i-C_4H_{10} and 0.5% of SF_6. The resistive plates are coated with graphite and high voltage is applied in order to have an electrical field inside the gas gaps. Between the two gas gaps, a read-out strip is placed. When a charged particle crosses the RPC, it ionizes the gas and the high electric field generates an avalanche. The induced charge is sampled by the read-out strips. In Figure 3.16, an illustration of a barrel RPC can be seen.

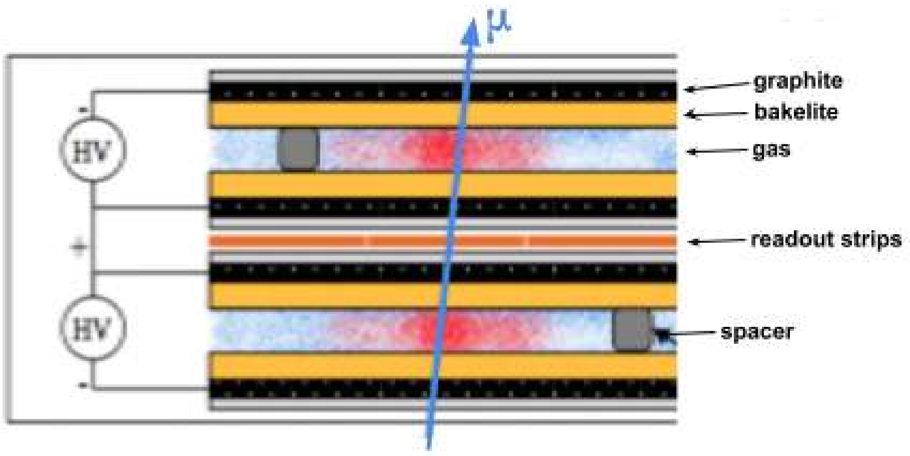


Figure 3.16: Schematic view of a generic barrel RPC, illustrating the principle of its operation. [36]

3.6.4 CMS muon system upgrades for the HL-LHC

As discussed in detail in section 2.4, the LHC will undergo a major upgrade to extend its operability and increase its luminosity (and collision rate) by a factor of five. The new conditions will require a major upgrade of the muon system. An upgrade of all systems is scheduled, as well as the installation of a new detector in the forward region.

The extension of the muon system in the forward region is required to compensate the weak B-field and the high background at HL-LHC. The forward region will be reinforced by two layers of Gas Electron Multiplier (GEM) detectors, covering the pseudorapidity region $1.6 < |\eta| < 2.4$. In addition, six extra layers of GEM detectors will be installed in the very forward region up to $|\eta| = 2.8$. Finally, improved RPC (iRPC) chambers will replace the current ones at the 2 outer stations of the endcaps, within $2.5 > |\eta| > 1.8$. The new chambers will have reduced pick-up charge, operational voltage and recovery time. The new muon system, including the GEM and iRPC detectors, can be seen in Figure 3.17.

Regarding the present muon system, the DT electronics will be replaced to cope with the higher radiation levels, and increase the trigger rate capabilities. Additionally, the frond-end electronics of the inner ring CSC chambers will be replaced to accommodate the increased occupancy and larger trigger rate [37].

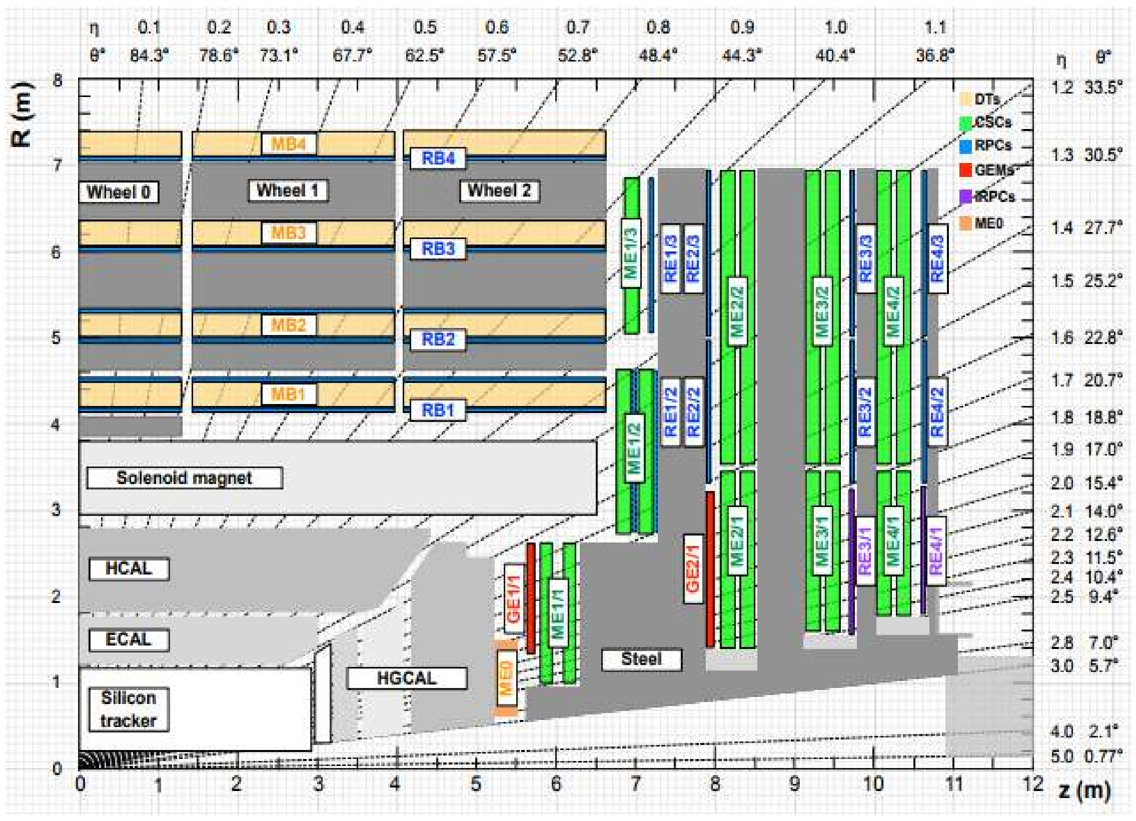


Figure 3.17: Layout of the proposed upgraded muon system in the forward region, including the new GE1/1, GE2/1, ME0 GEM detectors and the improved RPC RE3/1 and RE4/1 [36].

Chapter 4

Technology and electronics

4.1 Introduction

Observing phenomena at the subatomic level requires extraordinary instruments, particle accelerators and particle detectors. For this reason, a new field in physics was developed called Instrumentation Physics, where research and development about instruments for physical experiments are carried out. It brings together the fields of advanced electronics, detector technology and modern experimental physics, and advances the reach of scientific research. The CMS detector at CERN is one of the largest in the world, providing state-of-the-art scientific facilities to use in exploring what matter is made of, and how forces hold it together. It produces great quantities of data, whose acquisition, reduction and interpretation have made up a significant component of the experimental effort both technically and financially [38]. The progress in accelerator and detector technologies and in the associated readout electronics technologies is linked to the continuous evolution in the semiconductor industry. Such needs have led to the development and mass use of special Integrated Circuits (ICs) that excel in high speed parallel processing. The technology behind these ICs will be presented in this chapter.

4.2 Field-Programmable Gate Arrays

The FPGA acronym stands for Field-Programmable Gate Array and refers to re-programmable ICs that contain an array of Configurable Logic Blocks (CLBs). A gate array circuit is a prefabricated silicon chip circuit, with no particular function, in which transistors, standard *NAND* or *NOR* logic gates, and other active devices are placed at regular predefined positions and manufactured on a wafer, usually called master slice. The FPGAs are designed to be configured by a customer or a designer after manufacturing (in the "field") - hence the term *"field-programmable"*.

Unlike processors, FPGAs are truly parallel in nature, so different processing operations do not have to compete for the same resources. Each independent processing task is assigned to a dedicated section of the chip and can function autonomously, without any influence from other logic blocks. As a result, FPGAs can have very efficient hardware algorithms with a typical speed increase, compared to a traditional CPU, of 10 to 100 times. The FPGA chip adoption is also driven by their flexibility. Every FPGA chip is made up of a finite number of programmable CLB resources, with programmable interconnects to implement a re-configurable digital circuit and I/O blocks to allow the circuit to access the outside world. This allows the same FPGA, to be reconfigured many times, to test new functions, and to correct mistakes. Moreover, they are optimal for products where future updates are expected.

4.2.1 Contemporary FPGA Architecture

FPGA resource specifications often include the number of configurable logic blocks, the number of fixed function logic blocks such as multipliers, and the size of memory resources, like embedded block RAM. Of the many FPGA specifications, these are typically the most important, when selecting and comparing FPGAs for a particular application. An overview of a modern FPGA can be seen in Figure 4.1.

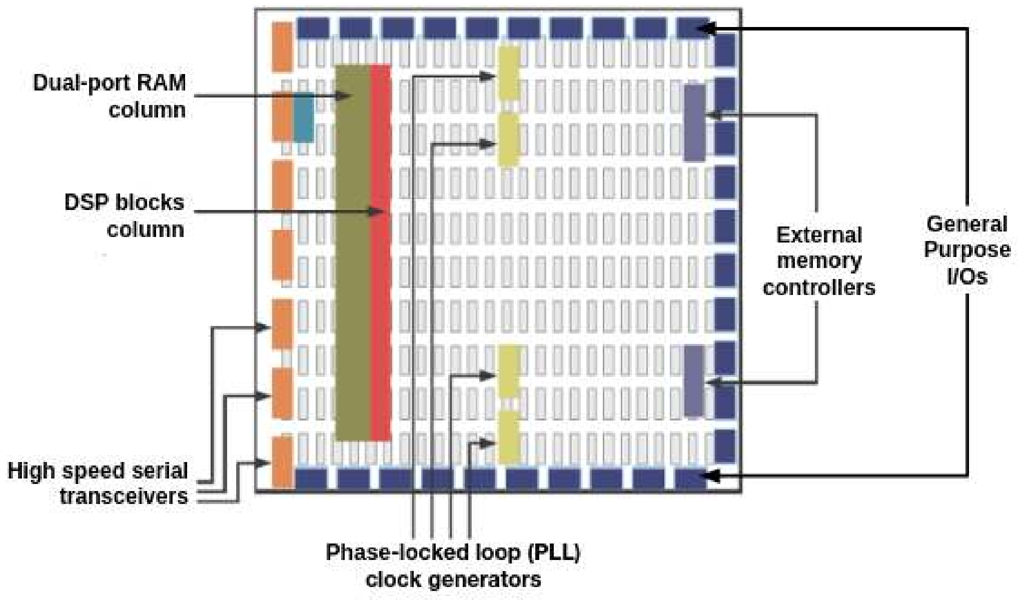

Dual-port RAM column
DSP blocks column
High speed serial transceivers
Phase-locked loop (PLL) clock generators
External memory controllers
General Purpose I/Os

Figure 4.1: A schematic of the architecture of a contemporary FPGA. The silicon of the device, known as the "fabric", is populated by CLBs, DSPs, BRAMs and DCMs, organised in columns.

The CLB is the basic logic unit of an FPGA. Sometimes referred to as *Slices* or *Logic Cells*, CLBs are made up of three basic components: *Flip-Flops* (FFs) [1], Look-Up Tables (LUTs) and *Multiplexers*

[1]Flip-flops are binary shift registers used to synchronize logic and save logical states between clock cycles within

(MUXs). On every clock edge, a flip-flop latches the 1 or 0 (TRUE or FALSE) value on its input, and holds that value constant until the next clock edge. LUTs are very small amounts of RAM, that implement much of the logic in a CLB. It is easy to assume that the number of system gates in an FPGA refers to the number of NAND gates and NOR gates in a particular chip. In reality, all combinatorial logic (ANDs, ORs, NANDs, XORs, and so on) is implemented in *truth tables*[2] within LUT memory. Finally, multiplexers are used to effectively combine LUTs, in order to permit more complex logic operations. In modern FPGAs, multiple LUTs from different CLBs can be combined, creating larger arrays of memory, called distributed RAM. Furthermore, FFs from different CLBs can be combined to create multi-bit shift registers [39]. Finally, special carry logic and dedicated fast interconnect between CLBs, boost the performance of logical functions (i.e counters) and arithmetic functions (i.e adders).

Modern applications require the use of memory, so FPGAs, in addition to the distributed RAM, now include relatively large chunks of embedded RAM, called Block Random Access Memory (BRAM). Depending on the architecture of the component, these blocks might be positioned around the periphery of the device, scattered across the face of the chip in relative isolation, or organized in columns, as shown in Figure 4.2. High-End FPGAs today contain hundreds of MBytes of BRAM.

FPGAs are very efficient for Digital Signal Processing (DSP) applications, because they can implement custom, fully parallel algorithms. DSP applications, however, use many binary multipliers and accumulators, that are best implemented in dedicated DSP resources. For this reason, devices today, include thousands of dedicated low-power DSP slices, combining high speed with small size, while retaining system design flexibility.

Finally, one of the most important aspects, when designing an FPGA, is driving all of its synchronous elements by a clock signal. All the synchronous elements of the device must receive their clock signal as close together as possible, in order to avoid high timing skew[3]. For this reason, clock signals are distributed through dedicated fast tracks, creating a structure called ”clock tree”. Furthermore, special blocks called Digital Clock Managers (DCMs) are driven by the external clock pins, and provide a number of secondary clocks.

4.2.2 Multi-Gigabit Transceivers

Due to their ability to process in parallel large amounts of data, FPGAs are used in applications where high-bandwidth is paramount. For example, the use of devices with very high data transfer capabilities is required by the trigger and read-out system of the CMS experiment (see chapter 5).

The traditional way to move large amounts of data between devices was to use a bus, a

an FPGA circuit.

[2]A truth table is a predefined list of outputs for every combination of inputs.

[3]Timing skew is a phenomenon in synchronous digital circuit systems in which the same sourced clock signal arrives at different components at different times.

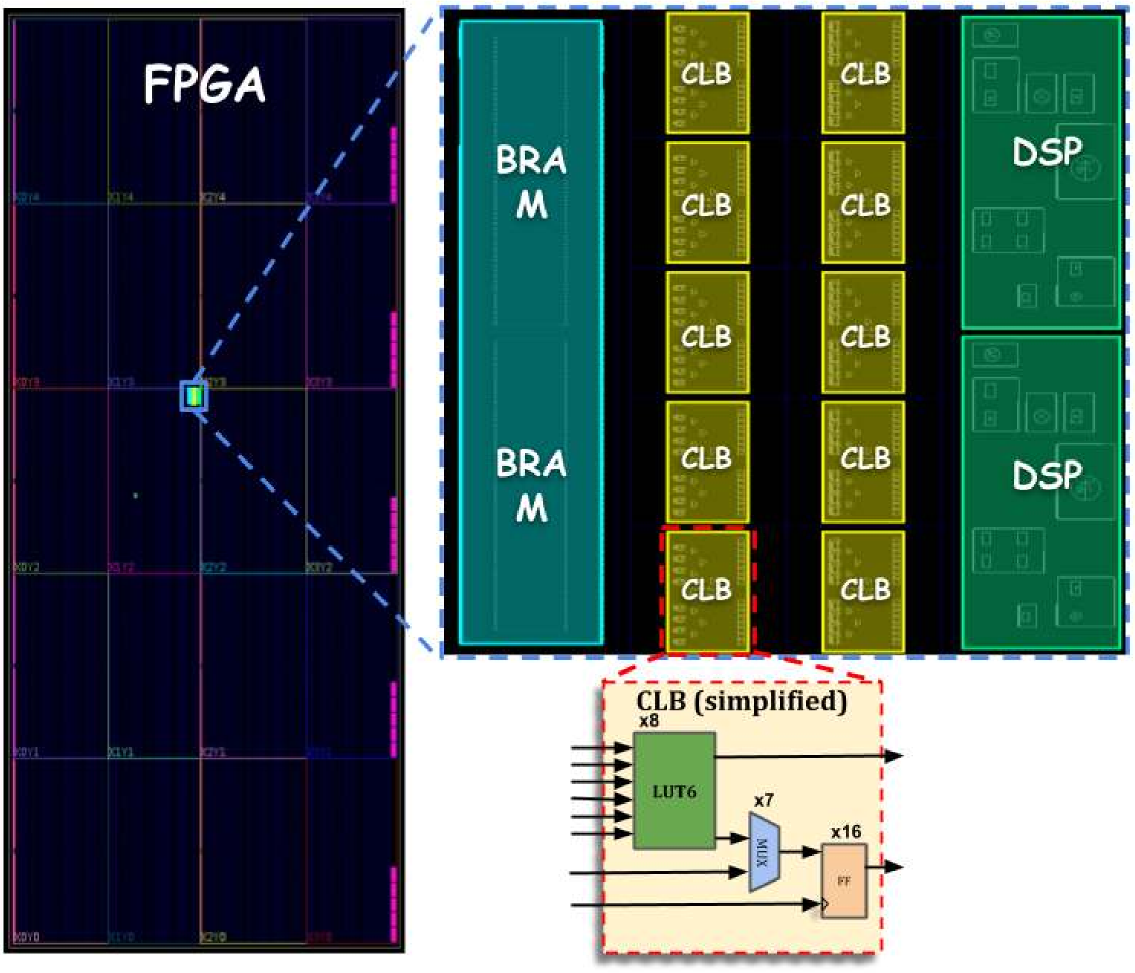


Figure 4.2: The architecture of a modern FPGA (Xilinx XKCU040) with all its main components.

collection of signals that carry similar data and perform a common function. The problem was that this required the use of large numbers of pins on the device, making PCB routing and signal integrity very difficult. For this reason, today's FPGAs include special hard-wired blocks that convert data between serial and parallel interfaces in each direction (SerDes). A Multi-Gigabit Transceiver (MGT) is a serializer/deserializer capable of operating at serial bit rates above 1 Gbps. MGTs are used increasingly for data communications, because they can run over longer distances, use fewer wires, and thus have lower cost than parallel interfaces with equivalent data throughput. Signal integrity is critical for MGTs due to their high line rates. Apart from the serialization and deserialization, MGTs incorporate a number of technologies, like encoders/decoders, error detection, alignment, clock data recovery, signal equalization and others. The quality of a given high-speed link is characterized by the Bit Error Ratio (BER)[4].

The MGT blocks are usually divided in two functional groups, the Physical Coding Sublayer

[4]BER is the ratio of bits received in error to total bits received.

(PCS) and the Physical Medium Attachment (PMA) sublayers. The PCS layer is responsible for data encoding and decoding (i.e 8b/10b or 64b/66b encoding), scrambling and de-scrambling, alignment marker insertion and removal, block and symbol redistribution, and lane block synchronization and de-skew. The PMA layer transmits/receives symbol streams to/from the PCS, onto to the serial differential signal interface. Furthermore, it provides clock recovery, link monitoring and signal equalization and termination functions.

Most of the research and firmware development presented in this thesis utilize the Xilinx Ultrascale FPGAs and their state-of-the-art integrated GTH and GTY MGT transceiver blocks. The GTH/Y transceivers are highly configurable and tightly integrated with the programmable logic resources of the FPGA. They support line rates from 0.5 Gbps to 30.5 Gbps in UltraScale FPGAs and 32.75 Gbps in UltraScale+ FPGAs. A block diagram of the transmitter and receiver of a GTY transceiver can be seen in Figure 4.3.

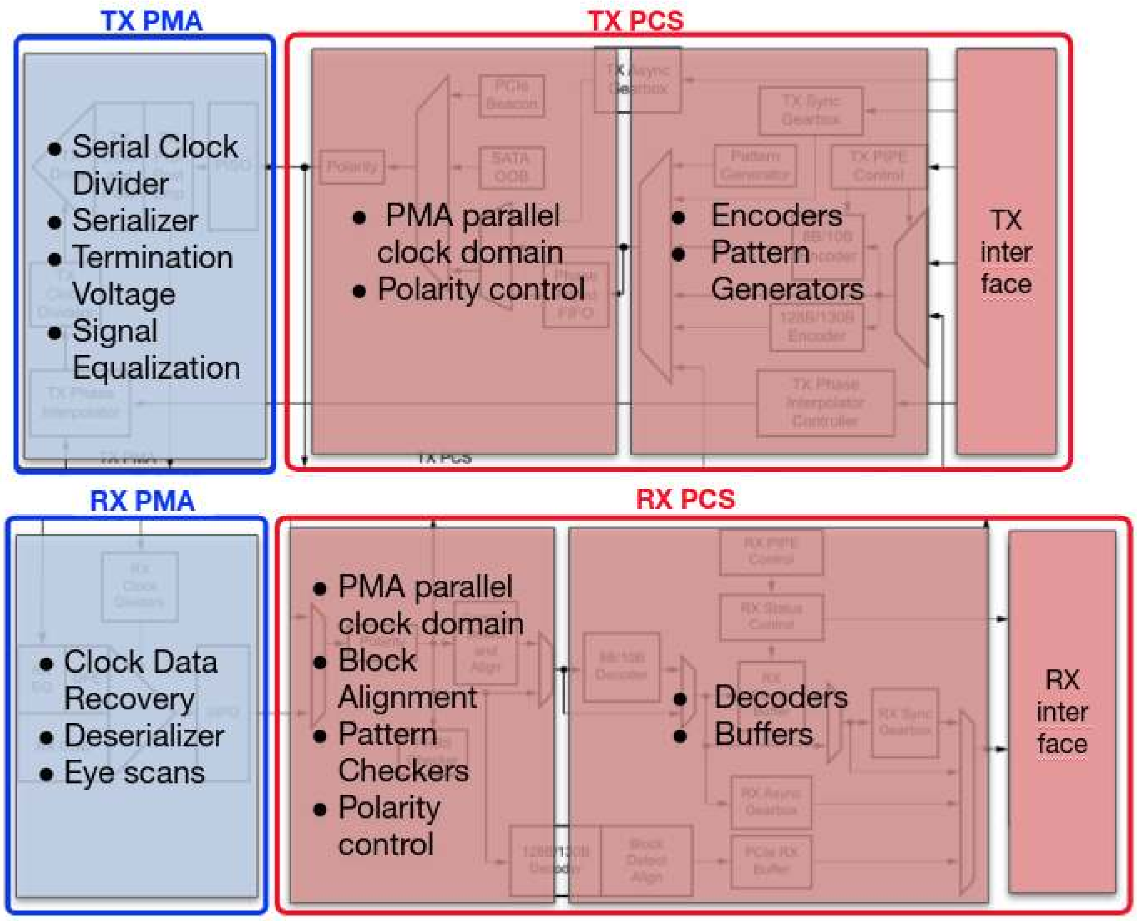


Figure 4.3: A block diagram of the transmitter and receiver of a GTY transceiver with their two sublayers, the PCS and the PMA.

4.2.3 FPGAs compared to other Programmable Logic technologies

CPLD vs FPGA : CPLD stands for Complex Programmable Logic Device and is typically comprised of less than one thousand function blocks (or logic blocks), which are accessible by a single interconnect. CPLDs start working as soon as they are powered up, and remain programmed, retaining their circuit after powering down. They have very low idle power consumption, they are more "secure" due to design storage within built in non-volatile memory, and might be cheaper for implementing simple circuits. On the other hand, FPGAs are huge compared to CPLDs (millions of more complex CLBs), and are much more capable, but they are more expensive. When a design requires simple *glue-logic* or similar functionality which doesn't need to be changed much, or when you need an instant-on circuit, then CPLDs are the best option. Otherwise, for most other applications, FPGAs are generally preferred. Sometimes, you can find both CPLDs and FPGAs in a design. In those designs, CPLDs generally do simple *glue-logic* as mentioned before, and are responsible for "booting" the FPGA, as well as controlling the reset and boot sequences of a complete system board. The complexity of the application is the decisive factor here.

ASIC vs FPGA: Application Specific Integrated Circuits are specific chips (as the name suggests), used to implement both analog and digital functionalities, in high volume or high performance. ASICs are full custom, therefore they require higher non-recurring, one time, development cost in order to design and implement. Unlike the FPGAs, however, they are not re-programmable and, therefore, a change requires a recurring high cost development. On the positive side, ASICs are much denser, and one can integrate several different functionalities into one chip, offering a small sized, low power solution. Furthermore, the cost per chip, after the development stage, is relatively cheap giving an advantage to ASICs over FPGAs, for mass production applications. In the past, FPGAs used to be selected for lower speed/complexity/volume designs, but today's FPGAs easily push the 500 MHz performance barrier. With an unprecedented logic density increase and the incorporation of other features, such as embedded processors, DSP blocks, clocking, and high-speed serial links at lower price points, FPGAs are nowadays a compelling proposition for almost any type of design. FPGA design flow, also eliminates the complex and time-consuming floorplanning, place and route, timing analysis, and mask/re-spin stages of the project, since the design logic is already synthesized to be placed onto an already verified, characterized FPGA device. When needed, Xilinx provides the advanced floorplanning, hierarchical design, and timing tools to allow users to maximize performance for the most demanding designs. FPGA and ASIC devices provide different features to designers, and they must be carefully evaluated before choosing one over the other.

Microcontroller vs FPGA : The structure of a *Microcontroller* is comparable to a simple computer placed on a single chip with all of the necessary components, like memory and timers, embedded. It is programmed to perform limited tasks, because it comes with its circuitry and instructions. A programmer has to abide by the restrictions while developing code. The very basic nature of FPGAs allows them to be more flexible than most microcontrollers. They are "field programmable", and can be reprogram to perform any logic task that can be accommodated within the available logic gates. The logic gates can be rewired as many times as required, to change the program and carry out a different task. Furthermore, you can create a microcontroller architecture using an FPGA, but the reverse is not possible. The flexibility of FPGAs comes at a price because they consume more power than typical microcontrollers, making them unsuitable for applications, where power drain is an issue. With microcontrollers, you can buy packages that are geared towards a certain task, and just program them to your exact specification relatively quickly. Microcontrollers read through each line of the program at a time, sequentially, which means the commands are processed in sequence. FPGAs can process commands concurrently and can execute numerous lines of codes in parallel. In conclusion, FPGAs are more suitable for more complex functions, were fast parallel processing is required, while microcontrolers are more suitable for simpler control functions.

4.3 Hardware Description Languages

Design languages provide the means to describe the operation of both software programs and hardware. Over the years, a large number of languages have been developed. Some are still in use today, while others have become obsolete. Digital Design languages are of two general types; Software Programming Language (SPL) and Hardware Description Language (HDL). At one time, designers were either software or hardware designers, and design teams were clearly distinguished by these separate roles. Today, however, designers are involved in both software and hardware design and need skills in both areas, although they may be specialized.

Software Programming Languages allow a software designer to create executable software applications, that will operate on a suitable processor. The target processor can be one of three types: microprocessor (μP), microcontroller (μC), or Digital Signal Processing. Essentially, software is developed as a software application to run on a workstation or PC, executing on a suitable operating system (Linux, Mac OS or Windows), or as embedded software to run on a processor within an embedded system. Some of the most used SPLs are C, C++, Python, Visual Basic, JAVA and others.

4.3.1 Overview

The Hardware Description Language design is based on the creation and use of textural based descriptions of a digital logic circuit or system. Hardware circuits or system designs created using HDL are generated at different levels of abstraction. At the highest level, the system idea or concept is the initial high-level description of the design that provides the design specification. At the next level, the algorithm describes the behavior of the design in mathematical terms. The actual implementation of the design is not yet described by neither the system idea or the algorithm. The algorithm structure in hardware is described by the architecture that is coded in an HDL language. The architecture level identifies the high-level functional blocks to use, and how the functions are connected. The algorithm and architecture levels describe the behavior of the design to be verified in simulation. The next level down from the architecture is the Register Transfer Level (RTL). The RTL describes the storage (in registers) and flow of data around a design, along with logical operations performed on the data. This level is usually used by synthesis tools that describe the design structure (the netlist of the design, in terms of logic gates and interconnect wiring between the logic gates). The logic gates are themselves implemented using transistors.

A HDL is highly similar to a software programming language, but there are major differences. Most programming languages are inherently procedural, with limited syntactical and semantic support to handle concurrency. HDLs, on the other hand, resemble concurrent programming languages in their ability to model multiple parallel processes, such as flip-flops and adders, that automatically execute independently of one another. Any change to the process's input automatically triggers an update in the simulator's process stack. Both programming languages and HDLs are processed by a compiler (often called a synthesizer in the HDL case), but with different goals. For HDLs, "compiling" refers to logic synthesis, meaning the process of transforming the HDL code listing into a physically realizable gate netlist. The netlist output can be a "simulation" netlist with gate-delay information, a "hand-off" netlist for post-synthesis placement and routing on a semiconductor die, or a generic industry-standard Electronic Design Interchange Format (EDIF). On the other hand, a software compiler converts the source-code listing into a microprocessor-specific object code for execution on the target microprocessor. [40].

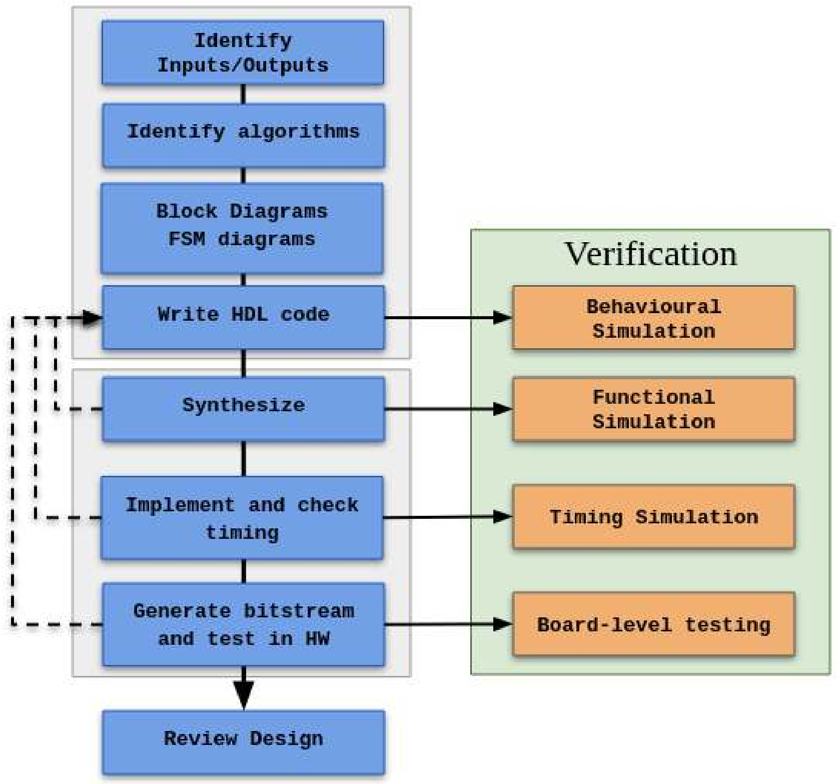


Figure 4.4: The typical FPGA Design Flow from HDL to physical implementation.

The first Hardware Description Languages appeared in the 1970s, but became popular in the 1980s, following the widespread adoption of new technologies like the Very Large Scale Integration

(VLSI)[5] and the appearance of the first Programmable Logic Devicess (PLDs). The first, and one of the two most popular, modern HDL, Verilog, was introduced in 1985 while a second, Very-high-speed-integrated-circuit Hardware Description Language (VHDL), was introduced in 1987. Those two languages emerged as the dominant HDLs in the electronics industry, while older and less capable HDLs gradually disappeared. All the development and coding presented in this thesis was done using VHDL, thus a brief description of the language will be presented in the next section.

4.3.2 VHDL

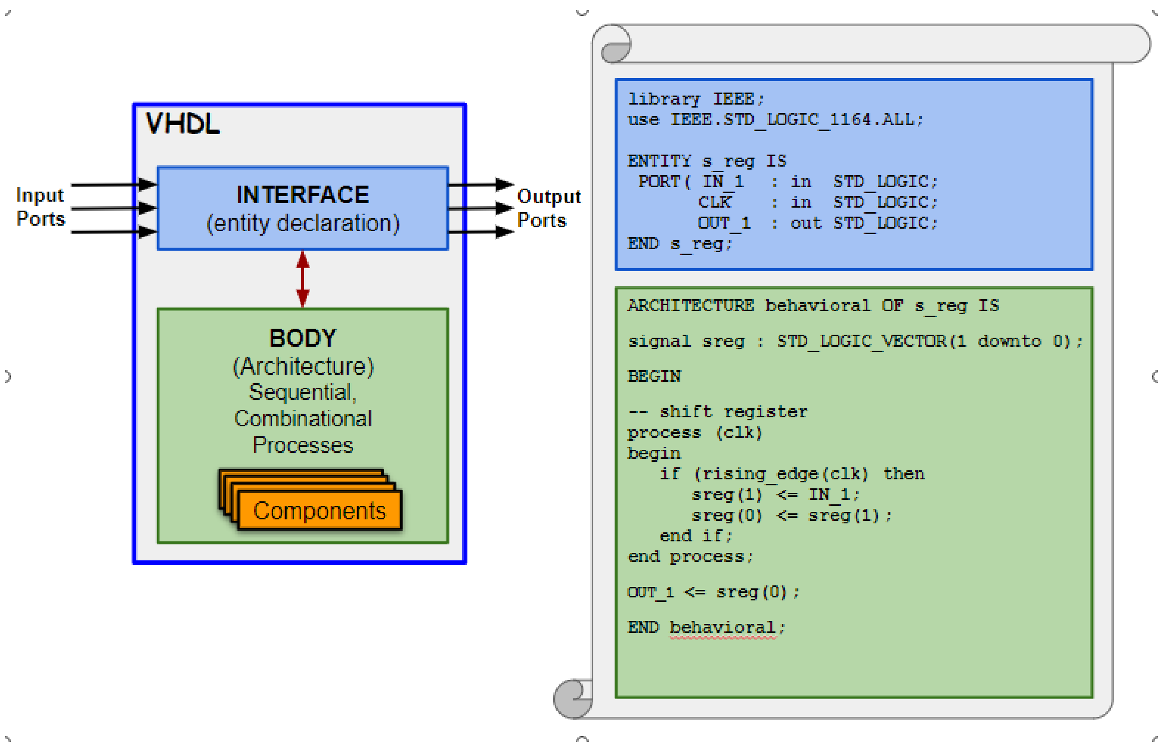
In 1983, the US Department of Defense (DoD), motivated by a lack of adequate documentation for the functionality of ASICs that were being supplied, sponsored a program to create a means to document the behavior of digital systems, that could be used across all of its suppliers. The new tool was implemented in a format similar to a programming language and included the ability to simulate the behavior of a digital system. In 1985, the first version of this tool, called VHDL, was released. In 1987, IEEE released the first industry standard version, titled "IEEE 1076-1987". The initial version went through a major revision of the standard in 1993 ("IEEE 1076-1993") and in 2008 ("IEEE 1076-2008").

A design is described in VHDL using the concept of a design *Entity*. A design entity is split into the entity declaration representing the external interface (inputs, outputs), and the *Architecture* body representing its internal structure or behaviour. The hierarchy of the design is defined by *Components*, the analogous to chip sockets. The *Components* are *instantiated* within an architecture, to represent a lower level hierarchical block. All the *Components* instantiated in the architecture are connected together using *Signals*, that represent an electrical connection, either a wire or a bus. Every *Component* has pins that are represented by *Ports*, and a *Port Map* is used to connect the *Ports* of a *Component Instance* to *Signals*. Every signal has a *Type* that defines both a set of values and a set of operations that can be performed on those values. In Figure 4.5 you can see a simple example of how to describe a shift register using VHDL.

The behaviour of an electronic circuit is described using *Processes*. All *Processes* are executed concurrently with respect to all other processes. The statements inside the process, however, are executed sequentially similar to the statements of a software programming language. A *Process* can be also be described using named *Functions* or *Procedures*. Common definitions to several *Entities*, *Architectures* or even designs can be stored in *Packages*. A *Package* can contain common *Types, Functions* or *Procedures*.

The VHDL code is typically typed in a text file. The text file is then compiled by a VHDL compiler, that builds all files necessary for simulation and synthesis of the design. The compiler first performs *Analysis*, during which it parses the VHDL source code for errors and then *Elaboration*, during which it links all the *Entities* and *Architectures* of the hierarchy [41]. To test a VHDL model, we use an enclosing model called *test bench*. The VHDL test bench file consists of an architecture body that contains an instance of the component to be tested (named Unit Under Test

[5]The process of creating an integrated circuit (IC) by combining millions of transistors or devices into a single chip



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

ENTITY s_reg IS
 PORT( IN_1   : in  STD_LOGIC;
       CLK    : in  STD_LOGIC;
       OUT_1  : out STD_LOGIC;
END s_reg;

ARCHITECTURE behavioral OF s_reg IS

signal sreg : STD_LOGIC_VECTOR(1 downto 0);

BEGIN

-- shift register
process (clk)
begin
   if (rising_edge(clk) then
      sreg(1) <= IN_1;
      sreg(0) <= sreg(1);
   end if;
end process;

OUT_1 <= sreg(0);

END behavioral;
```

Figure 4.5: A simple hardware design of a shift register as described in VHDL. The *Entity* defines the interface to the block of hardware while the *Architecture* defines its internal structure. Multiple entities can be instantiated inside an architecture representing a lower level hierarchical block.

or UUT), and processes that generate a sequence of values on signals connected to the UUT. The body can also contain processes that test the outputs of the UUT. Finally, we can use a simulator to observe the output and verify that the UUT operates correctly [42].

4.3.3 High Level Synthesis

Today, the electronic system design community is mainly concerned with defining efficient System-on-a-Chip (SoC) design methodologies, and manage the increasing algorithmic complexity of applications. Rapid prototyping is considered as a key to speed up the system design. However, hardware design is not flexible and time consuming, and consequently incompatible with both time-to-market constraint and efficient design space exploration.

It has been observed that many of the functions used in a hardware design are well known and have already been implemented. Current design trends give priority to the creation of reusable blocks of code, the so-called "Intellectual Property" (IP). The system design flow can be dramatically accelerated by re-using these blocks, instead of re-designing them from scratch. During the last 5 years, a new IP design process is gaining in popularity. The process is called High Level Synthesis

(HLS), and its main approach is to take an algorithm described in a high level software language (most commonly C, C++ , SystemC and Python) and create an HDL code. The development and validation of the functional correctness of the design for algorithms at the C-level consumes much less development time. Bridging hardware and software domains allows hardware designers to work at a higher level of abstraction, and software developers to accelerate the computationally intensive parts of their algorithms on a new compilation target, the FPGA.

The process, however, is not as simple as just compiling code for hardware. The algorithms must be written in a particular style that will enable the synthesis tools to identify and exploit parallelism. Skipping this code restructuring, the HLS tools will still derive a hardware realisation, but the resulting hardware will suffer from poor performance [43]. For more details on best practises when describing hardware using the HLS methodology, both Xilinx (*Vivado HLS Optimization Methodology Guide* [3]) and Intel (*Intel HLS Compiler:Best Practices Guide* [44]) provide detailed guides. The tools, developed by the major FPGA design companies have recently become very efficient, claiming to achieve 5-10 times faster development time with a performance penalty of 10% [45].

A High-level synthesis includes the following phases (see Figure 4.6):

- **Scheduling**: determines which operations occur during each clock cycle, based on the clock frequency, the time it takes for the operation to complete, as defined by the target device and user-specified optimization directives.
- **Binding**: determines which hardware resource implements each scheduled operation. To implement the optimal solution, high-level synthesis uses information about the target device.
- **Control logic extraction**: extracts the control logic to create a finite state machine (FSM) that sequences the operations in the RTL design.

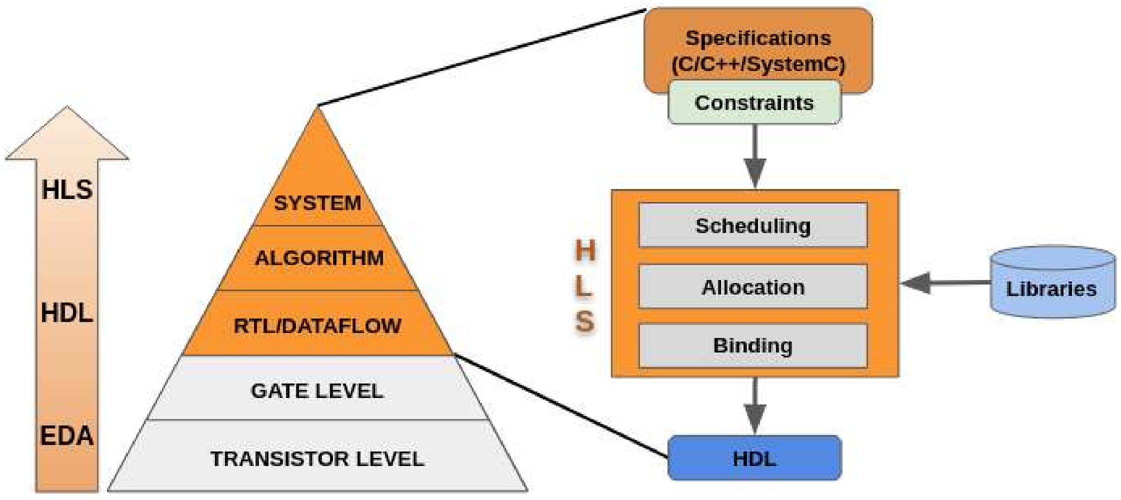


Figure 4.6: The typical flow of a High Level Synthesis design.

Chapter 5

The CMS Trigger System

The LHC provides proton-proton and heavy ion collisions every 25 ns, resulting in an interaction frequency of 40 MHz. At the nominal LHC luminosity of $10^{34}\ cm^{-2}s^{-1}$, approximately 20 - 25 simultaneous proton-proton collisions (pile-up) occur at each crossing of the proton bunches. During Run 2 (2015-2018), the maximum instantaneous luminosity achieved was $2 \times 10^{34}\ cm^{-2}s^{-1}$, with a pile-up of $\sim$50. This number of events produces an enormous raw data rate that exceeds 40 PByte per second. This data throughput is impossible to store, and must be reduced to a rate of $\sim$100 MByte per second before storage. The Trigger system of the CMS is responsible of reducing this rate by at least a factor of 10^6, through a physics event selection process. The rate reduction is performed in two steps; the first step is called Level-1 Trigger (L1T) and the second High Level Trigger (HLT). The L1T is a high-bandwidth, fixed latency system, based on FPGAs that use data from the calorimeters and the muon system. The HLT is a multi-stage iterative algorithm, running on a computer farm of fast commercial processors, and has access to all CMS sub-detectors. The maximum L1T output rate is 100 kHz, while the HLT further reduces the rate to a maximum of 1 kHz (Figure 5.1).

The trigger system was upgraded during the Long Shutdown 1 period with all electronics

5.1 The Level-1 Trigger

The Level-1 Trigger is the first stage of the on-line (during data production) event selection, and its purpose is to reduce the detector rate from 40 MHz to 100 kHz. This is achieved by discarding events that lie within our current understanding. Simultaneously, it applies a threshold for acceptance, sufficiently low, to allow all possible signatures of new physics to be recorded. Due to the enormous incoming data, and the requirement of a very quick selection decision, the system could not be based on iterative algorithms. The use of FPGAs and Application Specific Integrated Circuits (ASICs), allow for a high-bandwidth, fixed latency system, based on pipe-lined,

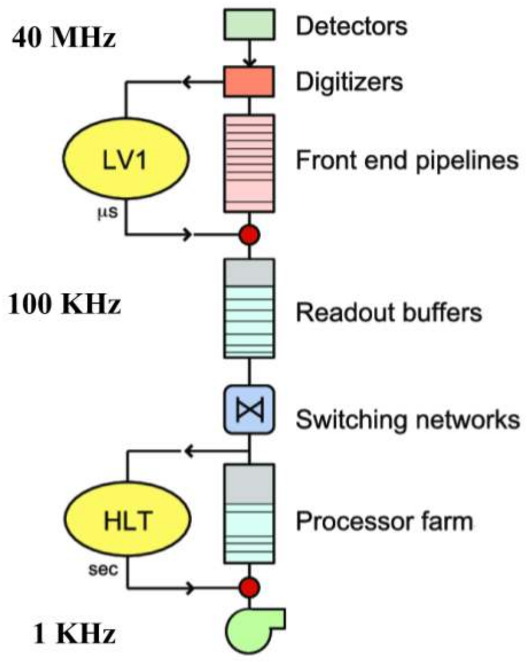


Figure 5.1: The CMS Level-1 trigger receives data from the calorimeter and the muon detectors and produces a yes/no decision, after fixed number of crossings (latency). The data are stored in pipeline memories until the Level-1 decision arrives. The High Level Trigger receives and processes complete events.

parallelized algorithms. The L1T only uses data from the calorimeters, and tracks from the muon system, since the tracker data size is too large to allow read-out on every bunch crossing. This sheer size of tracker data and the fact that they must be stored in pipeline memories of depth equal to the trigger latency, limits the trigger decision time to 160 bunch crossings (4 μs, if you consider 25 ns per bunch crossing). When the L1T generates an accept signal (L1A), all the event data are moved from the pipelined memories to a buffer, for read out and processing by the HLT.

Figure 5.2 illustrates the architecture of the L1T system during Run 1. It has local, regional and global components. The Trigger Primitive Generators (TPGs) build local primitives, based on information from the calorimeter and the muon chambers. Consequently, the Regional Triggers, combine the primitives information, and assign transverse momentum and spatial coordinates to the trigger objects. The objects are sorted by quality before they are sent to the global triggers. The Global Muon and Global Calorimeter components choose the best objects and send them to the Global Trigger. The latter, produces the final L1A signal and initiates the event read out process [46, 47].

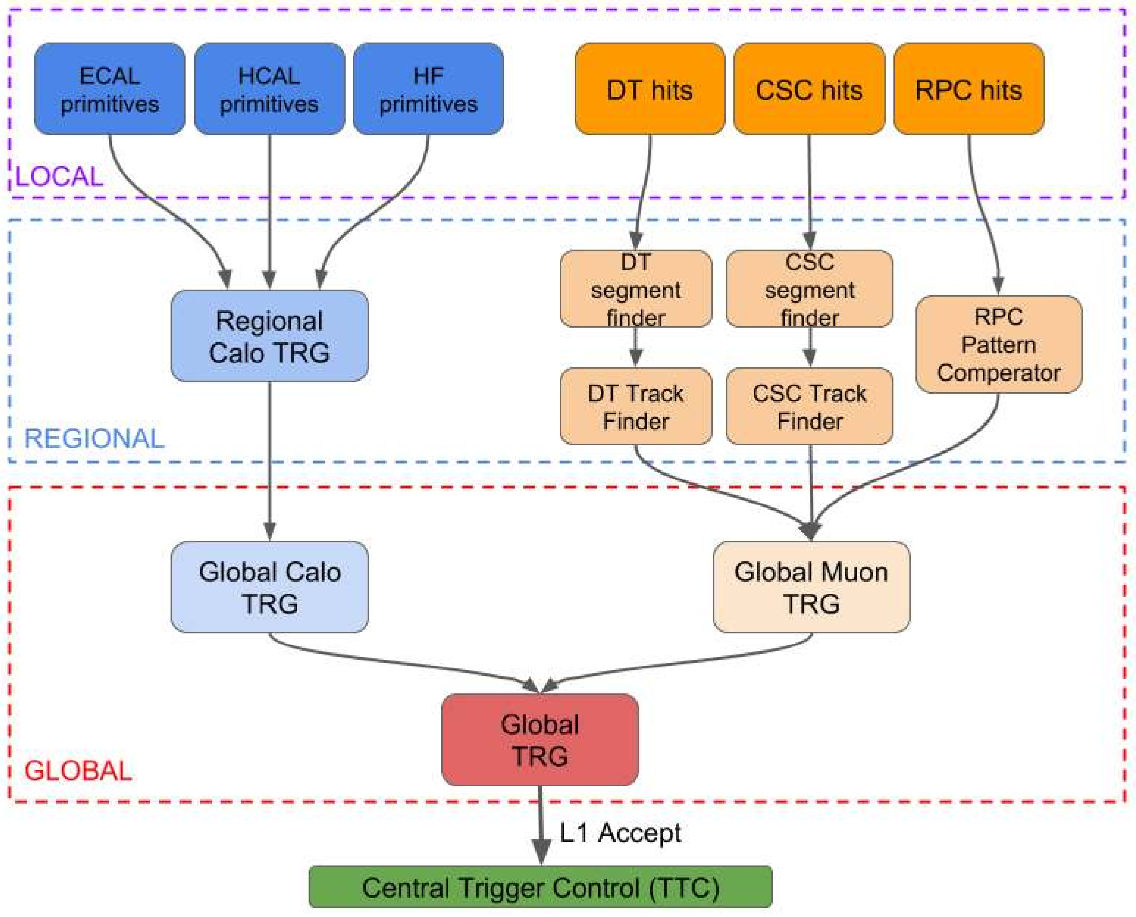


Figure 5.2: The CMS Level-1 Trigger architecture during Run 1. The muon and calorimeter trigger first build local primitives. The primitives are then sent to the regional trigger, that assigns transverse momentum, quality and spatial coordinates to the trigger objects. Finally, the global components send the best objects to the Global Muon Trigger, which produces the Level-1 Accept (L1A) signal.

5.2 The Level-1 Trigger upgrade for Run 2

The L1T system design was optimized for the LHC Run 1 conditions, at a nominal luminosity of $1 \times 10^{-34}\ cm^{-2}s^{-1}$ and an average pile-up of about 25 interactions per bunch crossing. However, after the Long Shutdown 1, the conditions changed and during the second period of operation (Run 2) which started in 2015, the LHC, reached a peak instantaneous luminosity of approximately $2 \times 10^{-34}\ cm^{-2}s^{-1}$ and an average pile-up of about 55. Under these new conditions, the online event selection became very challenging for the legacy trigger system to handle. The trigger rate would explode and would require a substantial increase of the trigger thresholds, in order to sustain a rate below 100 kHz. That would affect dramatically the number of trigger events available for analysis. In order to maintain the Run 2 low thresholds, the L1T system went through a major upgrade of its architecture, hardware and trigger algorithms. The trigger system used during Run

1 will be addressed as the legacy system, while the upgraded as current, or Phase-1 trigger system. Moreover, the current trigger system is scheduled to be upgraded for the Phase-2 HL-LHC (see Figure 5.3).

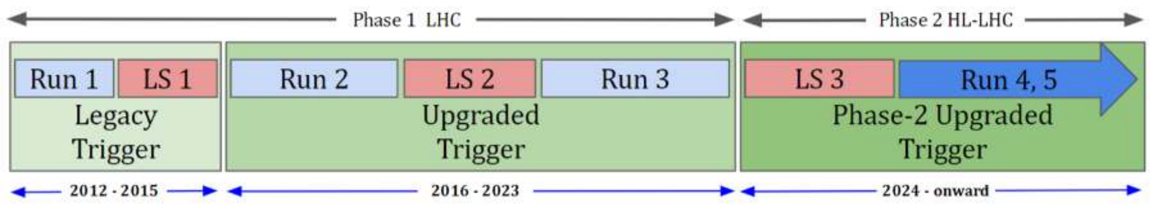


Figure 5.3: The CMS trigger time schedule. The data taking runs (cyan), Year-End Long Shutdowns (pink), and LHC phases and dates are indicated.

The most important features of the upgrade were the following:

- Improved muon p_T assignment, to avoid lower momentum muons to be mis-assigned as high p_T muons.
- Improved e/γ isolation algorithm.
- Improved τ identification algorithm.
- Pile-Up subtraction, on an event-by-event basis, based on estimations of the pile up energy density.
- Improved Global Trigger algorithms, including invariant mass calculations and muon *b-tagging* jets.

5.2.1 Architecture and Hardware upgrades

A major architecture and electronics upgrade was realized in order to support the algorithmic improvements of the L1T. The upgrade offered a more flexible system, adapted to the new conditions, and incorporated new triggering techniques as they arouse. This increased flexibility was accomplished by using high-bandwidth optical links for most of the data communication, between trigger cards, and by using modern, large FPGAs and large memory resources for the trigger logic. An overview of the upgraded trigger system can be seen in Figure 5.4.

The key technology upgrade was the replacement of the VME-based infrastructure with the Micro Telecommunications Computing Architecture (uTCA). uTCA is a modular, open standard, based on hot-swapped, plug-able processing cards, following the Advanced Mezzanine Card (AMC) standard. This standard supports several communication protocols, such as GbE, SATA/SAS, PCIe, and offers significantly more back-plane bandwidth. The uTCA crates used at CMS can host dual uTCA Carrier Hubs (MCHs), Power Modules (PMs) and Cooling Units (CUs),

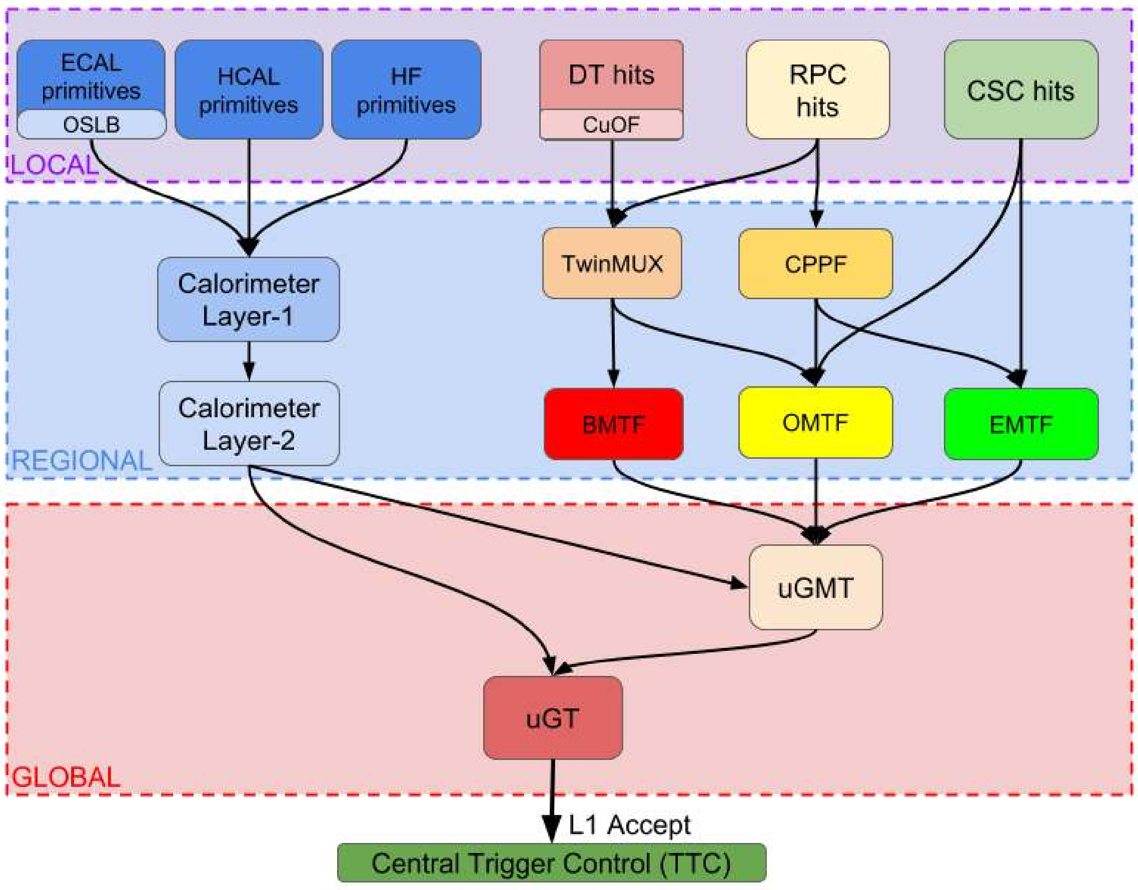


Figure 5.4: The CMS Level-1 trigger architecture after Long Shutdown 1. For the calorimeter, a 2-layer Time Multiplexed Trigger was selected, with the first layer pre-processing and multiplexing the data and the second layer processing the full event information. The muon trigger uses a more traditional trigger, based on regional reconstruction of all tracks of a given geometrical region to be built.

allowing full redundancy for systems that require it. The second MCH slot is used to distribute the LHC clock to all L1T subsystem boards [48].

The CMS trigger groups developed three varieties of AMC trigger processing cards for the upgrade, each one optimized for a different task. One type of board was used for data acquisition and clock distribution, and two different board types for data concentration. All AMC boards developed by the CMS research teams for Run 2 are described below.

1. **The Calorimeter Trigger Processor 7 (CTP7) board**, designed by the University of Wisconsin. It is the baseline hardware for the layer-1 of the new calorimeter trigger. It is based on the Xilinx Virtex-7 XC7VX690T FPGA [49], and uses a Xilinx Zynq System-on-Chip (SoC) device, running embedded Linux, to provide board support functions. The board provides 14 bidirectional links to the uTCA backplane, that can be run at up to

4.8 Gb/s. These dedicated connections, on a single uTCA crate, allow data sharing across the calorimeter regions. In addition, the board is equipped with Avago CPX and MiniPOD modules, that provide 67 optical inputs and 48 optical outputs running at up to 10 Gb/s. Additionally, there are Gigabit-Ethernet (GbE) and Data Acquisition (DAQ) links on the backplane [50].

2. **The Master Processor 7 (MP7) board**, designed by the Imperial College. The MP7 processor is based on a Xilinx Virtex-7 XC7VX690T FPGA [49], and Avago MiniPOD optics, and provides 72 optical input and 72 optical output serial links, capable of operating at over 10 Gb/s, giving a total bandwidth of 0.75 + 0.75 Tb/s. It was initially designed for the calorimeter layer-2 time-multiplexed algorithm, that requires no backplane interconnection, but a high optical bandwidth is needed to stream all data from an event to layer-2 for processing. However, the board's all-optical data interface made it a rather generic stream-processing engine, allowing other trigger subsystems to utilize it. During Run 2, CaloL2, BMTF, uGMT and uGT subsystems all used the MP7 board [51].

3. **The Modular Track Finder 7 (MTF7) board**, designed by the University of Florida. The name of the unit – Modular Track Finder – reflects the modular design of the board. It consists of three modules; the *Core Logic*, the *Optical* and the *Look-Up Table* modules. The *Core Logic* module contains the processing logic, based on a Xilinx Virtex-7 FPGA [49], a smaller control FPGA, power supplies and the clock circuitry. The *Optical module* contains the Avago MiniPOD optical receiver and transmitter modules, providing 84 input links and 28 output links at 10 Gbps. Finally, the *Look-Up Table* module contains 1 GB of low-latency, random-access memory, that is used for assigning the final transverse momentum of the muon candidate tracks. The card was designed for the needs of the Endcap Muon Track finder (EMTF), but was also used for the Overlap Muon Track Finder (OMTF) [52].

4. **The AMC13 (AMC13) board**, developed by Boston University. The AMC13's name is derived from the fact that it populates the uTCA crate slot intended for a redundant uTCA Carrier Hub, making it the 13^{th} AMC card of the crate. The AMC13 receives timing data from Timing and Control Distribution System (TCDS), and delivers payload data to the DAQ, thus acting as an interface for the subsystem's data processor boards, which populate the AMC slots [53]. The AMC13 is comprised of three different modules called "tongues". Tongue 1 is powered by a Xilinx Kintex FPGA and is the main processing unit of the board. Tongue 2 houses a Xilinx Spartan 6 FPGA and an Atmel microcontroller that provides power management and sensor monitoring functions. Finally, Tongue 3 houses the frond-panel connectors for JTAG and the MMC console. The AMC13 board collects all data from each AMC card, through a 5 Gbps backplane link. Upon receiving the L1A signal, it combines the buffered information from all AMC cards to a single framed output event, and transmits the information to the Central Data Acquisition (CDAQ) of CMS, through a fiber optic link, at 10 Gbps.

5. **The TwinMux board**, developed by the INFN in Italy. It is equipped with 6 front panel SNAP12 receivers and 2 Minipods (one transmitter and one receiver) for high speed data transmission (up to 13 Gbps). The heart of the system is a *Xilinx XC7VX330-3TFFG1761 Virtex-7* FPGA and the clock distribution is based on two very low jitter PLLs that can broadcast to all the FPGAs. An *ATmega128* microcontroler is included for power management and other MMC functions. The TwinMux merges several 480 Mb/s trigger links

to higher speed serial links, and compensates delays to provide bunch crossing alignment of the trigger data coming from the different inputs. Twelve of the 72 TwinMux inputs are routable to GTH inputs, to be able to handle the Gigabit Optical Link (GOL)-based 1.6 Gb/s links [54].

6. **The Concentration Pre-Processing and Fan-out (CPPF) board**, developed by the University of Chinese Academy of Sciences. The design of the card is based on two FPGAs. A large Xilinx *Virtex-7 C7VX415-2TFFG1158* FPGA implements the main functions, such as the algorithms and the high speed links, of the CPPF system. The second FPGA is a smaller *Xilinx Kintex-7 XC7K70-2TFBG484* device and it is used as a Control Unit, interconnecting the AMC to the network and controlling several components in the card. The board also includes minipod optics, one 32-bit microcontroller, DDR3 memory, Flash memory, I^{2}C and UART components. The Avago MiniPOD optical modules provide 48 input links and 12 output links at 10 Gbps. The CPPF boards are designed to concentrate, synchronize and transmit RPC data from the overlap and endcap regions to the OMTF and EMTF systems respectively [55].

All the uTCA boards that were developed for the CMS Level-1 Trigger upgrade, during the Long Shutdown 1, can be seen in Figure 5.5.

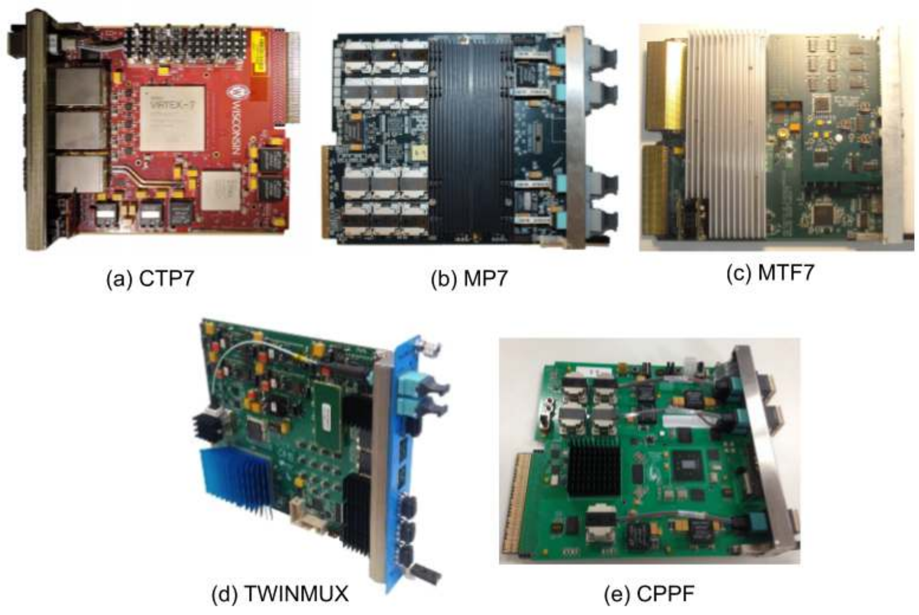


Figure 5.5: The uTCA boards that were developed for the CMS Level-1 Trigger upgrade.

5.3 The Calorimeter Trigger

The Calorimeter trigger system was upgraded during the Long Shutdown 1, to cope with the high luminosity conditions of the LHC. The ECAL TPG system was updated to provide an optical, rather than copper, output and the HCAL TPG system was completely replaced as part of the stage-1 upgrade of the HCAL back-end electronics. The upgraded system has a two-layer architecture; Layer-1 consists of *"pre-processor"* cards and Layer-2 of *"main processor"* or *"processing node"* cards as can be seen in Figure 5.6.

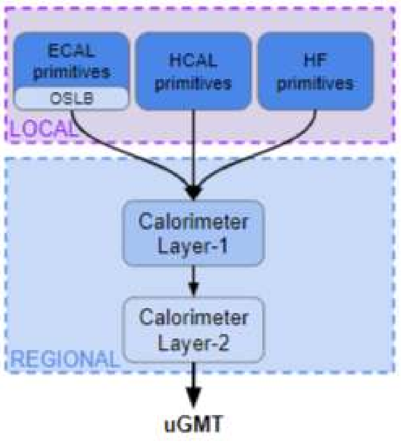


Figure 5.6: The CMS calorimeter trigger architecture after the first Long Shutdown.

For the architecture of the system, a spatially-pipelined, Time-Multiplexed Trigger (TMT) was chosen. In the conventional trigger, the regional segmentation was maintained in the processing stages that were operating synchronously, making the sharing between the different regional processors necessary. The main difference between the TMT architecture and the conventional one is that the data from an individual bunch crossing are routed to a single processor through a static multiplexing network. The processors are not synchronous, but are arranged to be one LHC clock cycle out of phase with their neighbours, so that the data should arrive sequentially. A comparison between a conventional trigger and the TMT is shown in Figure 5.7 [56, 57].

5.3.1 The Calorimeter Trigger Layer-1

The Calorimeter Trigger Layer-1 (CaloL1) receives data, in the form of an 8-bit E_T and a quality bit from the ECAL, HCAL and HF towers. In the barrel region, each tower is formed by 5×5 crystals, with an average size of approximately 0.087×0.087 in $\eta \times \phi$. In the endcap, the crystals are arranged in a x-y geometry, and the η dimension of trigger towers grows with η, with the number of crystals per trigger tower varying from 25 at η=1.5 to 10 at η=2.8. Finally, the forward region calorimeter is segmented in 4×18 towers in $\eta \times \phi$, corresponding to 4×4 HB or HE towers. The trigger towers are combined in 4×4 trigger groups, to form calorimeter regions which form the

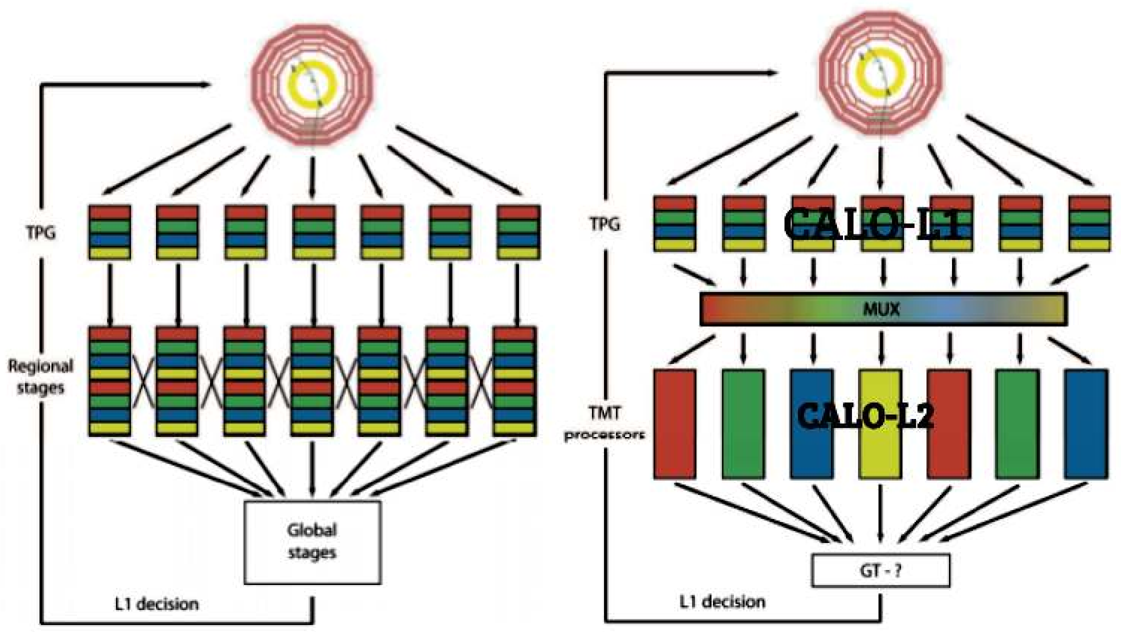


Figure 5.7: A comparison between a *conventional trigger* (LEFT) and TMT (RIGHT). In the conventional trigger, the regional segmentation is maintained in the processing stages which operate synchronously, necessitating sharing between the different regional processors, as illustrated by the diagonal lines between them. In the TMT data are routed through a multiplexing network, which directs all the data from an individual bunch crossing to a single processor. The processors are not synchronous, but are arranged to be one LHC clock cycle out of phase with their neighbours, so that data should arrive sequentially in a round robin fashion.

basis of the jet and energy triggers. In the forward region, each segment is considered a calorimeter region.

The complete CaloL1 consists of 18 CTP7 cards. Each card receives 64 optical input links in total, adding up to 576 input links running at 4.8 Gbps, and 648 input links running at 10 Gbps for the whole system. The Layer-1 is organized in 36 calorimeter slices, with each slice consisting of a calorimeter section spanning (0.5 in η) $\times$ 0.35 (4 towers) in ϕ (Figure 5.8). Each of the 18 CTP7s cover 2 calorimeter slices. The Layer-1 subsystem aligns and decodes input data, applies tower-level calibration in lookup tables, builds combined trigger tower words and streams them to the Layer-2 for further processing. The 18 CTP7 boards are placed in 3 uTCA crates (6 boards in each). The results are transmitted to the CaloL2 over 648 optical fibers (36 per board), using an asynchronous 10 Gbps protocol [50].

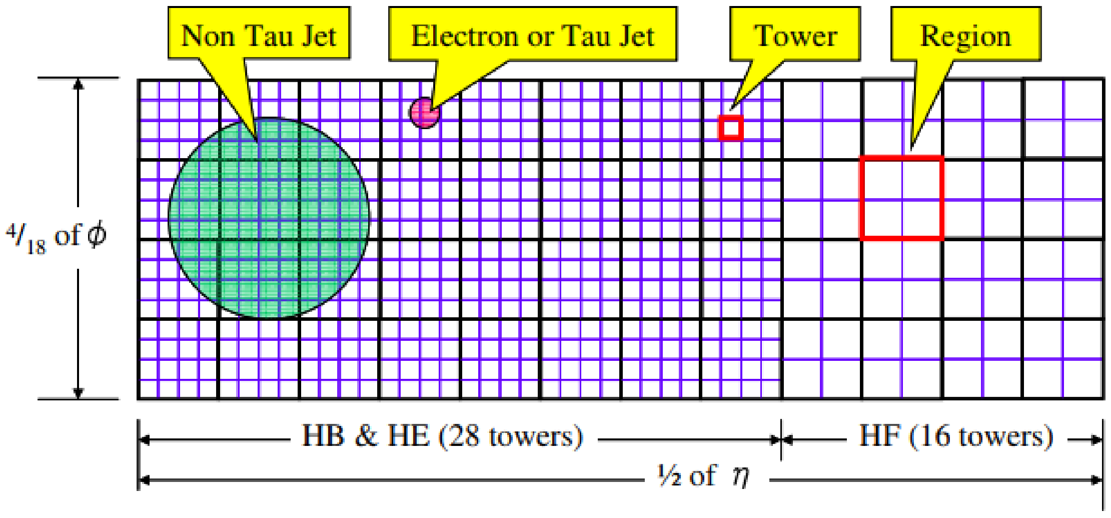


Figure 5.8: A graphical view of the CMS calorimeter trigger slice of the HCAL. The geometry of the barrel and endcap ECAL is identical [58].

5.3.2 The Calorimeter Trigger Layer-2

The Calorimeter Trigger Layer-2 consists of 9 MP7s that host all calorimeter algorithms, responsible for finding particle candidates, and computing global energy sums, and one MP7 that functions as a de-multiplexer board. Due to the use of the TMT, each processor has access to a whole event at trigger tower granularity. The volume of incoming data and the algorithm latency are fixed, so the position of all data within the system is fully deterministic, and no complex scheduling mechanism is required. Each of the processing boards receives energy sums for every calorimeter trigger tower over 72 optical links (four optical links per Layer-1 board), operating at 10 Gbps (Figure 5.9).

The layer-2 algorithms are fully pipelined and start processing as soon as the minimum amount of data is received. A dynamic clustering algorithm adapts the size of the cluster, to precisely match the electron footprint in the calorimeter, to reconstruct precisely lepton signatures, instead of using a a fixed-sized sliding window. This dynamical technique allows the construction of basic clusters, that are combined to reconstruct hadronic τ lepton candidates. The cluster shapes produced by the e/γ and τ finder algorithms are categorized to provide further background discrimination. The reconstruction of particle jets is based on a fixed-size sliding window, centered around a local maximum. The full calorimeter granularity allows the calculation of other global quantities, such as the total E_T, the missing transverse energy (MET) and HT, the jet-based equivalent of the total ET as well as MHT (the jet-based equivalent of MET). The pile-up energy around a jet candidate is calculated using a local pile-up correction technique, called “chunky donut”, on an event-to-event basis. Finally, the trigger candidates are sent to the de-multiplexer board, that formats the data and propagates them to the Global Trigger [59].

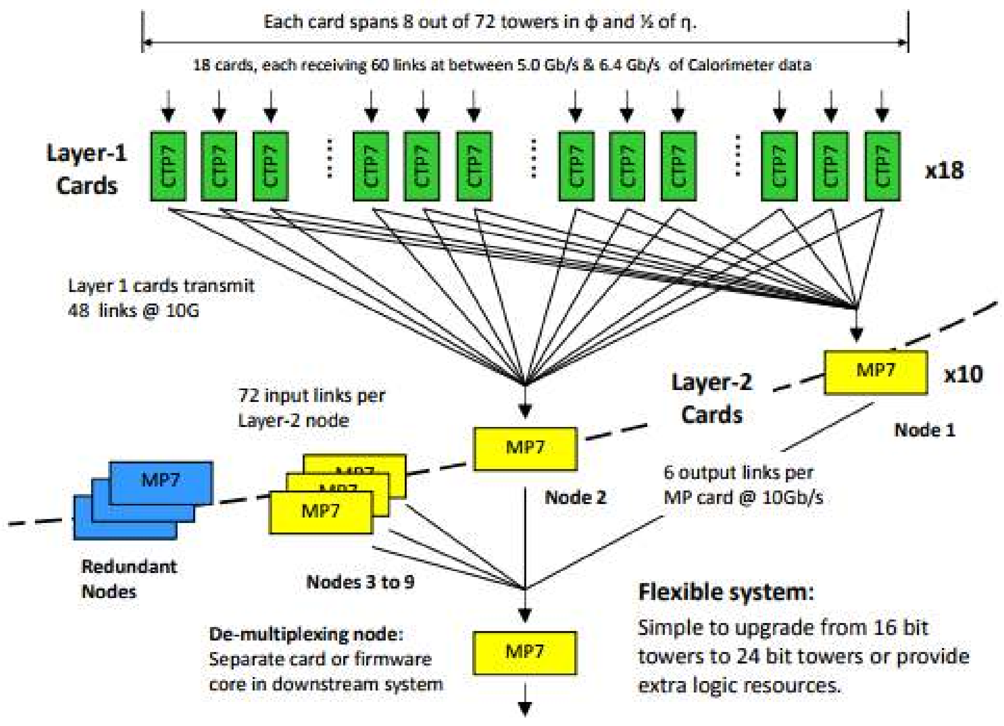

Each card spans 8 out of 72 towers in ϕ and ½ of η.
18 cards, each receiving 60 links at between 5.0 Gb/s & 6.4 Gb/s of Calorimeter data
Layer-1 Cards
CTP7
x18
Layer 1 cards transmit 48 links @ 10G
72 input links per Layer-2 node
Layer-2 Cards
MP7
x10
Node 1
Node 2
6 output links per MP card @ 10Gb/s
Redundant Nodes
Nodes 3 to 9
Flexible system:
Simple to upgrade from 16 bit towers to 24 bit towers or provide extra logic resources.
De-multiplexing node:
Separate card or firmware core in downstream system

Figure 5.9: The upgraded Time-Multiplexed Trigger architecture. The outputs of the Layer-1 CTP7 corresponding to one event are transmitted to single processing nodes in the second layer, Layer-2.

5.4 The Muon Trigger

The main focus of the muon trigger system is to identify and measure the momentum of muons using the magnetic field in the steel yoke of the CMS solenoid. In the legacy system, the redundancy of the muon detection systems was preserved, until they were combined in the Global Muon Trigger (uGMT). The upgraded muon system, however, utilises this redundancy earlier in the trigger processing chain, by dividing the L1 muon trigger in three subsystems, representing three η regions; the Barrel region ($0.83 < |\eta|$), the Overlap region ($0.83 < |\eta| < 1.24$) and the Endcap Region ($1.24 < |\eta| < 2.4$). By combining the muon hits from the different detector systems in each region, the number of chamber hits, along a muon trajectory being propagated to the Muon Track Finders, is maximized. This results in improved precision and higher number of position and angular measurements participating in the track. Moreover, isolation criteria are being applied by the uGMT system, using the energy deposits in the calorimeter, in order to further reduce the muon rate from heavy flavor jets.

The block diagram of the L1 muon trigger of CMS is shown in Figure 5.10. An extra layer was added, with respect to the legacy system, to act as trigger primitives concentrator. The updated L1 trigger system is divided in 4 layers:

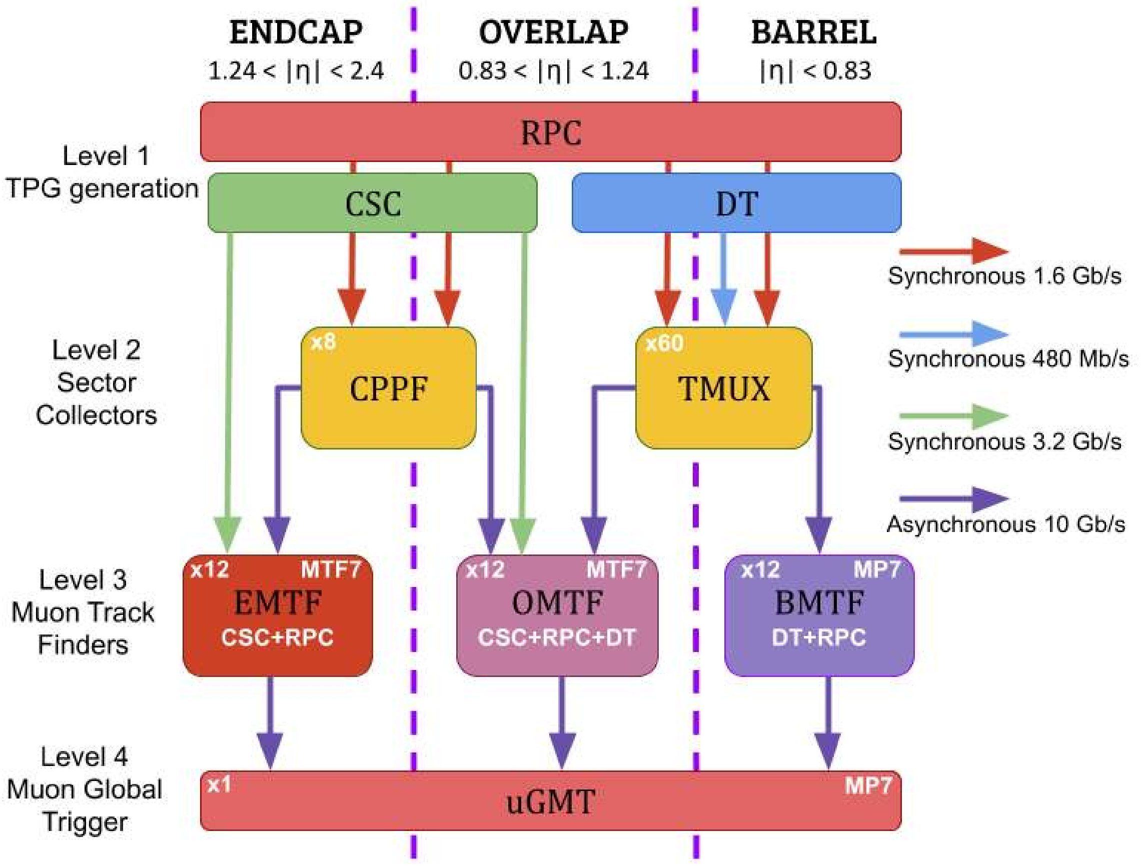


Figure 5.10: The L1 Muon Trigger architecture of the upgraded Level-1 muon system. The TwinMUX system receives DT track segments and RPC hits from the barrel region, and propagates them directly to the OMTF system, while constructing track segments by combining DT and RPC information for use by the BMTF. The CPPF concentrates hits from the RPC in the endcap region and propagates this information to the EMTF system. Hits from the CSC detectors are sent to the OMTF and EMTF without pre-processing. The OMTF also receives the RPC hits directly. The muon tracks are reconstructed by the Endcap, Overlap, and Barrel Muon Track Finders. The fully reconstructed tracks are propagated to the uGMT, where information from the CaloL2 system is used for isolation. Finally, the best eight muon tracks are sent to the uGT. The arrow colors indicate the optical link transmission protocols.

- **Layer-1 : Trigger Primitive (TP) generators**. They are the Front-End electronics of the detector.
- **Layer-2 : Trigger Primitive Concentrators**. All trigger primitives are being pre-processed and propagated to the muon track finders. TwinMux receives TPs from DTs and RPCs, combines them to the so-called *super-primitives*, and propagates them to the BMTF. CPPF receives RPC data from the endcap region and delivers them to the EMTF and OMTF.
- **Layer-3 : Muon Track Finders**. The main subsystems running the algorithms responsible for the muon detection and reconstruction of their tracks. There are three subsystems, one

for each η region of the detector.

- **Layer-4 : Ghost Cancelling, Sorting and Isolation.** The uGMT subsystem is responsible for applying a final sorting algorithm in the three track finders and cancels-out duplicate muons, found at the boundary between neighbouring track-finders. The uGMT also computes the isolation of a muon, based on the energy deposited in the calorimeter trigger towers around the muon's track.

5.4.1 The Barrel Muon Trigger

The barrel region of the detector is spanning in the $|\eta| < 0.83$ region. The upgraded muon trigger chain includes the adaptive layer for the track finder in the barrel region. This layer merges the DT, RPC and HO TPs, unburdening the trigger processors. The TwinMux boards are in charge of sending such combined primitives to the BMTF. In addition, a clasterization algorithm is applied to the RPC hits, and along with the DT data, they are sent to the OMTF. In both cases, a scale up in the transmission rate, and hence a reduction in the number of links, is provided. TwinMux is also responsible for duplicating the trigger primitives, in order to reduce connections between trigger processors, increasing the reliability of the system. In the barrel region, the combined primitives are sent to the BMTF, that implements the legacy trigger algorithm of the DT track finder, with the addition of several improvements in rate reduction at higher efficiency and quality [60].

The Barrel Concentrator (TwinMux)

Each TwinMux processor receives DT and RPC links from one sector of the barrel muon detector, adding to a total of 60 boards. The DT input consists of *trigger segments*, including position, direction, quality and BX information. The TwinMUX receives the DT track segments at 480 Mbit/s from the DT mini-crates via the Copper-to-Optical Fiber (CuOF) boards, that convert the galvanic inputs from the mini-crates to optical signals. The RPC hit information is sent by the RPC link boards at 1.6 Gbit/s. The RPC input consists of hits, including position and BX information.

The output data to the Barrel Muon Track Finder use the same data format as the DT Trigger Segments, and are obtained with the following algorithm. First, the input data are de-serialised and synchronised. Then, a clustering algorithm is applied to RPC hits, with the neighbouring hits being merged, and the resulting cluster position is assigned with half-strip resolution and converted into DT coordinates. In case the same RPC cluster fires in two consecutive BXs, the second one is suppressed. RPC clusters close, in ϕ, to DT Trigger Segments from the same chamber are matched in a $\pm$ 1 BX time window, centered around the Trigger Segment's BX. The closest RPC cluster is selected. If the $\Delta\phi$, with respect to the DT Trigger Segment, satisfies $\Delta\phi \leq 15mrad$, RPC and DT are considered matched and the *super-primitive* quality bit is set. If the DT Trigger Segment was built with less than 8 DT ϕ layers, the Trigger Segment BX is shifted to match the RPC cluster BX. If the DT Trigger Segment was built with all 8 DT ϕ layers, its BX is not changed. If no match with an RPC cluster was found, the original DT Trigger Segment is propagated to the

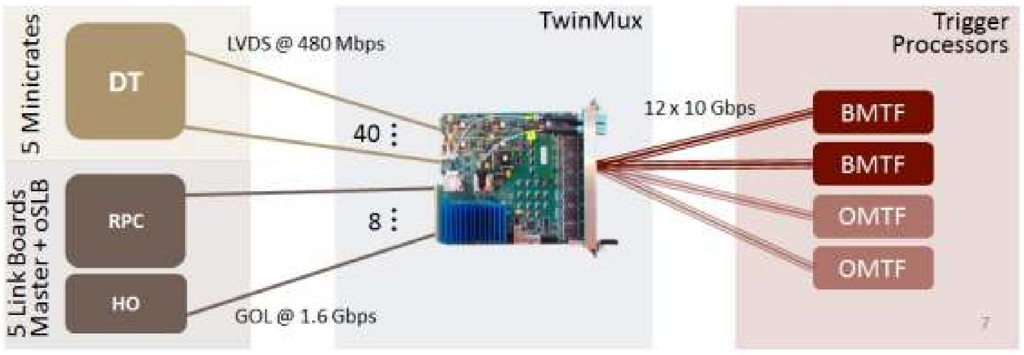


Figure 5.11: The barrel muon trigger chain. TwinMux collects optical links at 480 Mb/s from the DT minicrates and at 1.6 Gb/s from the RPCs and HO. It then merges and fans out data to barrel and overlap trigger processors through 10 Gb/s links [60].

BMTF processor, and the *super-primitive* quality bit is not set. The TwinMUX system constructs up to four *super-primitives* per muon station, and fans them out to the BMTF processors for a given wedge, as well as its neighbouring wedges. (Figure 5.11). It also forwards the track segment data along the η direction for each muon station.

Lastly, the TwinMUX boards also provide DT track segments, from the barrel muon stations belonging to the overlap area, towards the OMTF system. TwinMux boards are also used as the read-out system for the DT detector.

The Barrel Muon Track Finder (BMTF)

The Barrel Muon Track Finder receives all hits produced by the DT chambers and the RPCs installed in the central region of the detector for $|\eta| < 0.83$. For the upgraded BMTF, the MP7 boards were introduced to allow the development of more sophisticated algorithms, improving the muon p_T assignment performance. The more advanced FPGAs of the MP7 boards, also allowed the algorithms to be clocked at higher frequencies, reducing the latency of the system.

The Barrel region of the detector is divided into 12 wedges in the ϕ plane. Each of the track finder boards is looking for muons in one wedge, requiring 12 MP7 cards to be used in total. The cards are placed in two uTCA crates, with processors of wedges 1 to 6 placed in the top crate and processors of wedges 7 to 12 placed in the bottom crate, as illustrated in Figure 5.12. Each board collects the DT/RPC primitives, through 30 optical channels running at 10 Gbps, and sends the best 3 reconstructed muons to the uGMT, through one output optical channel also running at 10 Gbps.

Compared with the legacy muon track finder, the BMTF has improved the p_T granularity, by extending the Look-Up Tables (LUTs) and the assigned bits from 5 to 9. As a result, the p_T has a linear p_T scale starting from 3 GeV and increasing, in steps of 0.5 GeV, to a maximum value of

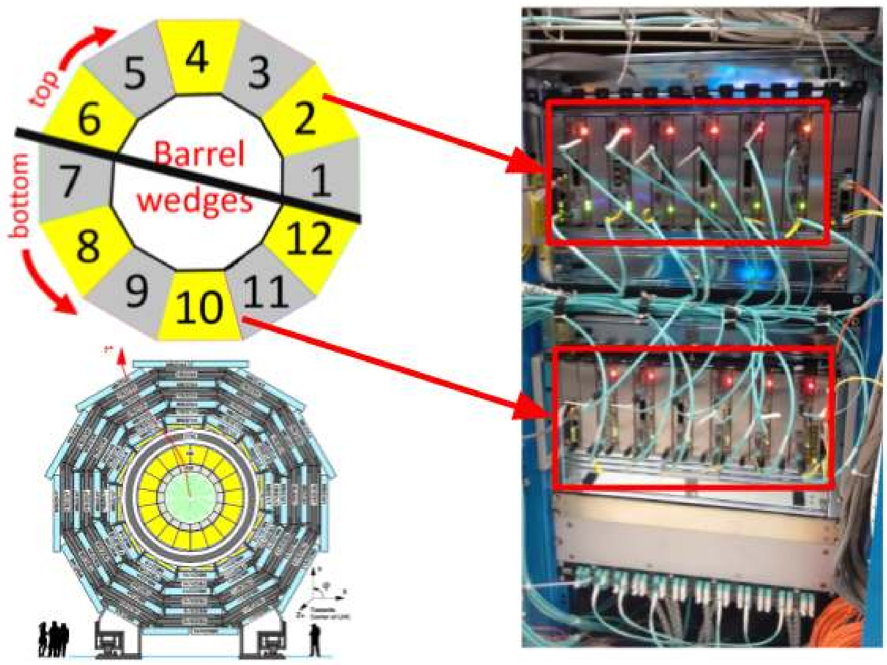


Figure 5.12: The BMTF system is split into 12 processors. Each processor looks for muons in one barrel wedge. The boards are placed in two crates, splitting the barrel in two areas. The TOP area includes wedges 1 to 6 and the BOTTOM area includes wedges 7 to 12.

140 GeV. The upgraded BMTF contains also an extra algorithm for p_T assignment. The coordinates of each muon are calculated, in each processor, by 12 ϕ-track finders and 6 η- track finders, running in parallel. More details on the BMTF system is presented in section 6.1, since it is an important part of the present thesis.

5.4.2 The Endcap Muon Trigger

The Endcap Muon Track finder (EMTF) receives all hits produced by the CSCs and the RPCs, installed in the forward region of the detector ($0.83 < |\eta| < 1.24$). For the upgraded EMTF, the MTF7 boards were used, as they provide 1 GB of low-latency random-access memory to accommodate the LUTs required for assigning the final transverse momentum of the muon candidate tracks. Each of the two end-caps is divided into 6 trigger sectors in ϕ, covering a 60° angle. Twelve EMTF processors are required to cover all 12 sectors. Every board is receiving CSC and RPC information from the CPPF boards that cover one sector plus 20° of the neighbouring sectors.

The EMTF receives Local Charged Tracks (LCTs) from the CSCs frond-end electronics, through optical links running at 3.2 Gbps. The RPCs send LCTs to the CPPF card, which concentrates them and sends the clustered RPC information, together with coordinates, to the EMTF, through optical links running at 10 Gbps.

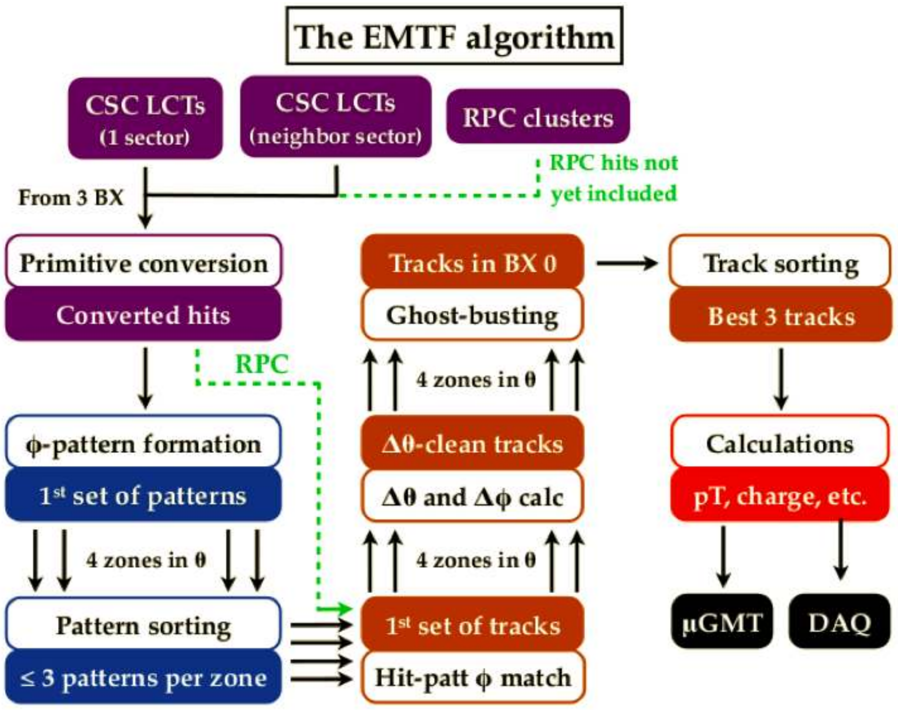


Figure 5.13: The block diagram of the Endcap Muon Track finder.

Figure 5.13 illustrates the consecutive steps of the muon track reconstruction in the EMTF. The EMTF algorithm firstly converts the LCT primitives into hits, combining θ and ϕ coordinates information. Each LCT is assigned to one or two zones, based on the θ value where the algorithm checks for patterns, matching the ϕ values of the LCT trigger primitive across the station. After the patterns based on ϕ information have been formed, only the three highest in quality patterns are kept. The quality is defined by the straightness of the formed pattern. The next step of the algorithm is the track building, where the RPC hits are matched (based on the ϕ) to the LCTs' patterns, in order to build the track candidates. After this step, LCTs that are too scattered in θ ($\Delta\theta > 0.15$ to at least one LCT in another station) are removed and duplicate tracks are discarded. Only the three highest quality tracks per sector are kept, and the p_T, charge, η and ϕ are assigned to them using look-up tables. The information of the 36 possible muon track candidates from the 12 EMTF boards is then sent to the uGMT, using fiber channels running at 10 Gbps [61].

5.4.3 The Overlap Muon Trigger

The overlap region is defined as the η slice where DTs, RPCs and CSCs overlap ($0.83 < |\eta| < 1.24$). The fact that the orientation of the muon detectors is different and the magnetic field in this region is not homogeneous, forced the development of a dedicated Overlap Muon Track Finder (OMTF). The OMTF system is comprised of 12 MTF7 boards, each of which receives and processes DT track segments, CSC track stubs, and RPC hit information from a 60° detector sector. The CSC input links run at 3.2 Gb/s, and the RPC and DT input links at 10 Gbps (sent by the data concentrators; TwinMux and CPPF).

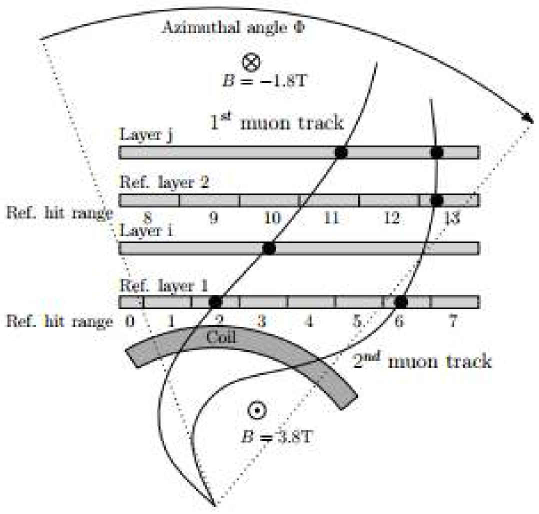


Figure 5.14: Schematic view of the analysis of a multi-muon event based on reference hits. In the example shown here, two muons are detected. Each of them generates three hits in different layers. Taking the reference hits, the OMTF algorithm, would first detect the 1st muon and then twice the 2nd muon. The *ghost-busting* algorithm at the end of the OMTF processor, filters out the superfluous detection of the 2^{nd} muon.

The OMTF algorithm is based on pre-computed hit patterns, called Golden Patterns (GPs), with given p_T range and charge sign. After the incoming data streams are synchronised to each other and to the LHC clock, the GP algorithm compares the hit patterns for a given event with the stored set of GPs. The algorithm calculates the $\Delta\phi$ between hits in all the detector's layers and a selected reference hit, and uses it in every GP to find the one with the smallest difference to the actual hit pattern (Figure 5.14). The track parameters, corresponding to the selected GP, are assigned to the muon. Finally, a *ghost-busting* algorithm is applied, to eliminate duplicate muon tracks, before transmitting the best three tracks per 60°sector to the uGMT. More details on the OMTF's GP algorithm can be found in [62].

5.4.4 The Global Muon Trigger

The Global Muon Trigger is implemented in an MP7 card that provides 72 input and 72 output links, running at line rates of 10 Gbps. Each track finder system propagates up to 36 muons to the uGMT, through 12 fibers, while the calorimeter trigger requires 28 links to provide the energy sum information, necessary to compute isolation variables. In total, uGMT uses 64 out of the 72 input channels of the MP7 board. The 8 best possible muons are transmitted to the uGT through 4 fibers. All input and output data is being received and transmitted in 32-bit words, using 10b/8b coding at 10 Gbps.

The uGMT algorithm receives the de-serialised data, and converts the local coordinates from each track finder subsystem to the global reference coordinate system. The next step is to remove all duplicate muons, that were detected in the overlapping regions of the neighbouring track finder subsystems. The remaining muons are ranked based on both reconstruction quality and transverse momentum, before finding the eight highest ranked muons, in a fast sorting module. Furthermore, the algorithm uses energy deposits, received from the calorimeter system, to compute isolation variables for each muon selected for transmission. These variables are being propagated to the uGT, in order to identify and reduce the number of muons originating from hadronic jets. In addition, uGMT extrapolates the muon tracks back to the interaction region, in order to improve the assignment of the isolation variables. The extrapolated azimuthal coordinates are additionally forwarded to the uGT, in order to improve the computation of the invariant mass. Finally, the best 8 possible muons are serialised and transmitted to the uGT [63].

5.5 The Global Trigger

The Global Trigger system is the last stage of the Level-1 Trigger system and decides whether to reject or accept a physics event for further evaluation by the High Level Trigger. As with the other CMS upgraded trigger systems, uGT follows the uTCA standard. The system is mainly comprised of two MP7 cards, capable of using up to 512 algorithms, running in parallel, but it is designed to support up to six MP7 boards in parallel. The Global Trigger receives trigger objects from the Calorimeter Trigger, the Global Muon Trigger and additional sources, the so-called *external conditions*. The 256 external conditions LVDS signals are being received using RJ-45 connectors. A galvanic patch panel concentrates them into 64 Very High Density Cable Interconnect (VHDCI) connectors, in order to allow the space efficient reception of all the signals on a single AMC module. A specialized AMC board, the AMC502, was designed by Vadatech, to accommodate these 64 external condition signals, convert them to the appropriate format and transmit them to the MP7 boards via optical channels. The AMC502 boards are equipped with a Xilinx Kintex-7 FPGA, capable of transmitting data at a rate of 12.5 Gb/s, and two mezzanine board expansion slots. The first mezzanine module accommodates the inputs for the 64 galvanic signals, while the second module transmits the data to the MP7 board via fast 10 Gb/s optical transceivers. An illustration of the external conditions signals data flow can be seen in Figure 5.16.

The upgraded uGT was designed to deal with the conditions of Run 2 of the LHC, explained in section 5.2. To achieve this goal, the system receives more trigger objects than the old system, and supports more sophisticated algorithms. For the muons, 8 instead of 4 candidates are being sent to the uGT. The upgraded calorimeter system also sends more data to the uGT; 12 e/γ objects instead of 8, 12 jets instead of 8, and 8 tau objects instead of 4 during Run 1. By the same token, the amount of external conditions signals, received from other detector sources, was doubled and is now 256 bits. The number of algorithms is also significantly higher than before. The upgraded Global Trigger supports the use of at least 512 different trigger algorithms, four times more than the legacy system.

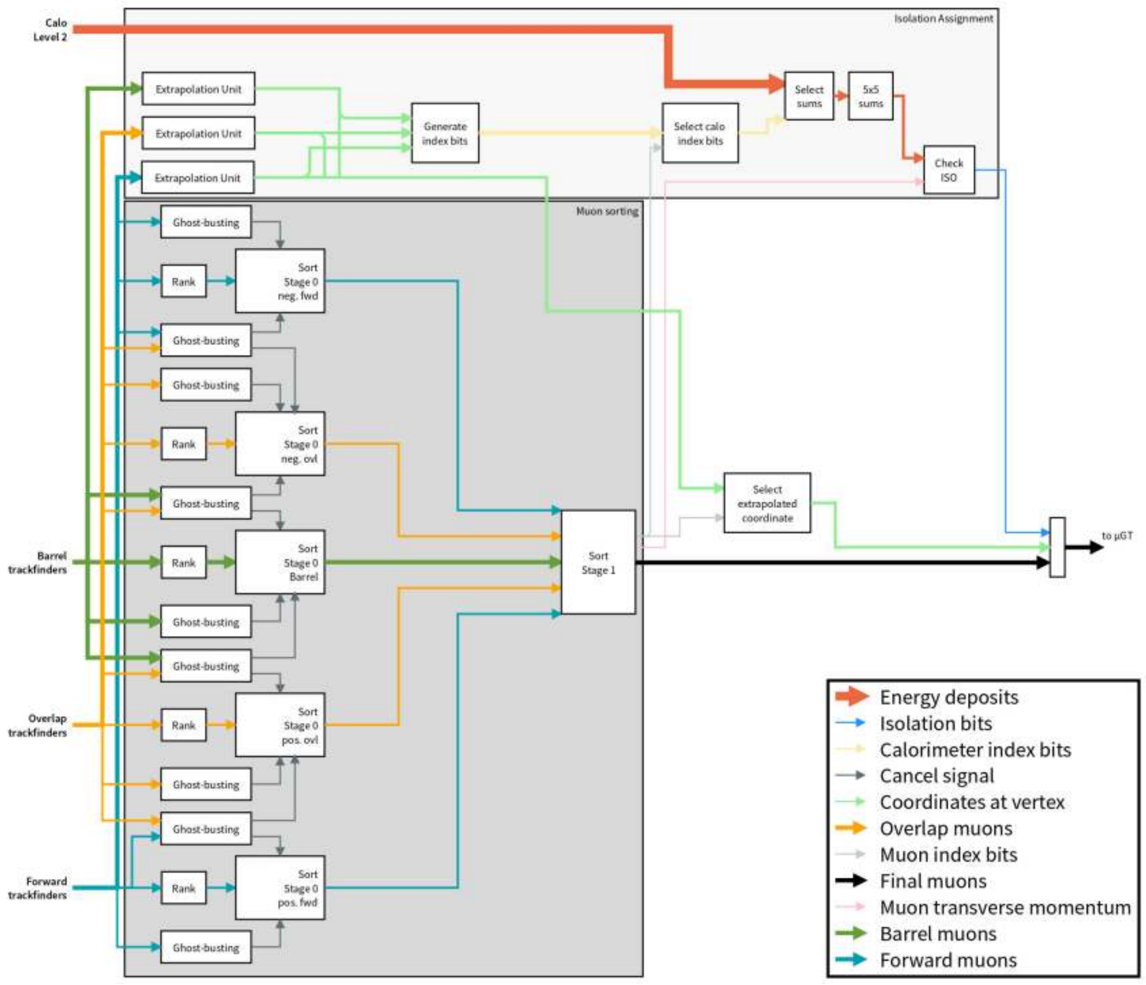


Figure 5.15: A block diagram of the uGMT algorithm. Muons received from the three track finding systems are sorted in two stages. In parallel, muon tracks are extrapolated to the interaction point, and muon isolation variables are computed by using energy sum information received from the calorimeter trigger. The best eight muons are sent to the uGT [63].

The trigger algorithms are compiled in a so-called "*trigger menu*". Access to this menu is possible using a special framework, that offers a graphical interface for modifying the L1T menu. The updated menu is stored in an XML file, which is used to generate the VHDL code for the uGT firmware. In addition, each algorithm can be programmed to only fire every nth detected event, a feature called prescaling. Finally, the increased performance of the modern FPGAs capable for more complex calculations, allowed the integration of invariant mass calculations in the trigger menu. In the legacy system, invariant mass calculations were possible only in the High Level Trigger.

During the 2016 run, the physics algorithms implemented in the trigger menu exceeded the chip resources of one MP7 module, thus a second board was added. The technical trigger and

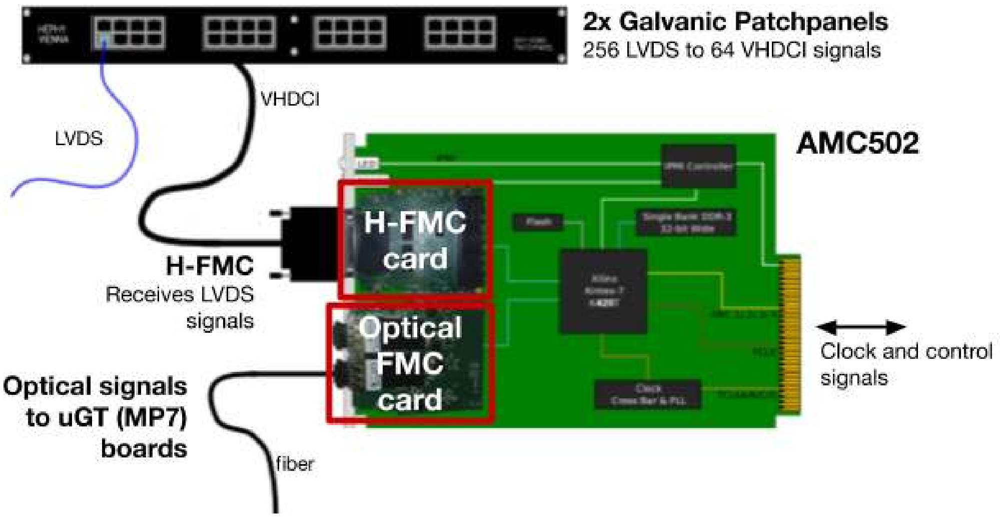


Figure 5.16: The External Conditions signals data flow.

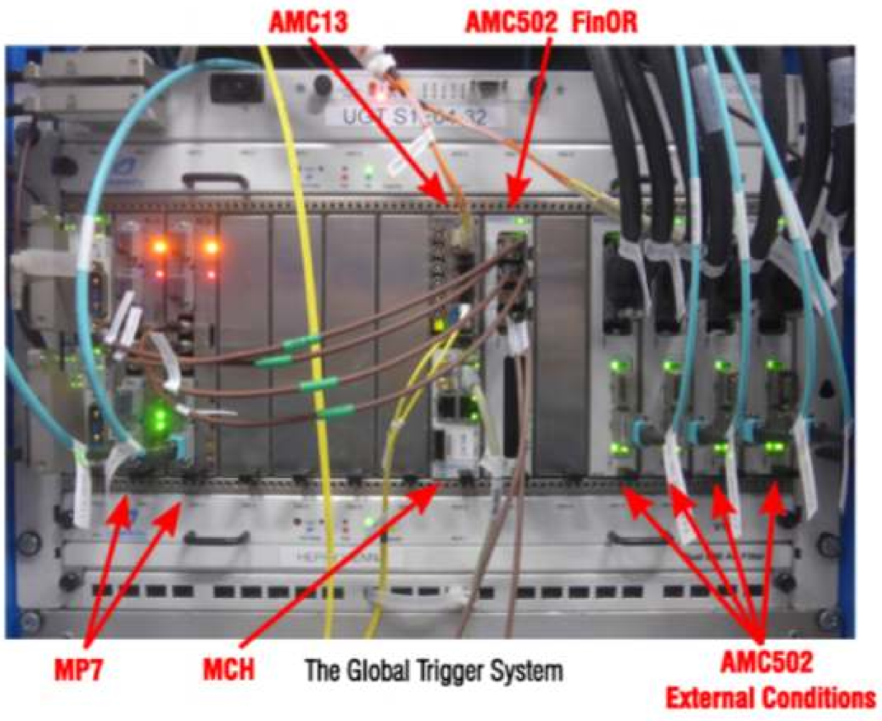


Figure 5.17: The Global Trigger hardware setup during 2017 data taking [64].

external conditions signals were merged, to allow them to be used either as veto signals or in combination with other inputs to trigger a read-out signal. All the trigger decisions from the two uGT boards and the combined external conditions and technical trigger signals are combined in an AMC502 card, that is responsible for the final L1 trigger signal, known as *Final-OR* (FinOR). The 2017 hardware setup of the uGT system can be seen in Figure 5.17 [64].

5.6 The High Level Trigger

The High Level Trigger (HLT) is the second and final level of the CMS trigger system. Its main purpose is to further reduce the recorded collision rate, to a level that is manageable by the Data Acquisition (DAQ) system and reconstruction. The maximum acceptance rate of the DAQ and storage is 1 kHz, which means that the HLT must reduce the L1T rate of 100 kHz by a factor of 100. The rate reduction is achieved by accepting the largest possible cross-section of the interesting physics events from the collisions and rejecting efficiently the non interesting ones.

While the L1T is based on FPGAs and ASICs, to run fast and relatively simple trigger algorithms, the HLT is software implemented, running on a farm of commercial computers that includes about 16,000 CPU cores. This architecture allows the use of the same sophisticated software, used for the offline reconstruction and analysis, but optimized, in order to comply with the strict time requirements of the online selection.

The HLT is divided in paths, with each path being a sequence of reconstruction and filtering, that reproduces the offline selection for a given physics object or a combination of them. Each path consists of blocks of object producers and filters, that are arranged in increased complexity. Faster algorithms are run first and their products are filtered. If a filter fails, the rest of the path, along with the more CPU-time consuming algorithms, is skipped. The final HLT decision is the logical OR of all the trigger paths within the menu. The modular structure of the HLT menu is illustrated in Figure 5.18.

The HLT paths use a reconstruction technique, widely used in the CMS analysis, called Particle Flow (PF). The algorithm identifies particles individually and clusters them into more complex objects, making use of the full detector information to describe the global collision event. One of the most demanding tasks of the HLT is track reconstruction. In order to reduce the CPU time, the algorithm iteratively removes hits once associated with tracks, reducing in that way the combinatorics in the subsequent steps, starting with prompt, high p_T tracks, then it looks for displaced ones and finally for less frequent track topologies. Lastly, the HLT algorithms provide efficient methods for Pile-Up mitigation, by applying minimum p_T thresholds and vertex constraints to tracks and other objects [65].

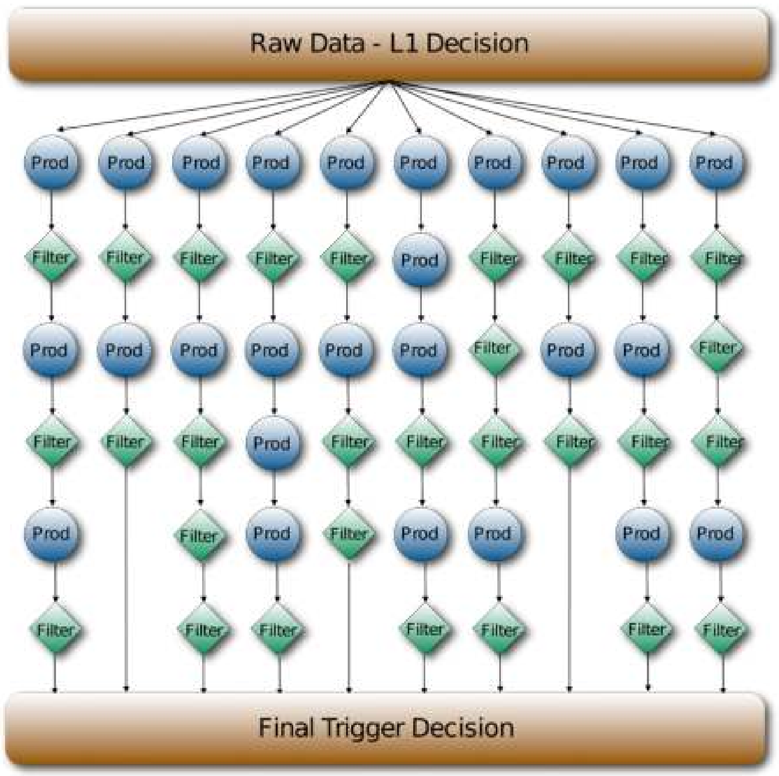


Figure 5.18: Schematic representation of the modular approach of the HLT menu in CMS. The final trigger decision is the logical OR of the decisions of all single paths [65].

5.7 The Level-1 Muon Trigger for Phase-2

The demanding conditions of the HL-LHC and the new features that will be provided by the new CMS sub-detectors will require a full upgrade of the L1T system electronics. The upgraded system will be based on the use of state of the art FPGAs and processors that will provide enough computational power for the new sophisticated and more demanding trigger algorithms. New high-speed optical links will accommodate the propagation of data from across the detector to the same processing boards. This global view of the detector by single boards will allow the precise calculation of global quantities, such as missing transverse energy. Finally, the adoption of a flexible, modular architecture will be required for system adaptation to different HL-LHC running conditions.

In Figure 5.19, the conceptual design of the CMS Phase-2 Trigger System is presented. The main differences to the current L1T system are the inclusion of information from the tracker and the High Granularity Endcap Calorimeter (HG-CAL), and the introduction of a Correlator layer that will combine information from multiple sub-detectors. The total L1T latency will increase from 4 μs to 12.5 μs and the output bandwidth from 100 kHz to 750 kHz. The new trigger will be implemented in four independent data paths: the Calorimeter path, the Tracker path, the Muons path and the Particle Flow path [66].

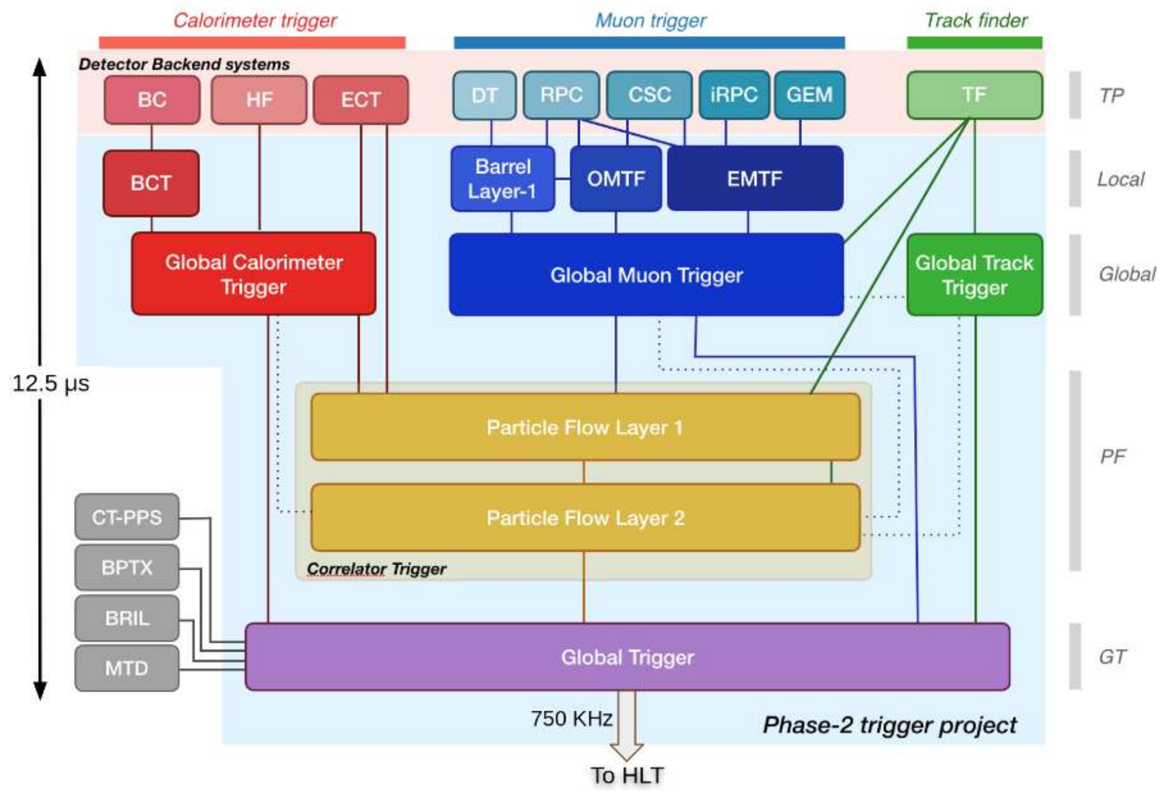


Figure 5.19: The functional diagram of the CMS Level-1 Trigger (L1T) design for Phase 2. The new trigger system will be split in four independent paths; the Calorimeter, the Tracker, the Muon and the Correlator path. Information from all paths will be combined into the Global Trigger that will calculate the final trigger decision. The total maximum L1T latency will be 12.5μs and the output bandwidth 750 kHz [66].

The Calorimeter Trigger path will process high-granularity information from the Barrel and Endcap calorimeters and from the forward calorimeter. The post-processing cluster and identification variables will be send to the Global Trigger Calorimeter, where the calorimeter objects (electrons, jets, hadronic taus etc.) will be built.

The Track Trigger path will receive reconstructed track parameters and quality information from the back-end track finder electronics of the outer tracker. The parameters will be provided to the trigger system, that will attempt to match them with calorimeter and muon objects. An extra independent processing layer, the Global Track Trigger (GTT), will provide tracker-only objects and reconstruct the primary vertex of the event.

The Muon Trigger path will retain its basic structure, covering three separate regions, the Barrel, the Overlap and the Endcap, with each region having a dedicated track finding subsystem

(BMTF, OMTF and EMTF respectively). The Global Muon Trigger (GMT) will receive standalone muons from OMTF and EMTF, and stubs from Layer-1 BMTF, along with tracker tracks from the outer tracker. After the removal of duplicate muons, the GMT will attempt to match the muon tracks to tracker tracks, producing the so called track-matched muons.

The Particle-Flow Trigger, known as the Correlator Trigger, will receive the Calorimeter clusters and tracks, the event primary vertex candidate from the GTT, and track-matched muons from the GMT. The particle flow reconstruction algorithm will be implemented in two steps. In the first step, the algorithm will match the calorimeter clusters with tracks, and produce the particle flow candidates. In the second step, it will build and sort the final trigger objects, and apply additional identification and isolation criteria.

Finally, the Global Trigger will receive information from all trigger paths, and will produce the L1A signal, based on a menu of algorithms. The L1A will be distributed to the TCDS, and will initiate the read-out to the DAQ.

Chapter 6

The Barrel Muon Track finder System and upgrades

6.1 The legacy Barrel Muon Track Finder algorithm

As described in section 5.4.1, the Barrel Muon Track Finder is responsible for identifying muons and reconstructing their tracks in the CMS barrel muon area, located at the central region of the CMS detector ($|\eta| < 0.83$). The BMTF receives super-primitives, built by the TwinMUX system, from DT and RPC hits. These super-primitives describe track segments within a detector module, and consist of the 12 bits of the ϕ-coordinate, 10 bits of the bending angle and 3 quality bits indicating the confidence in the measurement. The BMTF also receives hit information along the η direction for each muon station in a given wedge, consisting of 7 bits of η hits and 7 bits of η quality information per η-station. The RPC hits are used in the TwinMUX to correct the measurement of track segments, that were assigned to a bunch-crossing too early or too late, by the DT system.

There are 60 DT sectors, and each sector consists of 4 muon stations. The 3 innermost stations are made of 3 superlayers[1]. The central superlayer measures the η coordinate, while the two outer superlayers measure the ϕ coordinate. The fourth station has only two ϕ superlayers.

The CMS barrel muon detector is split in 12 wedges each measuring 30°in ϕ. Each wedge is split in five sectors in η. The local trigger electronics of each DT sector send at most two track segments per muon station to one TwinMUX board, along with the corresponding RPC information. The TwinMux boards merge the DT and RPC TPs, and fan-out the combined primitives to the BMTF. In addition, the information from neighbouring wedges is duplicated by the TwinMUX system and shared with the given track finder processor, in order to avoid inefficiencies at the edges of a wedge. The data are received over thirty 10 Gbit/s optical links from the 15 TwinMUX processors, responsible for a given wedge and its neighbours (Figure 6.2).

[1]A superlayer is made of 4 planes of drift cells glued together.

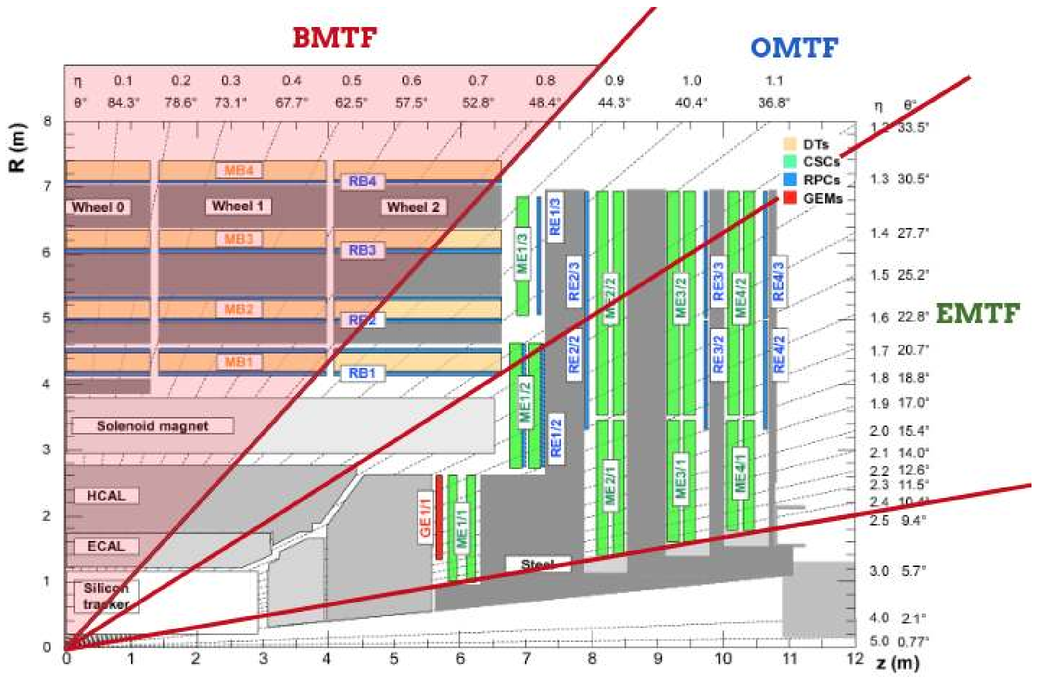


Figure 6.1: The muon detectors in a longitudinal view. The BMTF subsystem covers the central region of the CMS detector ($|\eta| < 0.83$) area.

6.1.1 BMTF inputs and deserializer units

Each input link for the BMTF will transmit the ϕ and η coordinates and quality of one super primitive per DT station, from the four DT stations of a sector. Since every DT station can transmit up to two trigger primitives, two input links are required, to send all the information to the BMTF processor.

Each deserialization unit receives the data from the two links (1 sector), and breaks them down to 6 x 32-bit words per link in the 240-MHz clock domain. It then proceeds to merge the information, and deliver a total of 384 bits to the algorithmic unit. A detailed view of the input payload from a single DT chamber can be seen in Figure 6.3. Apart from deserializing the data, the unit applies programmable quality thresholds, in order to reduce the noise from the DTs, and allow the masking of possible noisy DT channels on the fly. All track segments, with quality less than 2, are considered noise and are discarded. In addition, all track segments, with quality between 4 and 6, are considered of very high quality, and are assigned a “correlated flag” bit that is later used in the parameter assignment unit. Finally, the unit generates input data flags, used in the monitoring blocks.

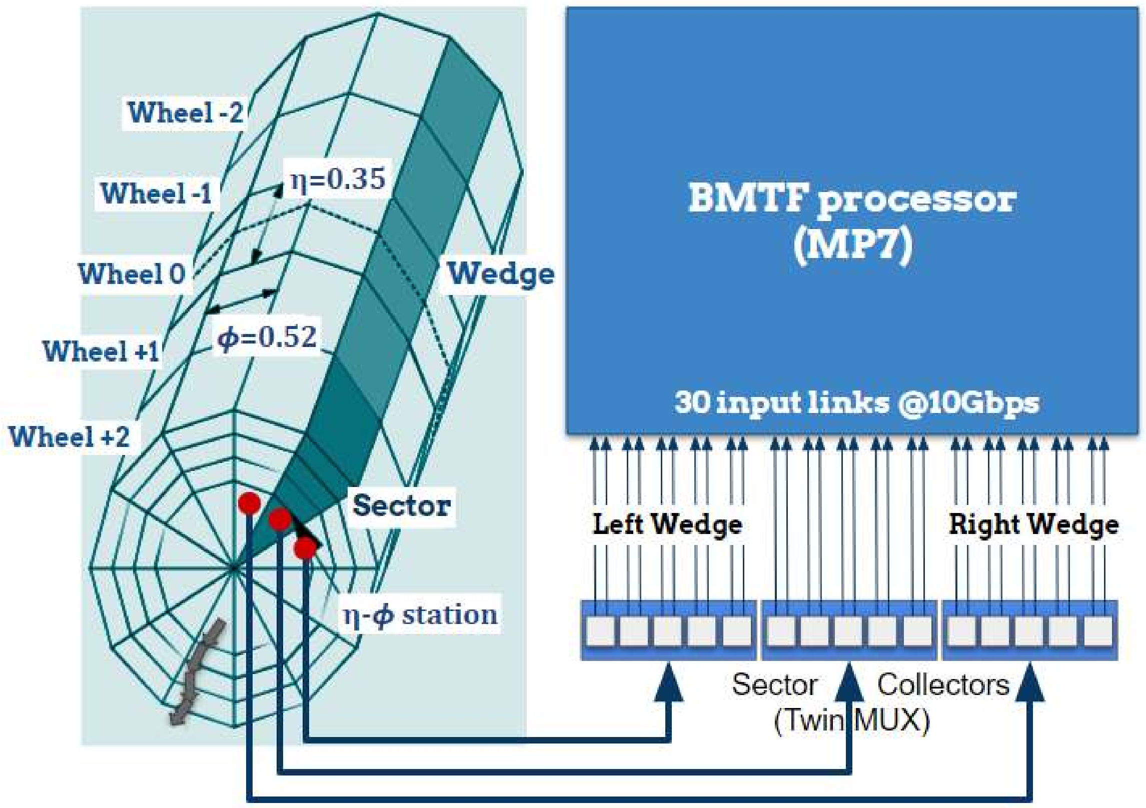


Figure 6.2: Each BMTF processor card receives information from one wedge and its two neighboring wedges, through 30 optical links (10 for each wedge) running at 10 Gbps. These links are grouped in pairs, and the 15 muon sectors from the 3 wedges are connected to one BMTF card.

6.1.2 The Phi Track Finder (PHTF)

After deserialization, the data are transmitted to the Phi Track Finder (PHTF) blocks. The main purpose of the PHTF is to identify and join compatible track segments, in order to reconstruct muon tracks and assign transverse momentum, charge, location and quality parameters. This is done in three steps; the extrapolation, the track assembly and the parameters assignment.

There is only one PHTF unit for every DT sector in a wedge, except for wheel zero. Wheel zero is split logically into "trigger wheel +0" and "trigger wheel -0", with each slice having its own PHTF. Since muons originating from the center of CMS can cross sector boundaries, every PHTF requires information from the 3 sectors of its own wheel and also from those of their neighboring wheels of higher $|\eta|$. In wheel 0, the "trigger wheel +0" PHTF block processes muons that either remain in wheel 0 or cross to wheel +1, while the "trigger wheel -0" PHTF block processes muons that cross from wheel 0 to wheel -1.

Frame 1	BX Cnt	NC	C	R	S	Qual st 1	ϕ-bending station 1 (10-bit)	ϕ station 1 (12-bit)
32 bits	31 30	29 28	27	26	25	24 23 22	21 20 19 18 17 16 15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
Frame 2	BX Cnt	NC	C	R	S	Qual st 2	ϕ-bending station 2 (10-bit)	ϕ station 2 (12-bit)
32 bits	31 30	29 28	27	26	25	24 23 22	21 20 19 18 17 16 15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
Frame 3	BX Cnt	NC	C	R	S	Qual st 3	Reserved or ϕ-bending st 3 (10-bit)	ϕ station 3 (12-bit)
32 bits	31 30	29 28	27	26	25	24 23 22	21 20 19 18 17 16 15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
Frame 4	BX Cnt	NC	C	R	S	Qual st 4	ϕ-bending station 4 (10-bit)	ϕ station 4 (12-bit)
32 bits	31 30	29 28	27	26	25	24 23 22	21 20 19 18 17 16 15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0

Frame 5	Reserved (10-bit)	ZS	Eta Hits/Qual st 3 (7-bit)	Eta Hits/Qual st 2 (7-bit)	Eta Hits/Qual st 1 (7-bit)
32 bits	31 30 29 28 27 26 25 24 23 22	21	20 19 18 17 16 15 14	13 12 11 10 9 8 7	6 5 4 3 2 1 0
Frame 6	Reserved (32-bit)				
32 bits	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0				

Figure 6.3: DT data sent to BMTF from TwinMUX over one bunch crossing. In every bunch crossing (25 ns), 192 bits are transmitted per link. The information is then deserialized in six 32-bit words. The first four words contain the relative position (ϕ, 12 bits), the bending angle (ϕ_b, 10 bits) and the quality (q, 3 bits) of a segment inside each of the four chambers of a sector. The fifth word contains the η hit coordinates and quality (7 bits) from the 3 η chambers in a sector. Finally, all the 32 bits of the sixth word are reserved.

The extrapolation unit

The extrapolating blocks determine if track segment pairs originate from the same muon track. Starting from a source segment, as shown in Figure 6.4, they look for target segments that are compatible with respect to the position and bending angle in the other muon stations. Extrapolation from stations 1 and 2 to all other stations are performed. For station 3, extrapolation to station 4 is not possible, due to its very small bending angle. However, a backwards extrapolation from station 4 to station 3 is performed. For each track segment pair, there is an extrapolation window, which is determined by its bending angle (ϕ_b) [46]. The bending angles are related as shown Equation 6.1.

$$\begin{aligned} \phi_{ext} &= \phi_{source} + \phi_{deviation}(\phi_{b,source}) \\ threshold_{ext} &\geq |\phi_{ext} - \phi_{target}| \end{aligned} \tag{6.1}$$

The extrapolation window parameters are pre-calculated and stored in FPGA memories called Look-Up Tables (LUTs). A successful track segment matching occurs when the destination ϕ position lies within the boundaries of the calculated extrapolation window. The pair is tagged with a bit. All extrapolation tags are sent to the track assembly units which link compatible track segments into complete tracks. Figure 6.4 illustrates the extrapolation logic in ϕ.

Parameter	bits	Range	Description
ϕ	12	-2048 - 2047	Relative position of a segment inside a sector
ϕ_b	10	-512 - 511	Bending angle
quality	3	0 - 7	Indicates the number of superlayers used
η hits	7	"pattern"	Each bit corresponds to one chamber area 0 : no hit (less then 3 SLs hit) 1 : hit (3 or more SLs hit)
η quality	7	"pattern"	Each bit corresponds to one chamber area 0 : 3 SLs hit 1 : 4 SLs hit
ZS	1	0, 1	0 : Zero Supression enabled 1 : Zero Supression disabled
BX cnt	2	0 - 3	The two LSB of the bunch counter

Table 6.1: Given in this table are the scale and definitions of the super primitive parameters transmitted to the BMTF.

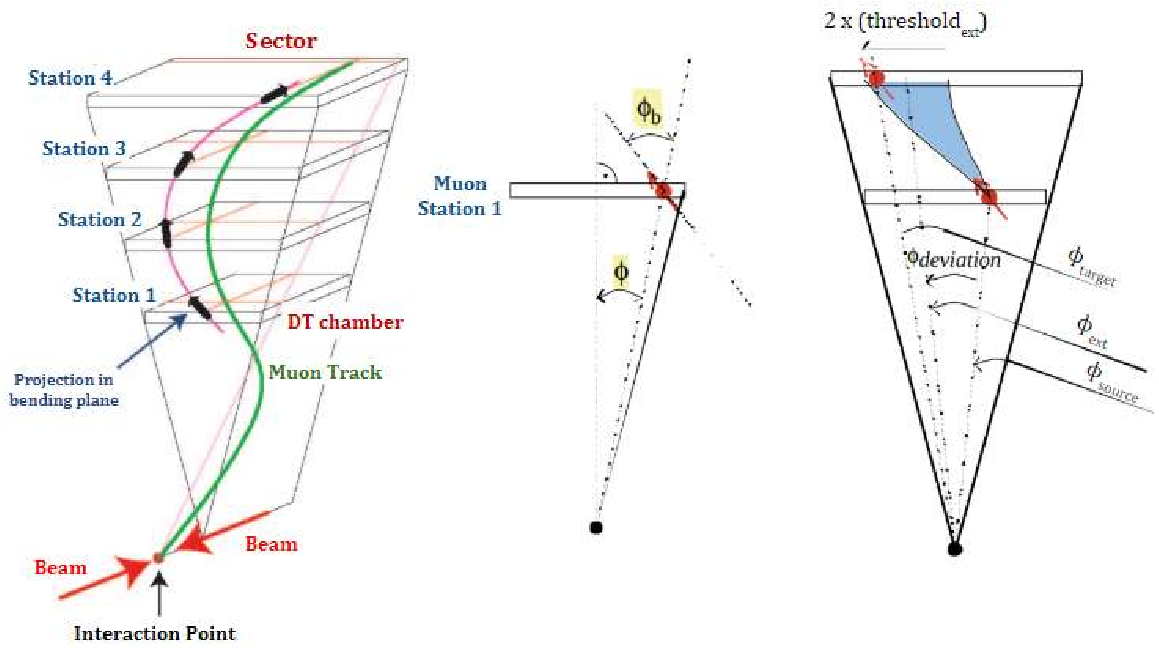
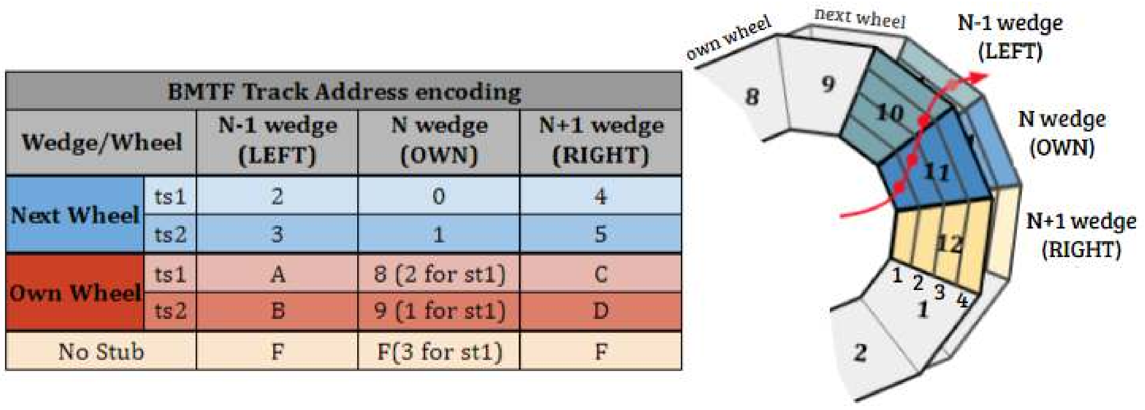


Figure 6.4: The basic concept of the track segment matching using extrapolation windows. If a track segment is found to be within the extrapolation window, given by ϕ_{ext} and $threshold_{ext}$, the extrapolation is considered successful.

The Track Assembly Unit

The next step is to determine which track segments originate from a single muon track. The track assembly unit combines the extrapolation results in a priority scheme, and builds the longest possible track. The procedure is repeated with the remaining (unused) track segments, in an attempt to build a second muon track. Every track segment from stations 2, 3 and 4, that was successfully matched, is given a 4-bit index number. A muon originating from the interaction point of CMS, can traverse one wheel of the barrel muon system, or cross to the next wheel of higher η. In the ϕ plane, due to the existence of the magnetic field, it will either turn right or left depending on its charge. Consequently, a muon can either traverse only one wedge, or cross to its right or left neighboring wedges. The track index number indicates the track segment's wheel (own wheel or next wheel), and wedge (center or neighboring, left or right) (see Figure 6.5). Furthermore, each DT station sends up to two *track segments* (*ts1* and *ts2*). The second track segment (*ts2*) is send to TwinMux, using the same channel, but with a delay. TwinMux then propagates both track segments to the BMTF, synchronously over two fibers. The index number for station 1 track segments, can only originate from the central wheel and wedge, thus we only need 2 bits to encode the information. All four station indices combined, form the 14-bit track address of the muon.

The encoding format of the muon track addresses, together with a track example, are shown in Figure 6.5. In this example, the muon leaves a stub in all four stations. The track assembler matches the tracks from *own-wedge* and *own-wheel* in stations 1 and 2. The muon, subsequently, crosses to the *left-wedge*, leaving a stub at station 3, and finally crosses to the *next-wheel*, leaving a fourth stub at station 4. If all track stubs matched were tagged as *ts1*, the address assigned would be *0x28A2*, while if they were tagged as *ts2*, the address assigned would be *0x19B3*.


BMTF Track Address encoding

Wedge/Wheel		N-1 wedge (LEFT)	N wedge (OWN)	N+1 wedge (RIGHT)
Next Wheel	ts1	2	0	4
	ts2	3	1	5
Own Wheel	ts1	A	8 (2 for st1)	C
	ts2	B	9 (1 for st1)	D
No Stub		F	F(3 for st1)	F

Figure 6.5: LEFT : The encoding format (in hex) of the muon track addresses. RIGHT : A muon, the track assembler matches the tracks from own-wedge / own-wheel station 1, own-wedge / own-wheel station 2, left-wedge / own-wheel station 3, left-wedge / next-wheel station 4.

The assignment unit

The track addresses, calculated in the previous step, along with the pipelined physical parameters of the track segments, are forwarded to the assignment unit. This unit uses the track addresses, to extract the spatial coordinates of the selected track segments, then uses these coordinates to assign transverse momentum (p_T), phi angle (ϕ), charge and quality to the reconstructed muons.

Muon Quality	15	14	13	12	0
DT stations used	1-2-3-4	1-2-3, 1-2-4, 1-3-4, 2-3-4	1-2, 1-3, 1-4	2-3, 2-4, 3-4	No Muon

Table 6.2: The 4-bit muon quality code assignment according to the matched track segments.

The assignment of p_T and ϕ is based on pre-calculated memory-based LUTs, stored in the FPGA memory.

There are two different methods of calculating the p_T. The first method is based on the difference in the spatial ϕ coordinate of the two innermost track segments. However, due to the fact that low-momentum particles bend backwards in the transverse plane, there are cases where the difference in ϕ coordinates is as low as in the case of high-momentum particles. The relationship between the difference in the spatial ϕ and transverse momentum is parametrized using two sets of functions, one set for low transverse momentum track candidates, and one set for high transverse momentum track candidates. The bending angle of the source track segment is used to choose between the "high" and "low" calculated p_T values.

The second method is used purely on the bending angle of the inner station, and uses extra LUTs. In order to extract a precise measurement, only high quality track primitives can be used with this method ($q \geq 4$, i.e. at least two DT superlayers used to measure the bending angle).

Both algorithms run in parallel, and the p_T with the lower value is selected in the end. The measurement of the transverse momentum is given with a precision of 9 bits. The resolution of the measurement is $0.5\ GeV$, with the lower limit being $3\ GeV$ and the upper limit $120\ GeV$.

The assigned ϕ coordinate values correspond to the spatial coordinate of the track segment in the second muon station. If the track candidates do not have a track segment in the second muon station, an extrapolation towards the second muon station position is performed, using memory based LUTs. The measurement of the ϕ coordinate is given with a precision of 8 bits [46].

The quality of the reconstructed muon track is calculated as a function of the number of participating track segments and the momentum resolution, which is better when assigned using the inner muon station track segments. The 4-bit quality bit code is assigned depending on the candidate track class (see Table 6.2). The most significant bit is always 1 for BMTF muons, and it is used by the uGMT to prioritise them over OMTF muons, when cancelling out duplicates.

6.1.3 The Eta Track Finder (ETTF)

The Eta Track Finder (ETTF) receives information from all 5 wheels for each wedge. Each wheel contains η chambers in stations 1, 2 and 3. Each station is segmented in 7 equal sized areas. The η track segments are sent in the form of two 7-bit words per station, one corresponding to the "quality" and one to the "hit". The hit-bit is set to "1", if at least three out of the four planes of the η-superlayer were hit. The quality-bit is set to "1", if all four planes were hit. If only two or less planes were hit, both hit and quality bits are set to "0". (see Figure 6.6)

The fact that the η primitives exhibit recurring patterns, led to the selection of a pattern matching method rather than an extrapolation one. The incoming hit patterns are compared to predefined track patterns generated using simulated data samples. If a track is found, the ETTF attempts to match it with the information from the PHTF. The latter, assigns a very low resolution (coarse) η, which is calculated based on if and when the track crossed the wheels. All the predefined patterns are grouped in categories and each category corresponds to one coarse η value (Figure 6.7). If an η pattern is found in the category that corresponds to the PHTF η, then the matching is considered successful and the low resolution η deduced from the PHTF is replaced by the finer resolution η found in the ETTF.

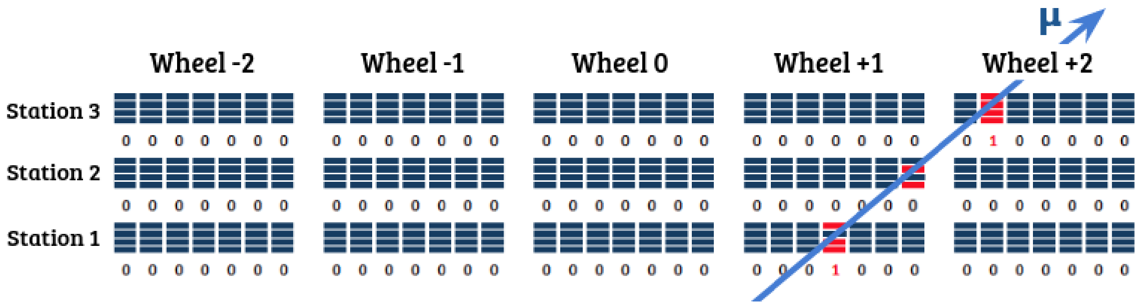
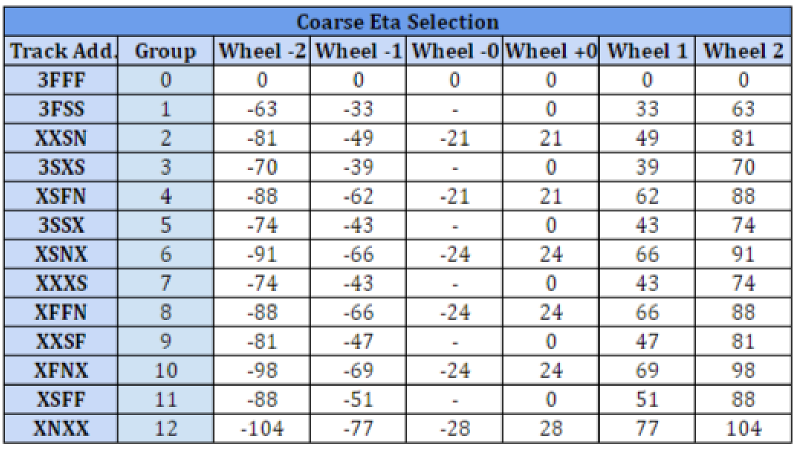


Figure 6.6: Pattern matching scheme in η. Each η chamber is segmented in 7 areas. The information coming from the DT local trigger is delivered as a 14-bit pattern, containing one hit-bit and one quality-bit for each of the seven adjacent chamber areas. A muon must leave a trace in at least 3 out of the 4 planes of the η superlayer, for a hit-bit to be set to "1".



Coarse Eta Selection							
Track Add.	Group	Wheel -2	Wheel -1	Wheel -0	Wheel +0	Wheel 1	Wheel 2
3FFF	0	0	0	0	0	0	0
3FSS	1	-63	-33	-	0	33	63
XXSN	2	-81	-49	-21	21	49	81
3SXS	3	-70	-39	-	0	39	70
XSFN	4	-88	-62	-21	21	62	88
3SSX	5	-74	-43	-	0	43	74
XSNX	6	-91	-66	-24	24	66	91
XXXS	7	-74	-43	-	0	43	74
XFFN	8	-88	-66	-24	24	66	88
XXSF	9	-81	-47	-	0	47	81
XFNX	10	-98	-69	-24	24	69	98
XSFF	11	-88	-51	-	0	51	88
XNXX	12	-104	-77	-28	28	77	104

Figure 6.7: All the low resolution (coarse) η hardware values assigned by the PHTF, based on if and when a track crossed wheels. In the encoded track address "S" corresponds to same wheel, "N" corresponds to next wheel, "3" or "F" corresponds to no hit and "X" means we don't care.

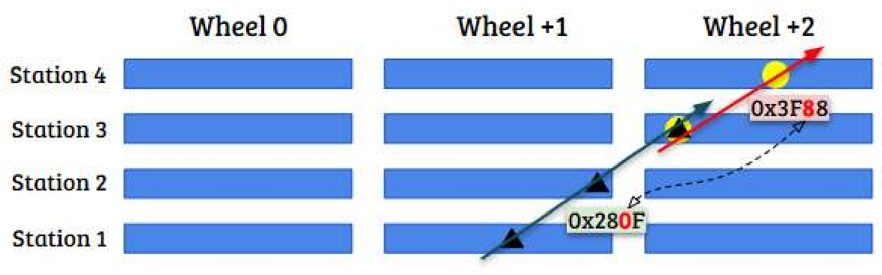
The measurement of the η-coordinate is given in 2's complement with a precision of 9 bits. The ETTF outputs are pipelined, in order to be synchronized with the PHTF outputs, and are then sent to the wedge sorter unit.

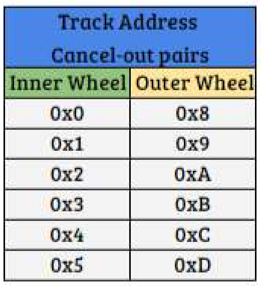
6.1.4 The Wedge Sorter (WS)

The wedge sorter is the final stage of the BMTF algorithm. Each of the six PHTF track finders of a wedge can identify up to two muons, adding to a total of twelve possible muons. The wedge sorter unit performs two main functions; cancelling-out duplicate muons found by two neighboring wheels, and sorting the muons keeping only the three of highest quality.

The cancel-out unit

If a muon track crosses the boundaries between wheels, two neighbouring PHTFs can build the same track, since they operate independently within their own sectors. Thus, a single muon can be reconstructed twice, and two muons could be forwarded to the subsequent stages of the trigger. To avoid this, a cancelling out algorithm was designed, based on the track addresses of the muon candidates and their quality. The cancelling is based on comparing muon tracks between neighboring wheels. If a track segment from muon stations 2, 3 or 4 is common in the two muon tracks, the muon with the lower track quality is discarded. If they have the same quality, then the one with the higher η value is discarded. In Figure 6.8, an example of a duplicate muon track is shown.





Track Address Cancel-out pairs	
Inner Wheel	Outer Wheel
0x0	0x8
0x1	0x9
0x2	0xA
0x3	0xB
0x4	0xC
0x5	0xD

Figure 6.8: LEFT : Example of a duplicate muon. Both PHTFs units in wheel +1 and wheel +2 have used the same track segment in station 3. The cancel out unit identifies the common hit (0x280F vs 0x3F88), and discards the muon with the lower quality, which in this case is the muon originating from wheel +2. RIGHT : A table with all the cancel-out track segment pairs.

The Sorter Unit

The final step of the BMTF algorithm is the sorting of the muons. The Sorter Unit sorts all muons by quality, from highest to lowest. Muons with the same quality are sorted by p_T, from higher to lower. The highest ranked muons are selected for transmission to the next trigger system, the uGMT.

6.1.5 Serialization and BMTF outputs format

The information for each muon is encoded in 64-bit data structures, transmitted within two 32-bit words over the link. The most significant bits (MSBs) of each link word are used to store control bits, with information global to the event under consideration, such as the three least significant bits (LSBs) of the bunch crossing number in the LHC orbit, as well as a bit indicating the beginning of the orbit. All data describing the three muons are transmitted in one bunch crossing, in 6×32-bit words at a data rate of 240 MHz. Figure 6.9 illustrates the input muon data format for transmission via the optical links, and Table 6.3 shows the scale and definitions of the muon parameters transmitted to the uGMT.

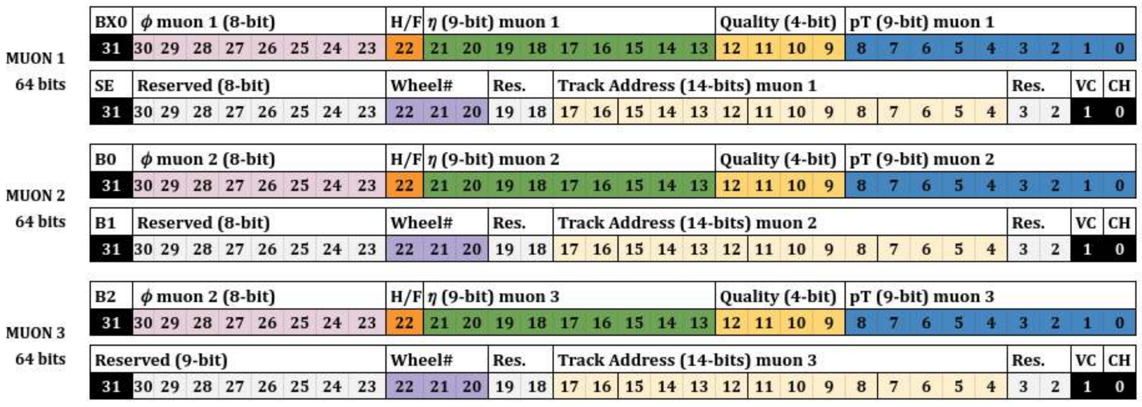


Figure 6.9: Muon data send from BMTF to uGMT over one bunch crossing. BMTF will transmit, at most, 3 muons per bunch crossing to the uGMT using 64-bits for each muon. Each row represents one 32-bit word, thus, two rows represent one muon.

6.2 A Kalman Filter algorithm for the Barrel Muon Track Finder

For the HL-LHC, the DT electronics will be upgraded, providing a drift time resolution of 2 ns. The current BMTF algorithm's memory look-up table approach, allows the use of only two stations for the calculation of transverse momentum. Using all four stations, would require 88 bits of memory address space. In order to improve the momentum resolution and exploit the full bending power

Parameter	Bits	Range	Description
pT	12	-2048 - 2047	The transverse momentum of the muon
η	10	-512 - 511	The pseudorapidity of the muon
quality	3	0 - 7	Indicates the number of track segments used for the track reconstruction
Track Address	14	Encoded	Encodes which track segments were used for the track reconstruction
η quality	1	0, 1	Each bit corresponds to one chamber area 0 : Coarse (no η hits were matched) 1 : Fine
Charge	1	0, 1	0 : Positively charged muon 1 : Negatively charged muon
BX0	1	0 - 3	The two LSB of the bunch counter
B1/2/3	3	0 - 3	
SE	1	0, 1	Sync error 0 : OK 1 : Error

Table 6.3: The scale and definitions of the muon parameters transmitted from the BMTF to the uGMT.

of the CMS solenoid, one more point was added to the calculation, by assuming that the track originates from the center (beam-spot) of the detector. This constraint, however, limits the ability of the BMTF system to identify displaced muons, originating from long-lived particles that decay far from the interaction point.

To address those two issues, a new algorithm featuring a Kalman filter, was developed by the CMS UCLA group. The algorithm was developed within the context of the Phase-2 HL-LHC R&D. However, a version of the Kalman filter was implemented in the current BMTF system during the Run 2 CMS data taking. The algorithm was implemented in HLS. The integration and commissioning of the Kalman filter, in parallel with the legacy algorithm, has been part of this dissertation, and is described in the next sections.

6.2.1 The new Kalman filter algorithm

The Kalman filter has long been regarded as one of the best solutions for tracking and data prediction tasks. Its use in the analysis of visual motion, has been documented frequently. A Kalman filter algorithm can be used in any dynamic system, where there is uncertain information. The Kalman filter can make an educated guess, about what the system is going to do next. The filter is constructed as a mean squared error minimiser, mathematically equivalent to a χ^2 fit [67].

The track parameters at each detector station are given by the state vector $x_n = (k, \phi, \phi_b)$, where $k = q/p_T$. A track is seeded by a stub in the outer available station, and the track parameters and their uncertainties are propagated inwards. The new (x_n) and old (x_{n-1}) states

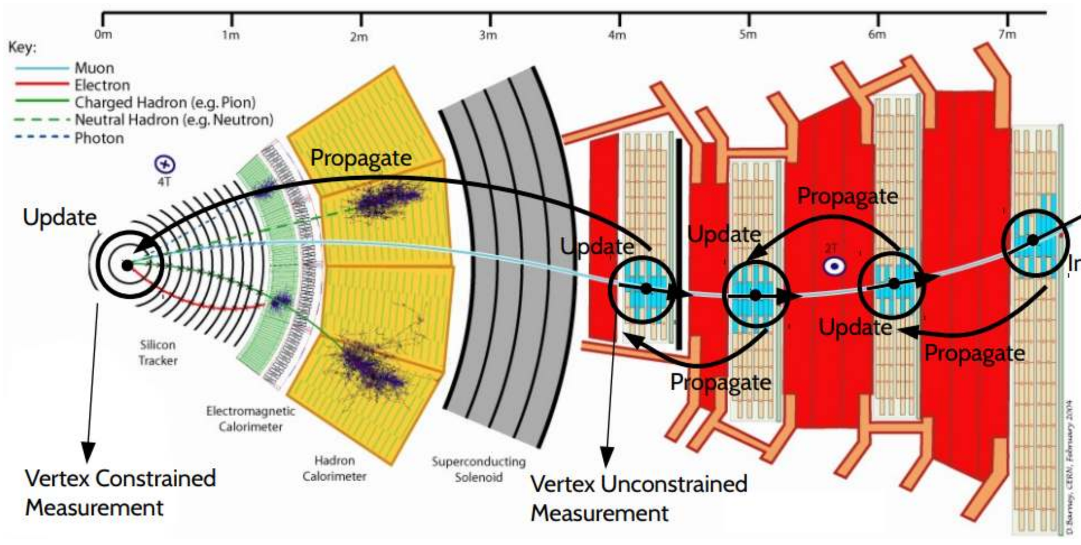


Figure 6.10: The Kalman filter is a sequential algorithm that propagates tracks inwards from station to station, and matches with a stub, updating the track parameters after each propagation. After reaching station 1, a measurement is saved without the vertex constraint, and an extra one after propagating to the vertex.

are related as:

$$x_n = Fx_{n-1} \quad or \quad \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_n = \begin{pmatrix} 1 & 0 & 0 \\ \alpha & 1 & \beta \\ \gamma & 0 & 1-\beta \end{pmatrix} \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_{n-1} \tag{6.2}$$

where the propagation matrix F, consisting of parameters (α, β, γ), is derived by the detector geometry and the B-field, and it differs from station to station. The energy loss is neglected for station-to-station propagation, and added inclusively at the end (see Figure 6.10).

The state uncertainties, expressed by a covariance matrix P, are also propagated to the next station by the transformation:

$$P_{n+1} = FP_nF^T + Q[k, x/X_0] \tag{6.3}$$

where Q is an additional covariance matrix, corresponding to multiple scattering in the return yoke between layers.

After propagating the state to the new layer, the closest stub, consisting of a vector $z_k = (\phi, \phi_b)$, is selected. The track parameters are updated based on the values and uncertainties of the measurement, and a residual r_n is calculated based on Equation 6.4.

$$r_n = z_n - Hx_n \ = \begin{pmatrix} \phi^{stub} \\ \phi_b^{stub} \end{pmatrix}_n - \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_n \tag{6.4}$$

The state update is given by:

$$x_n^{update} = x_n + Gr_n \tag{6.5}$$

where G is the Kalman Gain matrix, that depends on multiple scattering, and the hit-pattern of the already reconstructed track. To save valuable FPGA resources, instead of performing the matrix algebra [68], we pre-calculate the Kalman Gain as a function of k for each hit pattern. The pre-calculated values are stored in a single $4K \times 9$ bit read-only BRAM. The k is compressed in 12 bits, and the gain is defined in 9 bits.

After reaching the first station, a displaced (i.e. without the vertex constraint) measurement of the transverse momentum is stored. Sequentially, the algorithm propagates the track to the vertex, by constraining the trajectory to pass by the center of CMS. While the energy loss was neglected for the station-to-station propagation, for the propagation to the vertex it is taken into account, since the material budget of the calorimeters, the magnet, and the tracker is significant. A second transverse momentum measurement is stored, this time with the vertex constraint, corresponding to the p_T measurement of the legacy BMTF algorithm. This approximate Kalman Filter algorithm decreases the trigger inefficiency by 25% for muons from the beam-spot at the same rate, and provides an efficiency increase by a factor of four for particles displaced by more than 50 cm from the center of CMS [2].

6.2.2 The Kalman filter firmware

The algorithm consists of two major operations, to be implemented in firmware: the propagation of tracks inwards from station to station, and the update, at each step, of the track parameters. The track propagation is a recursive procedure and, as can be seen in Equation 6.2, it requires matrix operations. In order to save FPGA resources, all matrix operations are mapped to DSP slices.

To calculate the number of possible tracks, we have to consider that a track requires at least two stubs, and that the total number of stations are four. There are six 2-stub, four 3-stub and one 4-stub combinations, adding up to a total of 11 possible track combinations (see Figure 6.11). This number must be multiplied by two, because each station can deliver up to two stubs, leading to a total of 22 tracks. These 22 combinations of tracks are implemented in parallel, and each track update, in those 22 track chains, corresponds to a different Kalman gain, to be pre-calculated and mapped in one BRAM.

The second step, the state update, is also implemented in DSPs. Tracks which share hits are

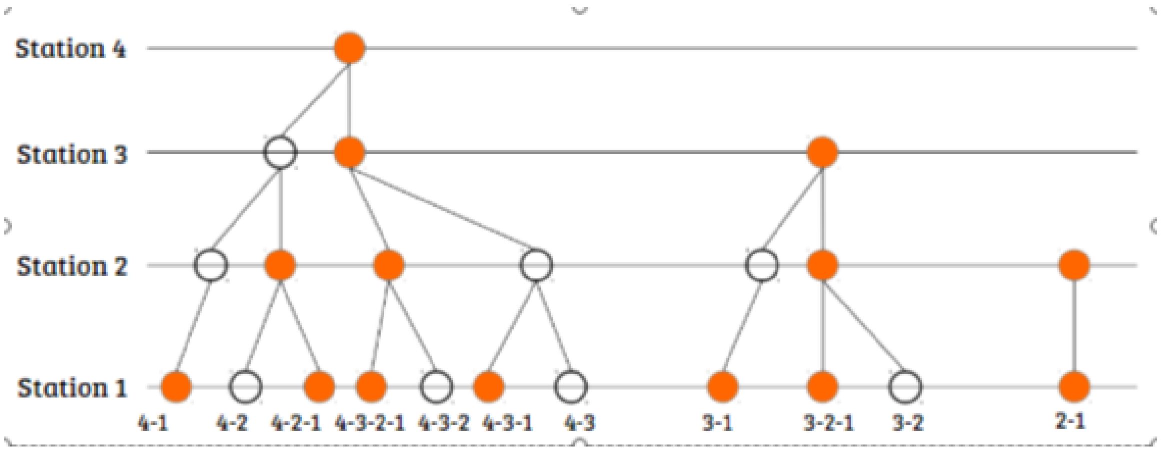


Figure 6.11: The 11 possible stub combinations to reconstruct a muon track.

then cleaned, based on the quality defined by the number of hits and the χ^2 merit function. Finally, in line with the legacy BMTF algorithm, the tracks are sorted by p_T and quality, and the top three are forwarded to the Global Muon Trigger.

6.2.3 Parallel Implementation of the legacy and Kalman filter algorithms

Although the Kalman filter algorithm was initially scheduled to run during Phase II, starting in 2026, the following factors made it possible to implement both algorithms in the same FPGA, and commission it during the 2018 data taking.

1. Low requirements of the Kalman filter algorithm in FPGA resources, due to the extensive use of DSPs.
2. The BMTF latency contingency of 4 BXs, made it possible to trigger with the Kalman filter algorithm firmware, which introduced a latency increase of 2.5 BXs to the legacy system.
3. The 2017 Year-End Technical Stop allowed the development and testing of the new firmware at P5, using cosmic data.

The new Kalman filter algorithm was integrated and debugged in the BMTF system, with simulated and real data. The commissioning of the new firmware was carried out during the 2018 p-p collisions runs. Being part of this dissertation, the tasks that were performed are:

1. Development of an interface between the Kalman IP core and the BMTF framework.
2. Duplication of the input primitives and adaptation of their format to the Kalman algorithm.

3. Addition of timing/area constraints, and usage of implementation strategies to achieve timing closure.

4. Integration of the Kalman algorithm in the simulation test-benches and the debugging software, for the verification of the algorithm.

5. Debugging of the new firmware with simulated and real data at P5.

6. Adaptation of the track address-based cancel-out algorithm between wedges of the BMTF in the uGMT, to reflect the inwards logic of the Kalman track addresses.

During the development stage, two different firmware designs were implemented. In the first implementation, the legacy BMTF outputs are propagated to the uGMT, while the Kalman filter algorithm outputs are read out by the DAQ, for each collected event out of any trigger path in CMS. This implementation exploited real data to debug and improve the algorithm. In the second implementation, the debugged version of the Kalman filter algorithm was used for trigger purposes, while the BMTF algorithm was read out by the DAQ. After the successful deployment of the latter implementation, at the end of 2018 during the heavy ions run, we plan to deploy it online as the default standalone track finder for Run 3 (2021-2023).

6.2.4 Area and Latency considerations

The parameter assignment of the legacy BMTF algorithm, as described in section 6.1, is based in pre-calculated LUTs, stored in the BRAMs of the FPGA. This leads to a very lightweight algorithm, in terms of the utilized resources. As shown in Figure 6.12, the BMTF algorithm utilizes only 12% of the registers (Flip Flops), 33% of the LUTs, while no DSPs are instantiated. After adding the latest Kalman IP core, the resource utilization is increased to 61% of the LUTs, 25% of the registers, and 29% of the DSPs (all dedicated to the Kalman filter algorithm). A comparison of the implementations of the BMTF-only and the BMTF+Kalman Filter algorithms, in the FPGA die of an MP7 board, can be seen in Figure 6.13.

One of the critical constraints of the CMS trigger system is the 3 μs or 120 BXs latency budget, imposed by the size of the read-out buffers of the tracker. The BMTF subsystem is on the critical timing path (the path with most the latency) of the trigger, making the latency budget of any new algorithm an important parameter. After the improvements in the barrel algorithm during the 2015 long shutdown, the total latency budget for the BMTF subsystem was 10.5 BXs. Furthermore, the final, fine tuned, version of the Kalman filter firmware, running at 160 MHz, requires 6.5 BXs for the algorithm, and 2.5 BXs for serialization/deserilization, adding to a total of 9 BXs (Figure 6.14).

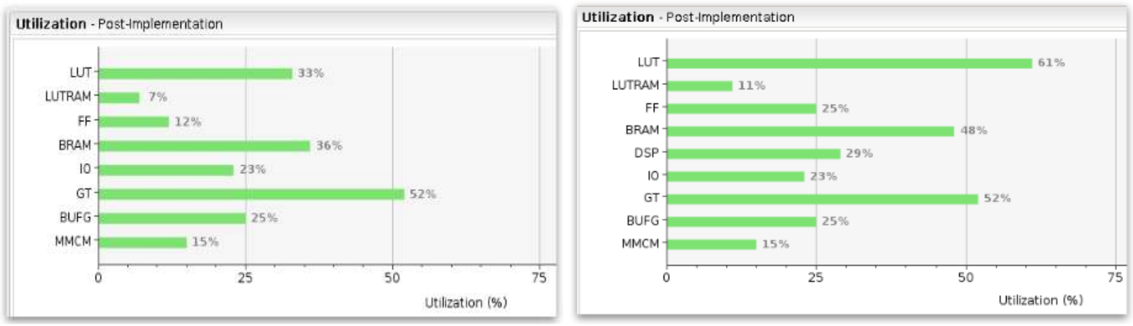


Figure 6.12: LEFT: The BMTF algorithm firmware utilization percentage, including the MP7 services. RIGHT: Firmware utilization percentage of the Kalman filter and the BMTF algorithms running in parallel, including the MP7 services.

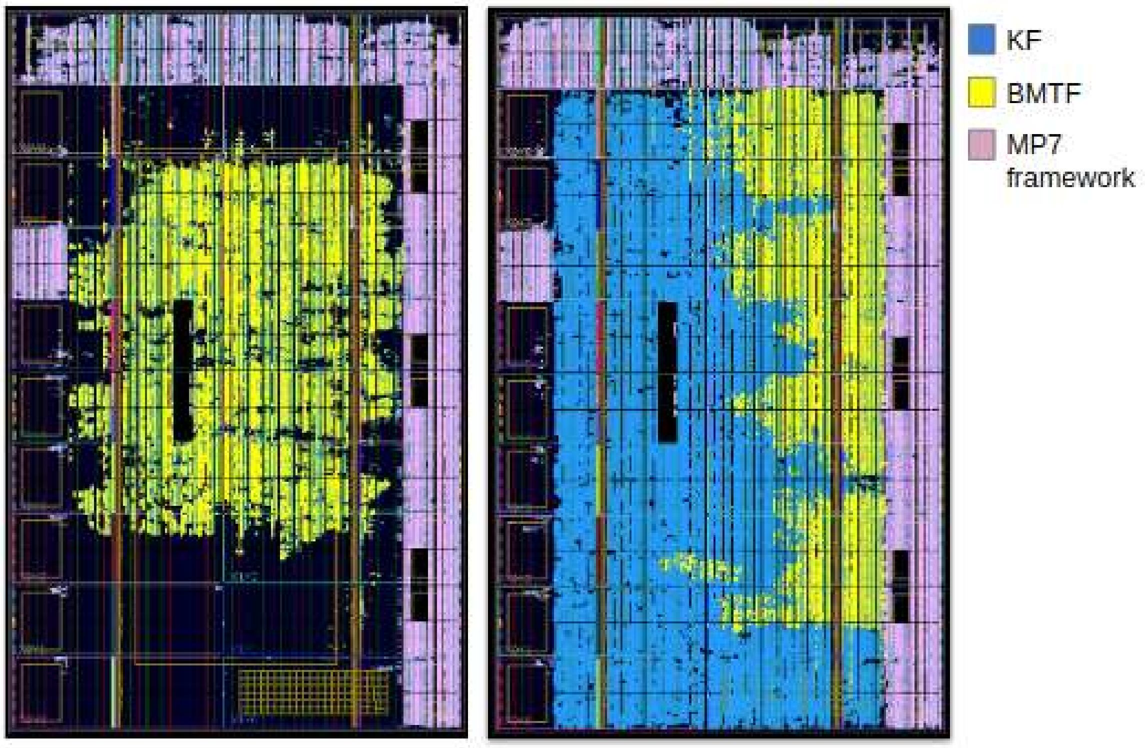


Figure 6.13: A comparison of the BMTF-only (left) and the BMTF+Kalman Filter (right) implemented designs, in the FPGA die of the MP7 board. The BMTF algorithm logic is shown in yellow, and the Kalman filter algorithm logic is shown in blue. The MP7 framework providing all additional services (DAQ, alignment, input-output buffers etc.) and links (10G asynchronous protocol, transceivers) is highlighted in light purple.

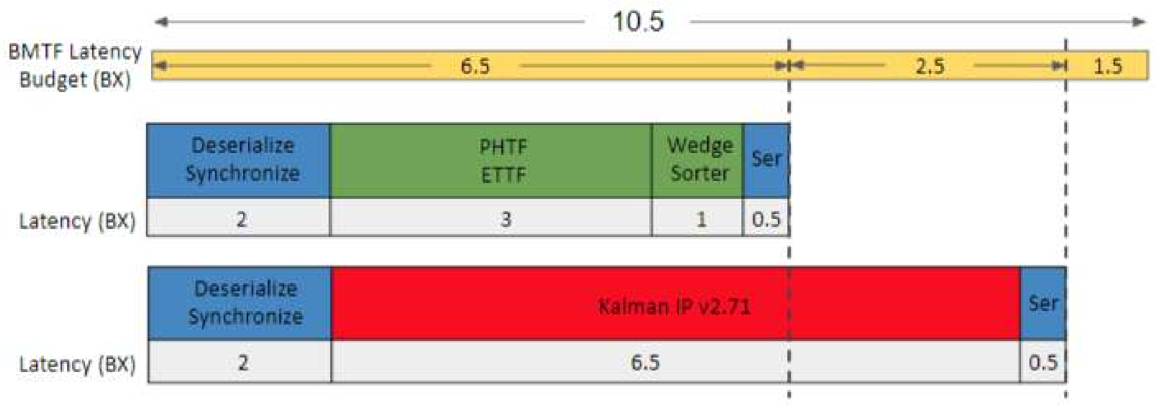


Figure 6.14: The Kalman filter algorithm (bottom) requires 9 BXs in total, that is 2.5 BXs more than the legacy one (top). However, the 4 BXs contingency of the legacy BMTF made it possible to trigger with the new algorithm.

6.2.5 Interface and timing considerations

The Kalman filter firmware is implemented in High Level Synthesis, and the final product is packed in an IP core. In order to instantiate the Kalman filter core, a wrapper module was written in VHDL (Figure 6.15). Special care was taken to correctly propagate the neighboring primitives, since the left-right wedges are reversed for the Kalman filter algorithm, due to its inwards logic. For example, in the BMTF legacy algorithm, for the N-wedge processor, we considered the N-1 wedge to be the left neighbor and the N+1 wedge the right neighbor. The opposite is true for the Kalman filter algorithm with N-1 wedge being the right neighbor, and the N+1 wedge being the left neighbor.

Furthermore, due to the Kalman filter algorithm's inwards logic, there is no need to split wheel 0 into "trigger wheel +0" and "trigger wheel -0". The source track segment is the one furthest in ϕ from the interaction point, and the one with the greatest $|\eta|$ value. This implies that the Kalman filter track address encoding will also be different compare to the legacy algorithm approach. The encoding of the Kalman filter track addresses can be seen in Figure 6.16, along with an example. In this example, the muon leaves a stub in all four stations, similarly to the example given in section 6.1.2. Taking into consideration the new algorithm's inwards logic, the corresponding track address would be *0x2022* for *ts1-tagged* track segments, and *0x1133* for *ts2-tagged* track segments. The first index number of the kalman filter track addresses corresponds to station 4.

Timing

One of the most challenging problems in FPGA designs is timing closure and meeting the timing requirements. All FPGAs have well defined time delays, to propagate inputs to outputs. The time

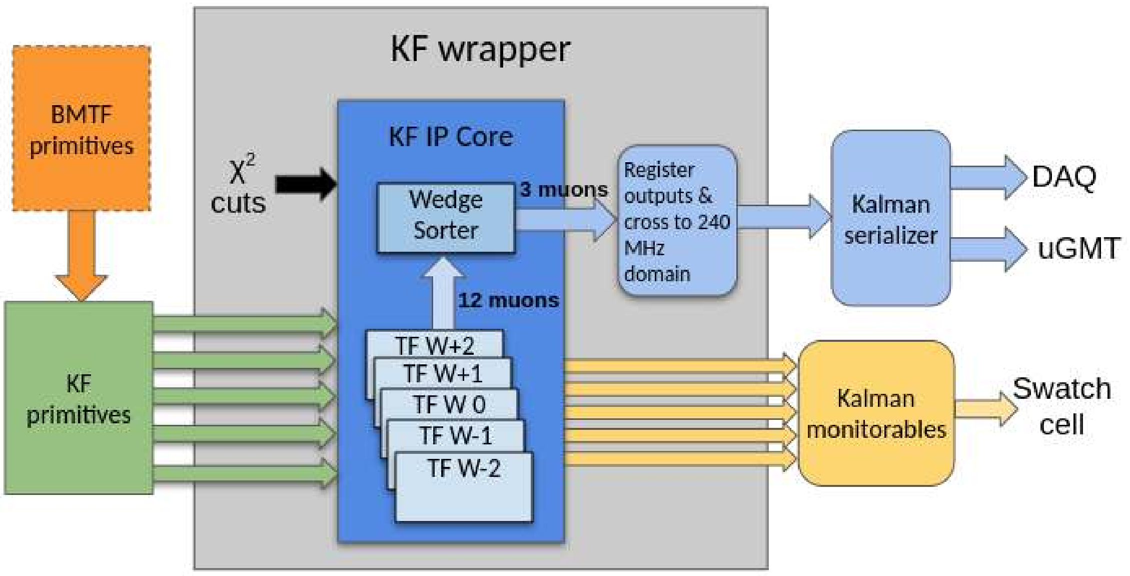
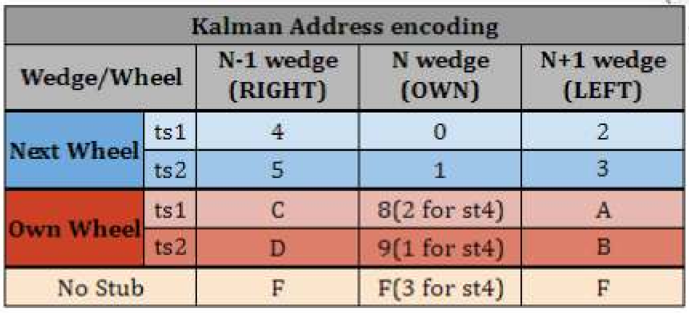


Figure 6.15: The HLS packs the final firmware in an IP core. A VHDL wrapper is required to instantiate the Kalman filter IP core, and synchronise its outputs.



Kalman Address encoding				
Wedge/Wheel		N-1 wedge (RIGHT)	N wedge (OWN)	N+1 wedge (LEFT)
Next Wheel	ts1	4	0	2
	ts2	5	1	3
Own Wheel	ts1	C	8(2 for st4)	A
	ts2	D	9(1 for st4)	B
No Stub		F	F(3 for st4)	F

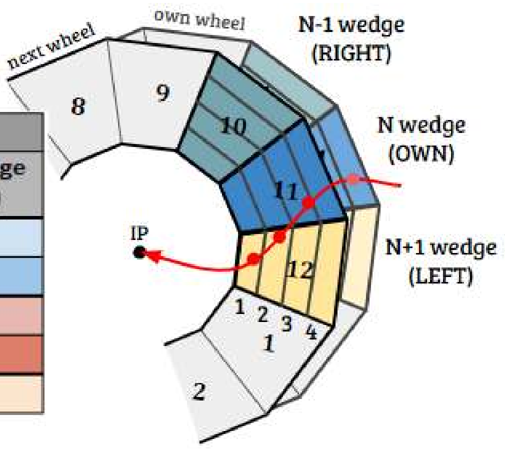


Figure 6.16: LEFT : The encoding format (in hex) of the Kalman filter muon track addresses. RIGHT : Example of a muon crossing own sector/own wheel station 4, own sector/own wheel station 3, left sector/own wheel station 2, left sector/next wheel station 1. For comparison, the legacy encoding can be seen in Figure 6.5.

delay along a path from the output of a synchronous element, through combinatorial logic gates, to the next synchronous element input, must be less than the time period between synchronizing clock pulses to the two sequential elements (i.e Flip Flops). In the case of the BMTF design, timing closure was achieved by the Xilinx Vivado tool, without the need of extra timing directives (*constraints*) or manual optimizations [69].

However, when integrating the new Kalman filter algorithm, the design utilized much more FPGA resources, resulting in less optimized placement and routing. Using the default strategies without extra timing directives, the design failed to satisfy timing closure.

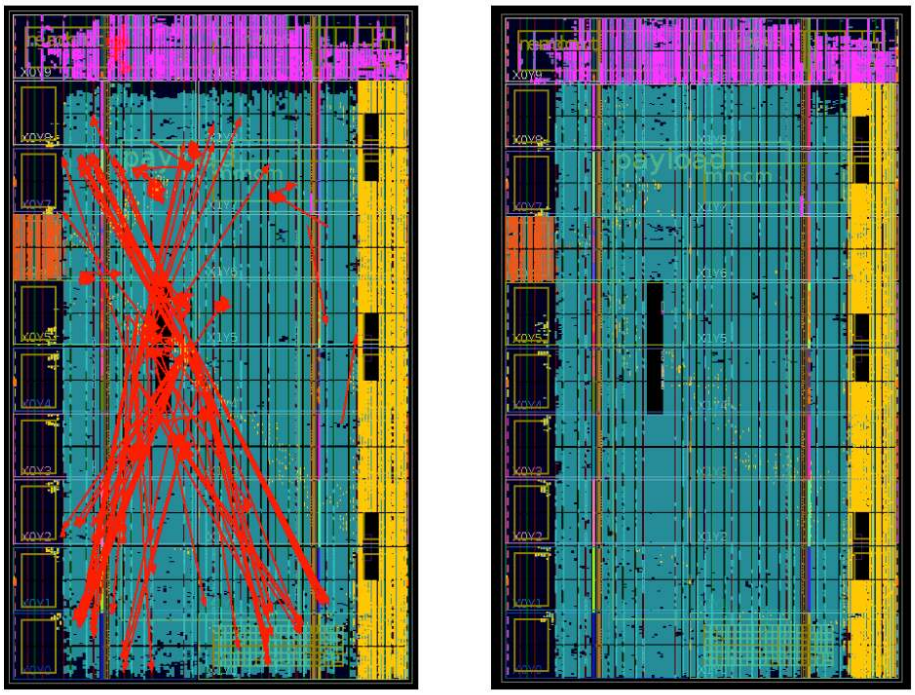
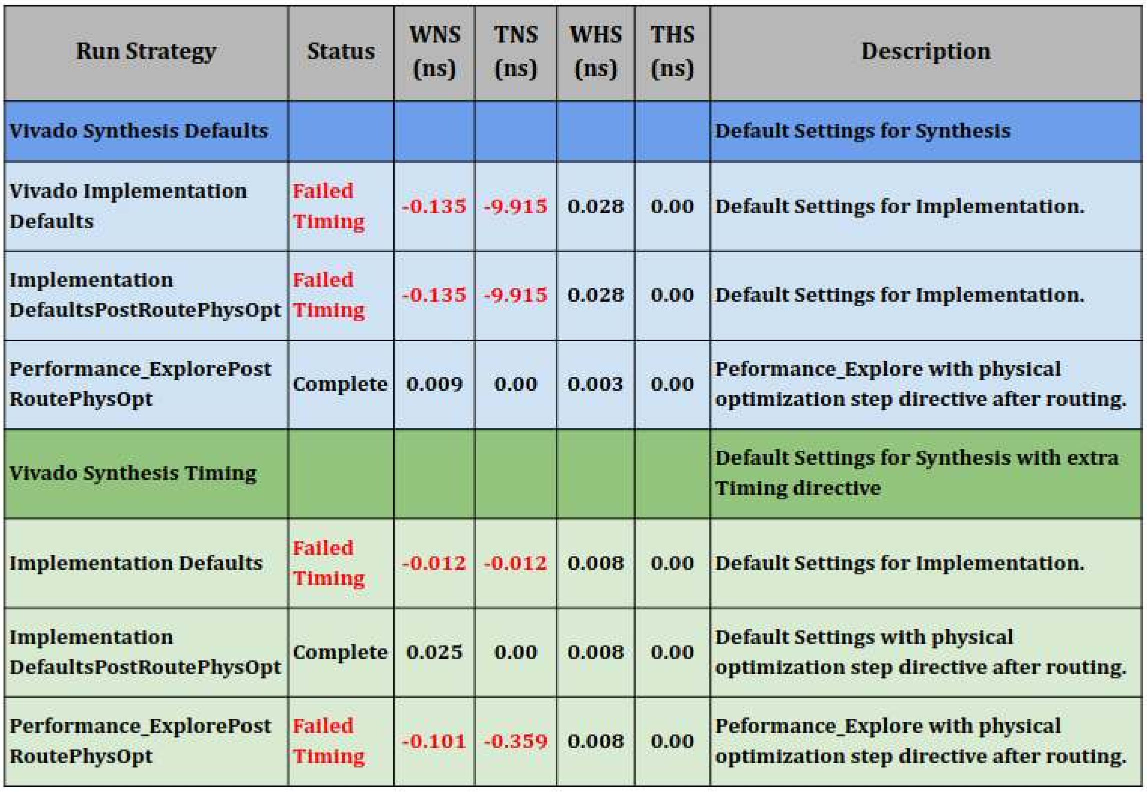


Figure 6.17: LEFT: The device view of the FPGA, after the implementation of both barrel muon algorithms. All the time critical paths that failed to meet timing constraints are shown in red. RIGHT: The device view, after the use of *Performance_Explore* strategy that uses multiple algorithms for optimization, placement, and routing, to get potentially better results, at the expense of increased implementation time. An extra command (*phys_opt_design*) was run in post-route mode, after the design was fully-routed, to add physical optimization on the design. No failing timing paths were found.

The most important critical data path in the design was crossing from the 160 MHz algorithm domain to the 240 MHz serialiser domain, which results in a maximum required time of 2 ns for the data propagation. In order to safely cross between the two clock domains, an extra constraint was added, instructing the placement tool to place the source and destination registers close together (see Figure 6.17). Furthermore, a combination of different strategies were tested, before finding the one giving the best timing. A list of the most frequently used strategies and their performance for the latest BMTF - Kalman filter firmware is depicted in Figure 6.18.

6.2.6 Kalman filter inputs

The input data to the Kalman filter are similar to the legacy data (Figure 6.3), however the following additions were required :



Run Strategy	Status	WNS (ns)	TNS (ns)	WHS (ns)	THS (ns)	Description
Vivado Synthesis Defaults						Default Settings for Synthesis
Vivado Implementation Defaults	Failed Timing	-0.135	-9.915	0.028	0.00	Default Settings for Implementation.
Implementation DefaultsPostRoutePhysOpt	Failed Timing	-0.135	-9.915	0.028	0.00	Default Settings for Implementation.
Performance_ExplorePost RoutePhysOpt	Complete	0.009	0.00	0.003	0.00	Peformance_Explore with physical optimization step directive after routing.
Vivado Synthesis Timing						Default Settings for Synthesis with extra Timing directive
Implementation Defaults	Failed Timing	-0.012	-0.012	0.008	0.00	Default Settings for Implementation.
Implementation DefaultsPostRoutePhysOpt	Complete	0.025	0.00	0.008	0.00	Default Settings with physical optimization step directive after routing.
Performance_ExplorePost RoutePhysOpt	Failed Timing	-0.101	-0.359	0.008	0.00	Peformance_Explore with physical optimization step directive after routing.

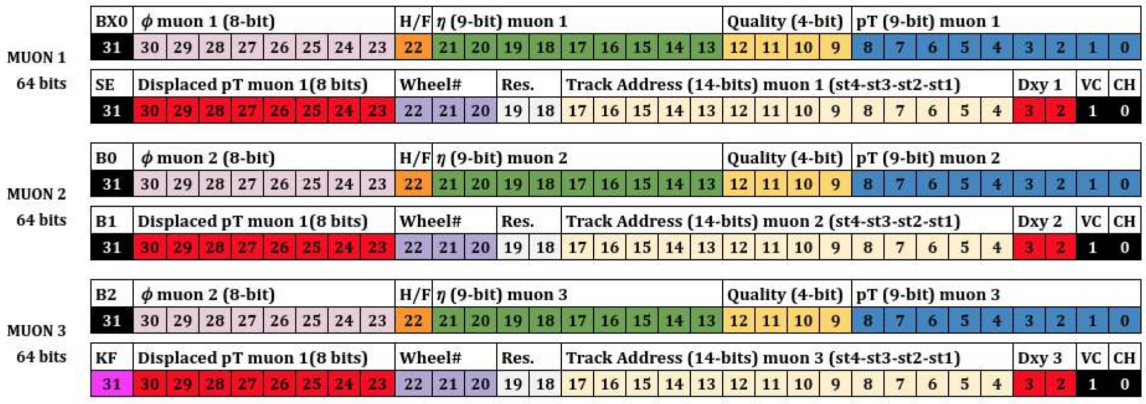
Figure 6.18: Different Vivado Synthesis and Implementation strategies that were used for the parallel BMTF - Kalman filter firmware to achieve timing closure. Both *Worst Negative Slack (WNS)* and *Worst Hold Slack (WHS)* must have positive values, for the routing to be considered successful. The *Total Negative Slack (TNS)* and *Total Hold Slack (THS)* values are the accumulated amounts of WNS and WHS respectively, for all failed routes, and must be zero.

- An additional kalman_valid bit was added to indicate the existence of a hit.
- In order to save one clock period of latency, each primitive is pre-assigned a Kalman filter track address code, according to the encoding shown in Figure 6.16.
- In the legacy algorithm, the bending angle in station 3 is not used, since it is always close to zero and was suppressed. However, it is propagated and used by the Kalman filter algorithm.

6.2.7 Kalman Filter outputs

The output muon data of the Kalman filter algorithm can be seen in Figure 6.19. The data format is the same as in the legacy algorithm case (see Figure 6.9), with the addition of the following quantities :

- The displaced transverse momentum, as calculated after reaching the first station, and prior to propagating the track to the vertex. The displaced p_T has the same range as the vertex constrained p_T, but lower resolution, since only 8 bits were available.
- The 2-bit $D_x y$, that provides a measurement of the distance of the displaced vertex from the primary vertex. Each unit corresponds to a 75 cm offset from the interaction point.
- One bit (bit 31 of the last 32-bit word) indicating which algorithm was used: "1" indicates the Kalman filter algorithm, and "0" the legacy algorithm.



	Field	Bits
MUON 1 64 bits	BX0	31
	ϕ muon 1 (8-bit)	30 29 28 27 26 25 24 23
	H/F	22
	η (9-bit) muon 1	21 20 19 18 17 16 15 14 13
	Quality (4-bit)	12 11 10 9
	pT (9-bit) muon 1	8 7 6 5 4 3 2 1 0
	SE	31
	Displaced pT muon 1(8 bits)	30 29 28 27 26 25 24 23
	Wheel#	22 21 20
	Res.	19 18
	Track Address (14-bits) muon 1 (st4-st3-st2-st1)	17 16 15 14 13 12 11 10 9 8 7 6 5 4
	Dxy 1	3 2
	VC	1
	CH	0
MUON 2 64 bits	B0	31
	ϕ muon 2 (8-bit)	30 29 28 27 26 25 24 23
	H/F	22
	η (9-bit) muon 2	21 20 19 18 17 16 15 14 13
	Quality (4-bit)	12 11 10 9
	pT (9-bit) muon 2	8 7 6 5 4 3 2 1 0
	B1	31
	Displaced pT muon 1(8 bits)	30 29 28 27 26 25 24 23
	Wheel#	22 21 20
	Res.	19 18
	Track Address (14-bits) muon 2 (st4-st3-st2-st1)	17 16 15 14 13 12 11 10 9 8 7 6 5 4
	Dxy 2	3 2
	VC	1
	CH	0
MUON 3 64 bits	B2	31
	ϕ muon 2 (8-bit)	30 29 28 27 26 25 24 23
	H/F	22
	η (9-bit) muon 3	21 20 19 18 17 16 15 14 13
	Quality (4-bit)	12 11 10 9
	pT (9-bit) muon 3	8 7 6 5 4 3 2 1 0
	KF	31
	Displaced pT muon 1(8 bits)	30 29 28 27 26 25 24 23
	Wheel#	22 21 20
	Res.	19 18
	Track Address (14-bits) muon 3 (st4-st3-st2-st1)	17 16 15 14 13 12 11 10 9 8 7 6 5 4
	Dxy 3	3 2
	VC	1
	CH	0

Figure 6.19: Muon data sent from BMTF to uGMT over one bunch crossing. The Kalman filter algorithm transmits at most 3 muons per bunch crossing to the uGMT, using 64-bits for each muon. Each row represents one 32 bit word, thus two rows represent one muon. Compared to the BMTF outputs, the displaced p_T and D_{xy} are the additional variables for each displaced muon.

6.2.8 The Kalman filter cancel-out logic in the uGMT

The cancel out of duplicate muons in the η direction is performed by the wedge sorter, in each BMTF processor, since it receives trigger segments from all 5 wheels of the detector. However, the cancel-out in the ϕ direction would require the presence of muons coming from the 2 neighboring wedges. For this reason, the cancel-out between BMTF possessors is carried-out by the uGMT. The uGMT uses two different cancel-out schemes, a coordinate based scheme for muons from two different systems (i.e. between BMTF and OMTF), and a track address based for inter-system cancel-out (i.e. neighboring BMTF wedges). While the coordinate based cancel-out is not affected by the use of the new algorithm, the track address cancel-out logic will not function properly with the Kalman filter track address encoding. As described previously, the Kalman filter algorithm's inwards approach results in the reversal of the neighbor wedges and the next wheel assignment. In Figure 6.20, a comparison of the expected track address encoding (left) and the actual encoding coming from the Kalman filter algorithm is shown. Furthermore, since wheel 0 was split in two, extra cancel-out logic was included to cancel-out inter-wheel 0 muons. This is not the case in the Kalman filter algorithm, for which this logic was removed.

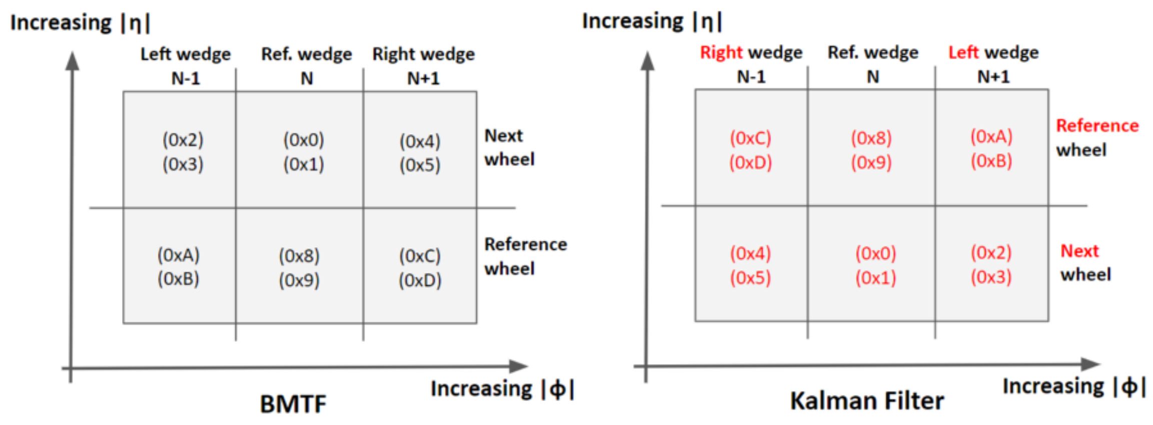


Figure 6.20: A comparison between the legacy BMTF track address encoding and the Kalman filter track address encoding.

Run 325113 Total Muons 9108	Number of Mismatches	Number of Mismatches/Total	Agreement %
No of muons	12	0.00132	99.87
p_T	6	0.00066	99.93
ϕ	26	0.00285	99.71
η	29	0.00318	99.68
displaced p_T	3	0.00033	99.96
D_{XY}	15	0.00165	99.83

Table 6.4: Kalman firmware versus Emulator comparisons in Heavy Ion collisions run 325113.

Taking into careful consideration all the changes in the track address scheme, the cancel-out logic of the uGMT was re-written, and the new firmware was tested during proton-proton collisions at P5. Figure 6.21 shows comparisons between the old and the new cancel-out logic for single and triple muon triggers that were affected. The old cancel-out logic was used only for runs 325103 - 325113, where there is a visible increase in the rate (inside the red box). The VHDL code of the updated cancel-out module is found in section A.3.

6.2.9 Firmware - Emulator comparisons

After achieving timing closure, and verifying its functionality in the lab, the new firmware was commissioned and tested with real data at P5. The latest version of the firmware was tested during the Heavy Ions collisions at the end of 2018. In Figure 6.22, comparison histograms between the Kalman firmware and the Kalman emulator are shown. The number of mismatches of the Kalman system, as compared with the Kalman emulator, is lower than 0.5%, indicating very good agreement. The data were taken from the heavy ion collisions run 325113. Detailed results can be seen in Table 6.4.

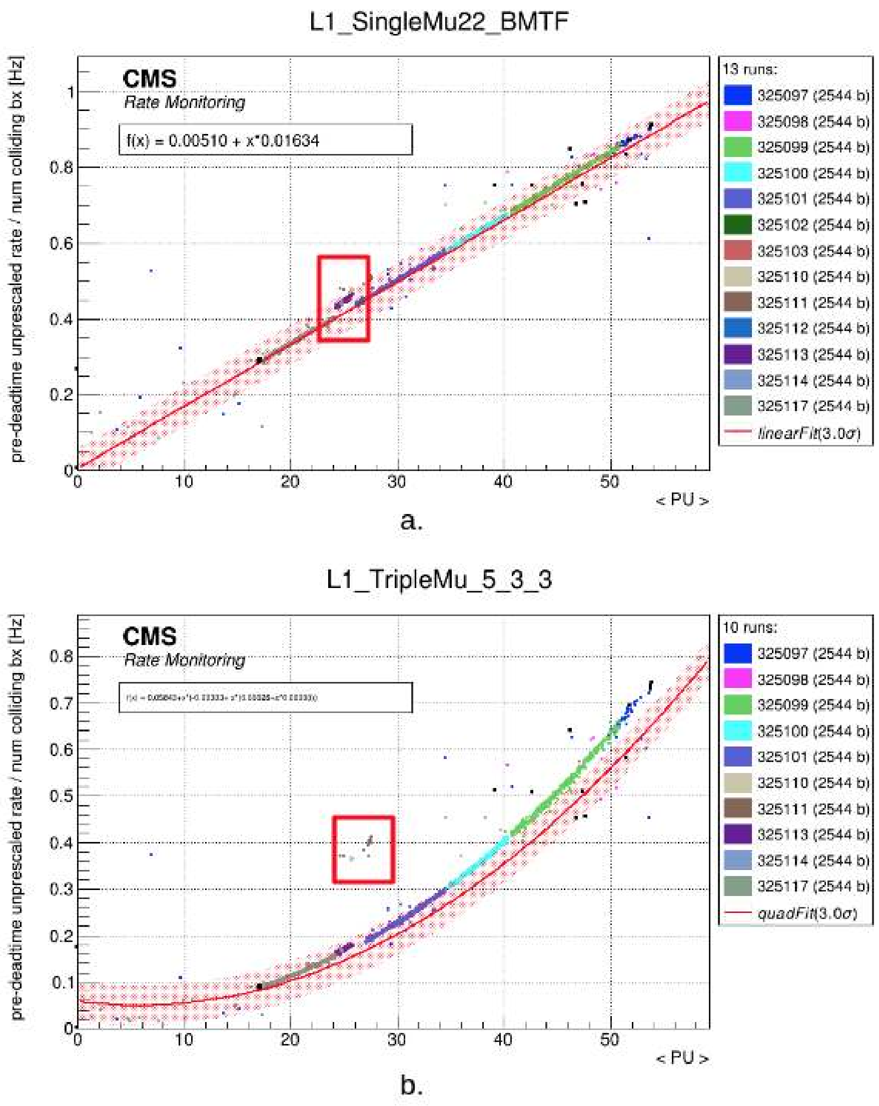


Figure 6.21: The uGMT unprescaled single (a) and triple (b) muon rates, taken from the CMS Web-Based Monitoring (WBM) platform. For runs 325103 - 325113, the old cancel-out scheme was used. The increase in the muon rate, due to duplicate muons, is visible in the red box.

6.2.10 Efficiency and rate of the new algorithm

The firmware and emulator comparisons performed, indicate that the offline emulator of the Kalman algorithm is in agreement with the firmware, and therefor this allows the use the offline emulator model, to calculate trigger rates and efficiencies of the new algorithm. The plots shown in Figure 6.23 demonstrate that the Kalman algorithm is more efficient compared to the legacy BMTF algorithm.

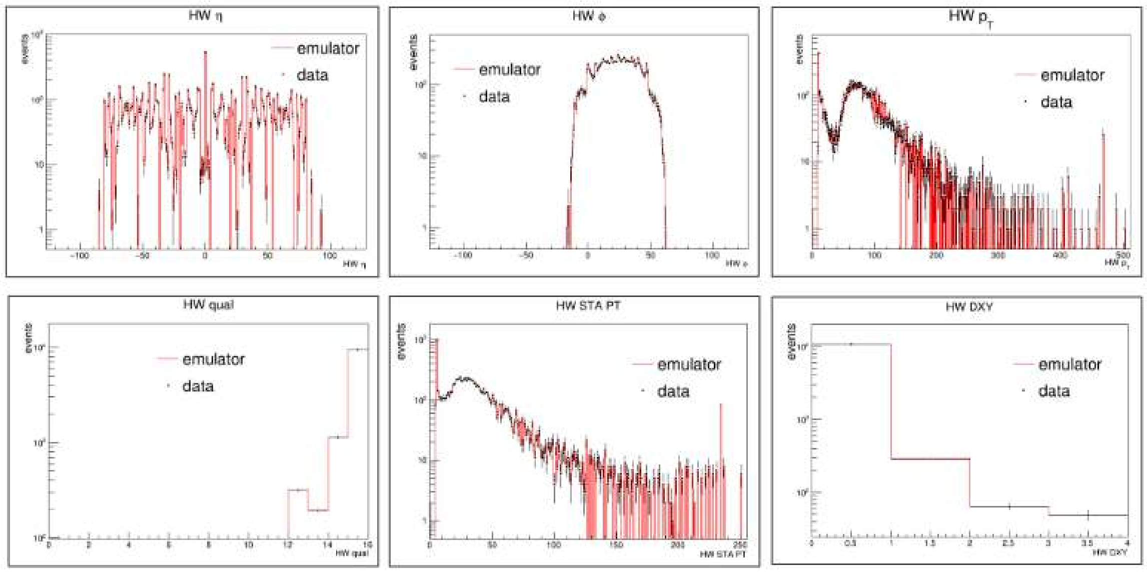


Figure 6.22: Kalman Muon properties in Cosmic runs. Hardware values are shown in black, while emulated values are shown in red. The results show 99.5% agreement, higher than the 99% agreement threshold inposed by the CMS trigger community.

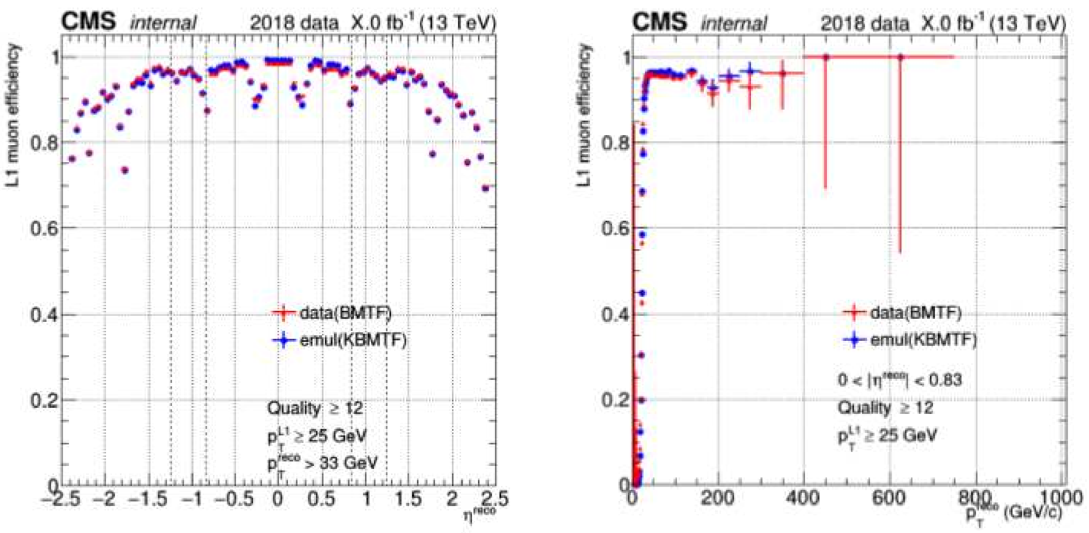


Figure 6.23: Efficiency comparisons between the legacy BMTF algorithm (red) and the Kalman algorithm (blue), as a function of η (left), and as a function of p_T (right).

Furthermore, as shown in Figure 6.24 (left), the rate, in the p_T region above 13 GeV/c (the CMS single muon threshold is equal to 22 GeV/c), is significantly lower than the legacy algorithm one. The most important improvement, however, resides in the intrinsic capability of the Kalman algorithm to detect displaced muons. This capability is obvious in Figure 6.24 (right), with the efficiency doubling or even tripling for muons originating from secondary vertexes 40 to 100 cm away from the primary vertex.

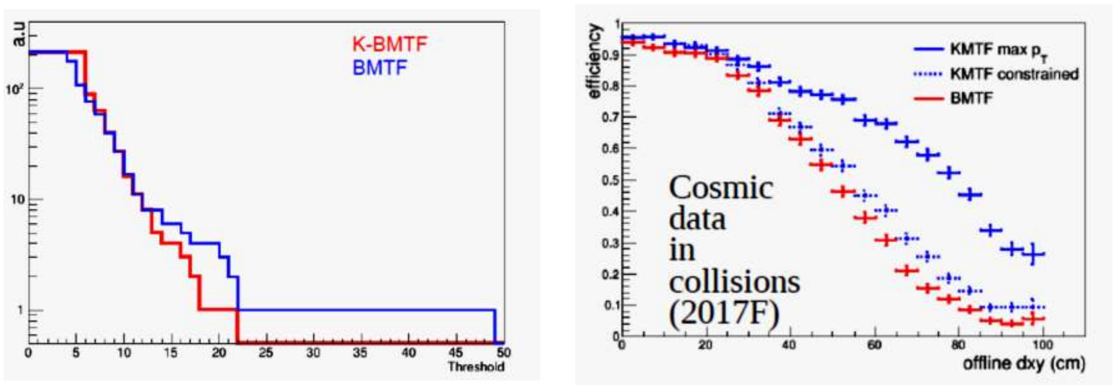


Figure 6.24: Comparisons between the legacy BMTF and Kalman algorithms. Rate (a.u.) as a function of the p_T threshold (left). Efficiency as a function of the impact parameter D_{XY} (right).

6.3 A demonstrator board for the Phase-2 Layer 1 Barrel Muon Trigger

Within the scope of Phase 2 R&D, a new Level-1 Trigger processor card was designed by the Greek CMS Trigger team, to provide a hardware environment for developing and evaluating new Level-1 trigger muon designs and technologies. The board comes with state-of-the-art fiber optics technologies, using micro-footprint optical interconnects. For testing purposes, a new firmware was developed, implementing asynchronous 16 Gbps GTH links. The links use the 64b/66b encoding scheme, with an overhead of 2 coding bits per 64 bits, considerably more efficient than the previously-used 8b/10b encoding scheme.

6.3.1 Architecture of the Barrel Muon Trigger System for Phase 2

The function of the muon trigger is to identify muon tracks in the experiment, and measure their momenta and other parameters, for use in the trigger menu. Within the Phase-1 muon detector system, the muons were identified and measured standalone. However, in Phase-2, the muons will be identified in conjunction with the Track Trigger, when matched to tracks in that system. Moreover, acceptance of muon-like particles from long-lived particle decays, displaced from the collision vertex, will be included, as well as heavy stable charged particles (HSCP), that reach the muon system with $\beta < 1$.

The standalone reconstruction of muons, within the muon detection system, will be segmented into three regions in η, as done in the Phase-1 system; a barrel muon track-finder (BMTF), an overlap muon track-finder (OMTF), and an endcap muon track-finder (EMTF). The output of these systems will be sent to a Global Muon Trigger (GMT) system, which will also receive tracks from the Track Trigger system. Because the firmware is anticipated to be relatively light, the BMTF algorithm will be consolidated onto the same boards as for the GMT, rather than implemented as a separate physical system like OMTF and EMTF. The GMT will match standalone muons to tracks from the Tracker; but to improve efficiency (such as in the gaps between muon detectors), the GMT algorithm will also match tracks to muon stubs. Therefore, the OMTF and EMTF will transmit trigger primitives, along with standalone tracks to the GMT.

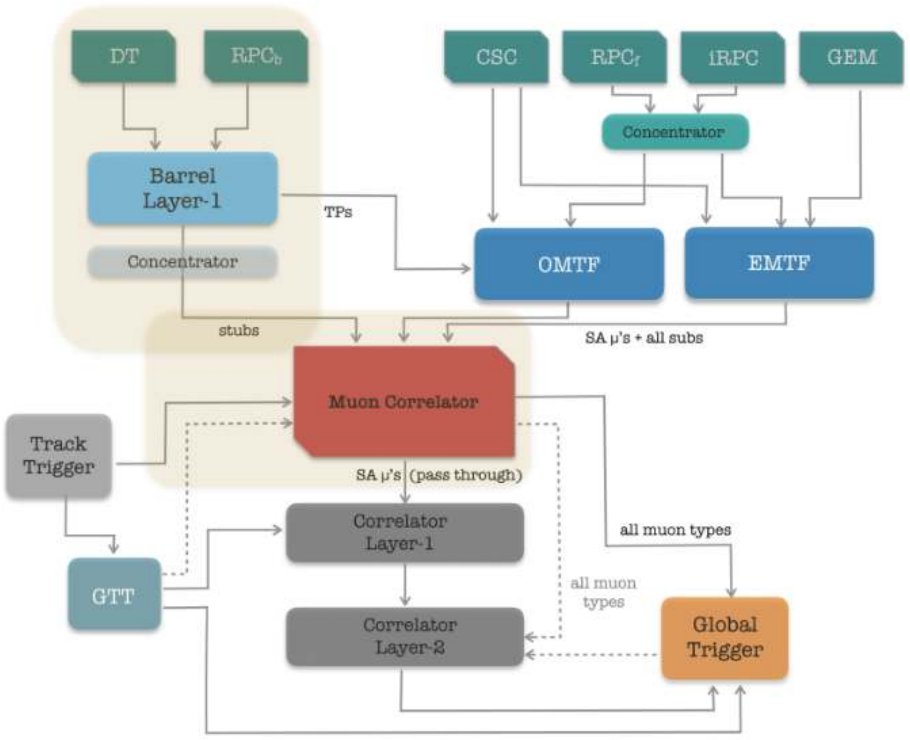


Figure 6.25: The Muon trigger architecture for Phase 2. The Barrel Muon Trigger is highlighted in yellow ([66].

The Barrel Muon Trigger for Phase-2 is divided in two layers. In Layer-1, the barrel DT and RPC hits will combine into "super-primitives". In Layer-2, the BMTF logic will reside within the GMT boards. The inputs to the BMTF come from Layer-1 of the barrel muon system, which is segmented into 60 sectors. If these 60 sectors are mapped onto 60 boards, with a single output fiber to each target, the number of links to each GMT board will need to be reduced via data concentration. Thus, 3-4 boards would be dedicated to the link concentration towards the GMT. If the 60 barrel sectors are mapped onto 30 output fibers to the BMTF target, the concentration layer is not strictly necessary. The link bandwidth will be 16 Gbps in either case. A diagram of the muon trigger architecture, with the barrel muon system highlighted is shown in Figure 6.25.

6.3.2 The hardware

The L1-BMT demonstrator board is powered by a Kintex UltraScale FPGA, providing 20 next-generation GTH transceivers, that reach speeds up to 16.3 Gbps. The board comes with state-of-the-art fiber optics technologies from Samtec. The high performance interconnect system uses active optical engines over 12 full-duplex channels, at data rates up to 16 Gbps. Furthermore, 4 FPGA transceivers are routed to a QSFP28 connector, allowing data rates of up to 28 Gbps per channel over 4 channels. In total, the board's 16x16 Gbps links add up to a total optical bandwidth of approximately 256 Gbps, in each direction, making it a high-performance all-optical data-stream processor (Figure 6.26). A Xilinx ZYNQ System-on-Chip device is used as the control interface, for the Kintex UltraScale FPGA. The system controller sets up or queries on-board resources, such as the power controllers and the programmable clocks.

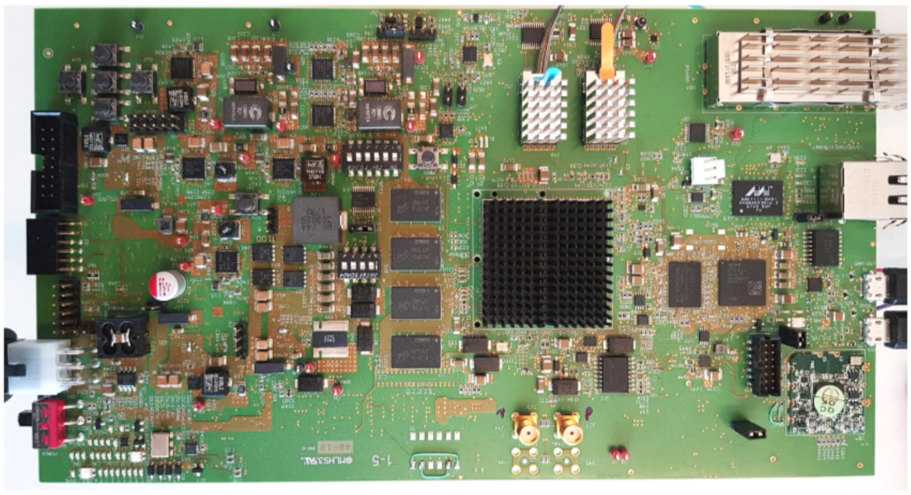


Figure 6.26: The Layer 1 Barrel Muon Trigger demonstrator board hosting a Xilinx KU040 FPGA located at the center of the board. Twelve transceivers are connected to the two FireFly modules located at the top center of the board. Four more GTH transceivers are connected to the QSFP28 module located at the top right corner. The four DDR4 RAM modules are placed near the left side of the FPGA and the ZYNQ XC7Z010 SoC controller is placed on the right side. The right half of the board is populated mainly by the power modules and the programmable clock generators.

The FPGA

The board has been designed to utilize the XCKU040 part, a mid-range Xilinx Kintex Ultrascale high-performance FPGA. It has a high number of DSPs, BRAM-to-logic blocks and next-generation transceivers. Combined with low-cost packaging, it enables an optimum blend of capability and cost. The part is available in a FFVA1156 package, with all the high-speed MGTs placed on the left side of the part. The ultrascale architecture provides key innovations like next generation routing, ASIC-like clocking and enhanced logic blocks, for a target of 90% utilization high-speed memory cascading, to remove bottlenecks in DSP and packet processing [70]. The board also includes 2 GB of DDR4 memory (four [256 Mb x 16] devices) at 1200 MHz / 2400 Mbps

Clocking

The board includes 3 low-jitter programmable clock sources (Figure 6.27). The GTH transceivers, connected to the high-speed Firefly modules, are clocked by a dedicated low jitter quad-clock generator (Si5338). A low-jitter frequency generator (Si570) is connected to the QSFP28 transceivers, and can also be used as a secondary clock source to the Firefly transceivers. A jitter attenuator (Si5328B), is used to reduce the jitter of a recovered clock received. A fixed frequency clock source (Si5335A) provides four frequencies (33.3 MHz, 90 MHz, 125 MHz and 300 MHz), that can be used as a free running clock for reset and initialization FSMs. Finally, an SMA external clock input is also available. All programmable clocks are accessed through a dedicated I^2C bus. The Si5328B and Si570 can be programmed using the ZYNQ system controller, and the Si5338 from the Kintex FPGA.

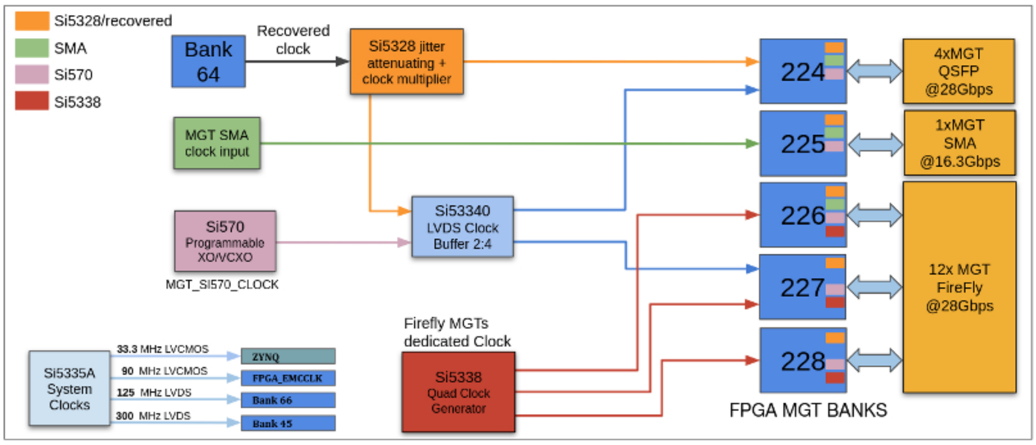


Figure 6.27: The clock distribution tree of the L1-BMT demonstrator board.

PCB

The board provides 32 high speed differential pairs, running at 16 Gbps. The Firefly transceivers were placed very close to the FPGA's right side, achieving closer proximity to the Banks, where all the MGTs reside, to simplify board layout and enhance signal integrity (Figure 6.28). For the substrate, the Panasonic Megtron-6 was chosen, due to its excellent high-frequency performance and impedance properties. A ground-plane has been placed between each layer containing high-speed traces, resulting in a 16-layer stack-up. All high speed differential pair's route lengths were matched using serpentine routing (Figure 6.28). Finally, to avoid additional signal distortion, caused by the plated-through hole (PTH) VIAs, the excess via stubs were removed, using a technique known as back-drilling.

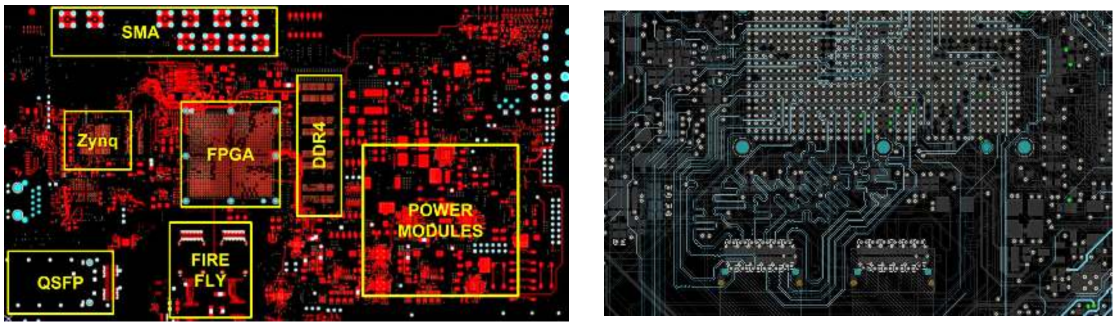


Figure 6.28: PCB details. LEFT: the TOP layer of the PCB with all the major board components. RIGHT: Detail of the PCB where the serpentine routing is visible.

High-Speed Optical Links

The data-interface consists of 16 optical links, operating in excess of 16 Gbps, making use of four of the MGTs available on the Kintex Ultrascale FPGA. Twelve of the optical links are routed to the FireFly optical flyover assembly (Figure 6.29), that is placed next to the FPGA. The FireFly configuration consists of 12 separate transmitter (TX) and receiver (RX) optical modules, joined in a "Y" configuration. These terminate to a single 24 fiber MPO connector. The connectors are placed mid-board, and the data "fly" over the PCB, allowing easier routing. Four more MGTs are routed to a QSFP28 (Finisar FTLC9551REPM) transceiver module. This module has a hot-pluggable QSFP28 form factor, and supports 103.1 Gbps of aggregate bit rate. However, the aggregate rate is limited by the MGT's maximum speed of 16 Gbps, to 64 Gbps.

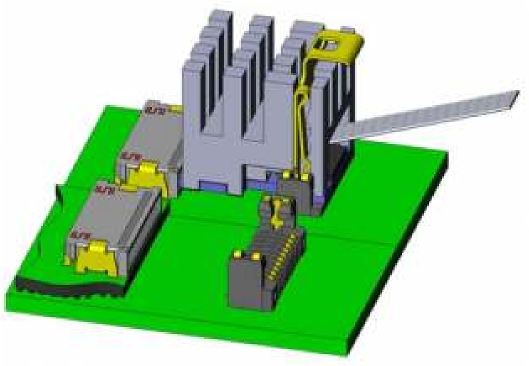


Figure 6.29: Miniature Patent Pending On-board Optical FireFly Micro Flyover System (source: Samtec).

6.3.3 Debugging and testing of the optical links

The FireFly optical links were tested using the integrated Vivado serial I/O analyzer (iBERT). A visual test method to study the Bit Error Ratio (BER), is by using an *eye scan*, a 2-D colour histogram, in which a voltage corresponding to the data signal from a receiver is repetitively sampled (Y-axis), while the data rate is used to trigger the horizontal sweep (X-axis). The colour index (logarithmic Z-axis) is the BER.

At high line rates, the received eyescan, measured on the printed circuit board, can appear to be completely *closed*, even though the internal eye, after the receiver equalizer, is open. The GTX/Y receiver eyescans provide a mechanism to measure and visualize the receiver eye margin, after the equalizer. The horizontal sampling position is determined by the Clock and data recovery function. The differential voltage, corresponding to the data, is indicated as data sample in Figure 6.30. To enable the GTH/Y eye scan functionality, an additional sampler is provided with programmable (horizontal and vertical) offsets from the data sample point. This is indicated as offset sample in Figure 6.30. A single eye scan measurement consists of accumulating the number of data sample (sample count) and the number of times that the offset sample disagreed with the data sample (error count) [71].

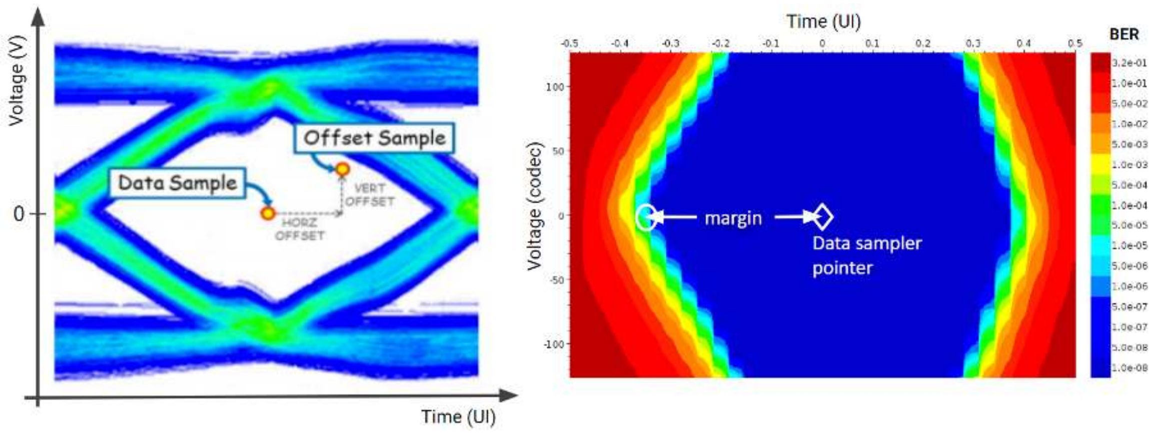


Figure 6.30: The eye scan counts all data errors continuously, over a certain period, for BER calculation.

In Figure 6.31 and Figure 6.32, the eyescans taken with the iBERT, for the 12 different optical channels, are shown for line-rates of 10 Gbps and 16 Gbps respectively. For the scans we used a 10 meter MTP cable, to create a 20 meter optical loopback path.

The QSFP links were tested using a copper loopback module and an optical loopback module. The tests revealed a problem in the routing; two, out of the four, transceivers had their transmitter outputs routed to the QSFP module receiver outputs, while the receiver inputs were connected to the QSFP module transmitter inputs. This problem resides in the PCB substrate, thus could not be fixed. The two remaining links showed very good performance at 16 Gbps, indicating that the routes were correctly matched. However, when using the optical module, the performance was

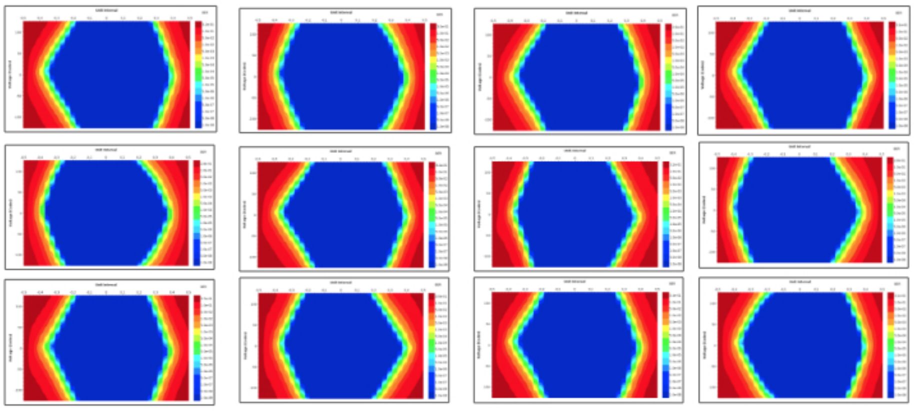


Figure 6.31: Eyescans for all twelve FireFly optical links at 10 Gbps.

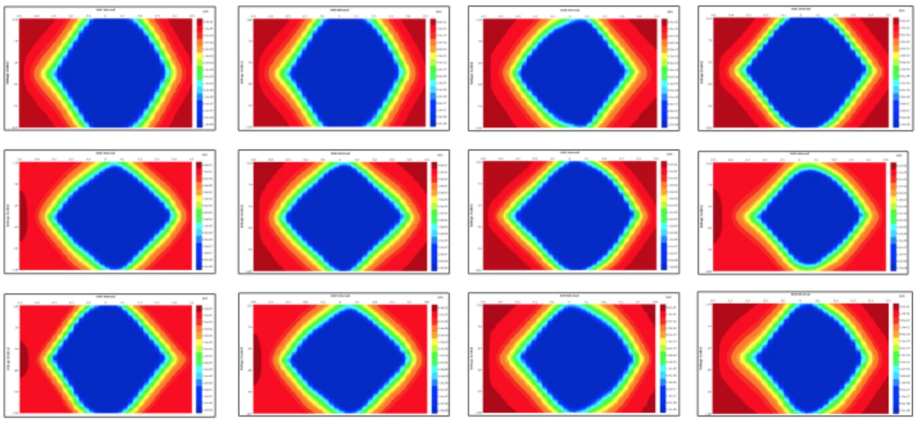


Figure 6.32: Eyescans for all twelve FireFly optical links at 16 Gbps.

poor, with BER of the order of 10^{-5}. This is due to the fact that this specific module is designed for 100 Gb Ethernet applications, and requires forward error correction software to improve the BER. Furthermore, its internal Clock and data recovery (CDR) circuit is locked at 28 Gbps, further degrading the performance of the module at other frequencies. The iBERT eyescans, for the two working links of the QSFP, can be seen in Figure 6.33.

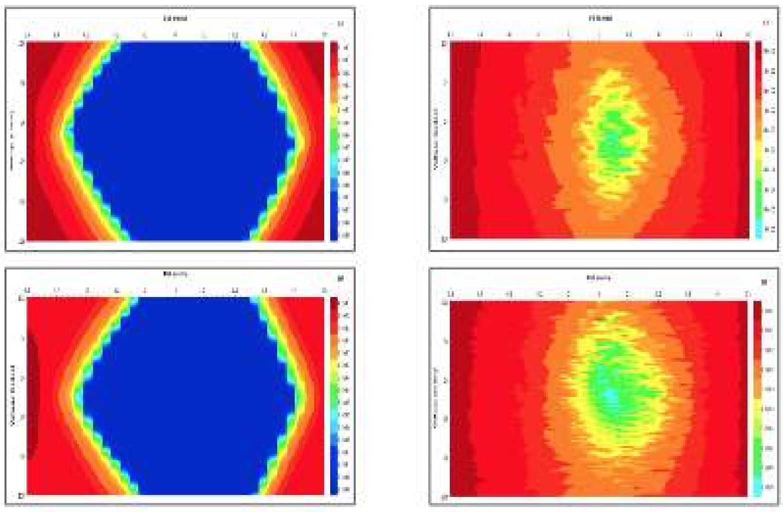


Figure 6.33: LEFT: Eyescans for the two working links of the QSFP, through a copper loopback module at 16 Gbps. RIGHT: Eyescans for the two working links of the QSFP, through an optical loopback module at 16 Gbps. The QSFP optical engine is optimized for 100Gb Ethernet, and its CDR is locked at 28 Gbps per channel. This results in very poor performance ($BER \sim 10^{-5}$) at all other frequencies.

Chapter 7

Hermes: A high speed link protocol for the HL-LHC CMS experiment

7.1 Introduction

The upgraded HL-LHC, after the third Long Shutdown (LS3), will provide an instantaneous luminosity of $7.5 \times 10^{34}\ cm^{-2}s^{-1}$ (levelled), at the price of a dramatic increase of the number of pileup interactions. It is generally expected, that the number of pileup interactions could reach 200 per bunch crossing. The upgraded detector will be read-out at an unprecedented data rate, of up to 50 Tb/s and an event rate of 750 kHz.

The CMS Trigger groups identified 16.000 Gbps and 25.78125 Gbps line rates, with 64b66b encoding, for wide use in Phase-2 trigger applications. The trigger protocol and the amount of physics payload, carried over the links, have important implications towards algorithm logic and algorithm developers. Within the scope of Phase-2 R&D, a new payload framing link protocol, along with the firmware infrastructure, was defined and developed. The protocol uses the 64b66b encoding scheme, with an overhead of 2 coding bits per 64 bits, that is considerably more efficient than the 8b10b encoding scheme. CRC error detection logic was also included. The design was named after the greek mythological god *"Hermes"*, who was the emissary and messenger of the gods [72].

7.2 The 10 Gbps asynchronous link protocol for the Phase-1 CMS trigger system

During Phase-1, all three CMS muon track finders received and transmitted information using a common asynchronous 9.6 Gb/s protocol Figure 5.10. This protocol was designed based on 10 Gb/s serial links, with 8b10b encoding [73], which gives a usable bandwidth of 8 Gb/s, and a payload of 32 bits at 250 MHz. The asynchronous protocol injects one additional idle character (0xF7F7F7F7), every 25 data frames, on the transmitter side. This character is rejected in the receiver side, and therefore reduces the 10 Gb/s link throughput to 9.6 Gb/s (see Figure 7.1).

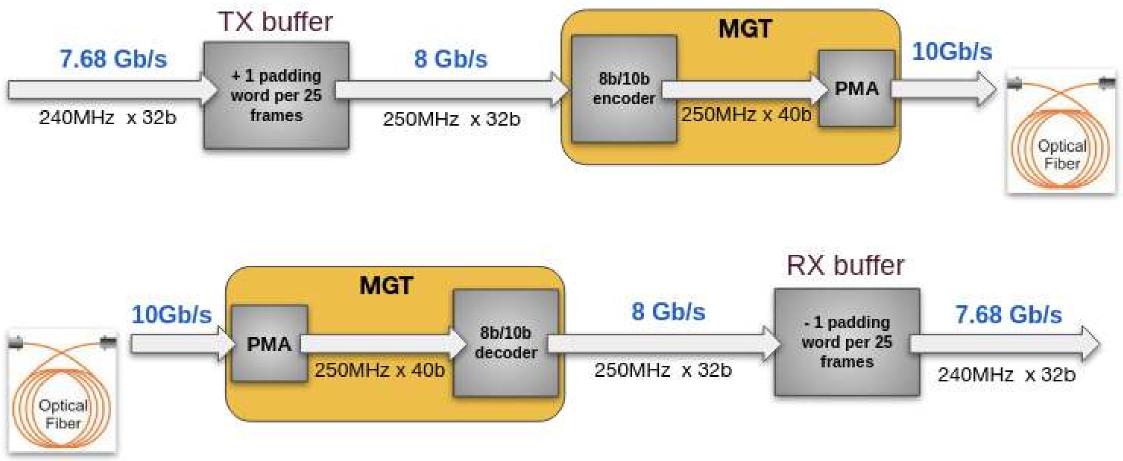


Figure 7.1: A simplified diagram of the asynchronous links used in the CMS trigger during Phase-1. Illustration of the transmitter (top) and receiver (bottom) logic and datapath. Data arrive from the processing units in 32-bit frames at 240 Mhz. The TX buffer writes the data to the 250 Mhz clock domain of the transceiver. The data are encoded to 8b10b codes, serialized and transmitted at 10 Gb/s. On the receiving end, the data are deserialized, decoded and cross to the 240 MHz processing clock domain, through the RX buffer.

The main advantage of using asynchronous links, is that the transceiver's reference clock and the LHC-driven logic clock are independent. Each board has its own differential oscillator clock source, which is very stable and ideal to use as reference clock for theMGTs, which are very sensitive to *jitter*[1] and phase changes. The unstable LHC clock is also received, and can be used to drive the processing logic, and to align and synchronise to the common read-out signal. Using a local stable clock to drive the transceivers, assures a stable link, running free of alignment losses, due to the clock phase uncertainties of the LHC clock. The data frames pass through the two clock domains (transceiver and processing), using the idle character and elastic buffers. This technique keeps the link stable, and also ensures a secure crossing over the data domains.

When no valid data words are scheduled for transmission, the data words are replaced with

[1]Jitter is the deviation from true periodicity of a presumably periodic signal.

a 8b10b coding word (0x505050BC), so that the 8b10b byte alignment can be performed. On the receiving end, the RX elastic buffer, apart from performing the clock bridging from the link clock domain, is also used to delay the data for link alignment and latency control. The alignment is achieved by sending an alignment marker at the start of a packet. Upon receipt of the align marker from the master link, all links are checked for alignment. If a link is not aligned, the RX buffer read pointer is incremented. This process is repeated, until all links are aligned. Lastly, a CRC checksum is injected at the end of a packet, for error checking [74].

7.3 Link requirements for the Phase-2 CMS trigger system

The CMS trigger system is tailored to suit the needs of the read-out and online event selection system of the CMS detector, making the high speed communication industrial standards not optimal for use. For this reason, a custom made firmware, implementing the interface between the trigger subsystems, is required. The link protocol, for the Phase-2 CMS trigger, must fulfill certain requirements in order to provide the best possible custom-tailored solution for the trigger systems. The main requirements are listed below.

Alignment and latency control

Each trigger subsystem will receive detector information through multiple serial links. A globally controlled mechanism that aligns all the links, and also controls the total latency of the system, must be included. This will result in a small increase in latency, but will make the system more flexible.

Asynchronous operation

The reference clock of the MGTs must be very stable, since the transceivers are very sensitive to jitter and phase changes of the "reference" clock that drives them. The LHC clock, that is distributed to all the trigger subsystem boards, is not stable, and is subject to many phase changes, making it not optimal to use as the reference clock for the MGTs. This is why an asynchronous approach is preferred, in which the transceivers are clocked by a stable, locally generated, clock. The processing logic is clocked by the LHC clock.

Low firmware resource footprint

The firmware, implementing the protocol of the high speed serial links, usually does not occupy a large area, since many of the functions are included in the MGT cores. However, since in most cases, every board includes tens of links, they can quickly occupy a significant percentage of the FPGA's resources. Even minor area optimizations per link can make major difference.

Error Detection

One of the most important features of a high speed link is the capability of error detection during transmission. The error detection mechanism must be able to detect either streams of errors or single errors. When a stream of errors is detected, an auto link recovery mechanism should be triggered. For single errors, an error counter must be included for monitoring. A Checksum or equivalent should be sent at least once per LHC orbit.

Automatic link recovery

During the LHC beam fills, CMS strives to maximise the effective time of quality data taking. The data transfer should be continuous and error-free. When a high speed serial link is lost, resetting would bring it back up. However, this takes a lot of time, which translates to dead-time for the experiment. A better solution would be to use an auto link recovery feature in the firmware, that eliminates the need for a global reset.

Low Latency

As discussed in section 5.1, the trigger decision time is limited by the the sheer size of tracker data, and the fact that they must be stored in pipeline memories of depth equal to the trigger latency. The trigger latency budget for Phase-1 was set to 160 BXs (4 μs), and is expected to increase to 500 BXs (12.5 μs) for Phase-2. However, the trigger system must be optimized to use as much of this latency budget as possible for new robust algorithms, while keeping the latency of the service tasks as low as possible. That is why it is important to strive for low-latency links (ideally equal or less than that of the Phase-1 link latency), while increasing the flexibility and bandwidth.

Low Bit Error Ratio

The Bit Error Ratio is the number of bit errors divided by the total number of transferred bits, during a time interval. In a communication channel, bit errors are bits of a data stream that have been altered due to noise, interference, distortion or bit synchronization errors. The BER limit for the CMS experiment is set to a maximum of 10^{-12}, which is the 16 G fiber channel standard [75]. During the development and testing of the links, however, we strive for at least 2 orders of magnitude lower BER, due to the optimal conditions of the testing fiber channels.

Metadata

Finally, the link protocol should reserve special words, to transmit metadata such as a Header, a Trailer, the Orbit Tag etc. During the development and testing of the new high speed links, all the above requirements were taken into consideration and addressed.

7.4 The *Hermes* asynchronous link protocol for the Phase-2 CMS trigger system

7.4.1 The 64b66b encoding

In high speed data transmission, the data stream is always encoded, to ensure enough data transitions, as required for DC balance and clock recovery at the receiver end. The 64b66b encoding is one of the most commonly used in data transmission, where bandwidth above 1 Gb/s is required. It was defined by the IEEE 802.3 working group , as part of the IEEE 802.3ae-2002 amendment. It replaced the 8b10b encoding in many modern high speed communication protocols, due to its reduced *overhead*[2]. The overhead of the 64b66b encoding is 2 coding bits for every 64 payload bits, or 3.125%. In comparison, the overhead of the 8b10b encoding is 20%, making the 64b66b encoding much more efficient (96.875% compared to 80% efficiency of the 8b10b encoding). The main design goals of the encoding were clock recovery, stream alignment, DC balance, transition density and small run-length[3]. A comparison of the 8b10b and the 64b66b encoding is found in Table 7.1.

	8b10b	**64b66b**
Run Length	5	Relies on Scrambler
DC balance	Excellent	Not guaranteed Demanding for receiver
Bit Synchronization Clock Recovery	Excellent	Relies on Scrambler, but at least one transition per 66 bits
Word Synchronization	“Comma” K-Characters	Sync-Header
Control Characters	K-Characters	Control Codes
Overhead	20%	3.125%

Table 7.1: Comparison of the 8b10b and 64b66b encoding.

The encoding is based on the addition of 2 bits of overhead for every 64 bits of payload, creating 66-bit block codes to transmit. All block codes have eight octets of data or control information preceded by a 2-bit sync header. *Data block* codes come with a *0b01* sync header, while *control block* codes have a *0b10* sync header. The *data block* contains 64 bits of data information, while the *control block* contain an 8-bit word to specify the type of the block (Figure 7.2). Sync

[2]The ratio of the number of added coding bits to the number of raw payload bits.
[3]Run-lenght is the number of concecutive 0s or 1s.

header values *0b11* and *0b00* are illegal and must be flagged as *soft errors*[4] whenever they appear. The use of the *0b01* and *0b10* sync headers was selected to guarantee at least one bit transition every 66 bits, which means that a run length of more than 65 cannot represent valid data. It also allows easier clock/timer synchronization, as a transition will be seen every 66 bits.

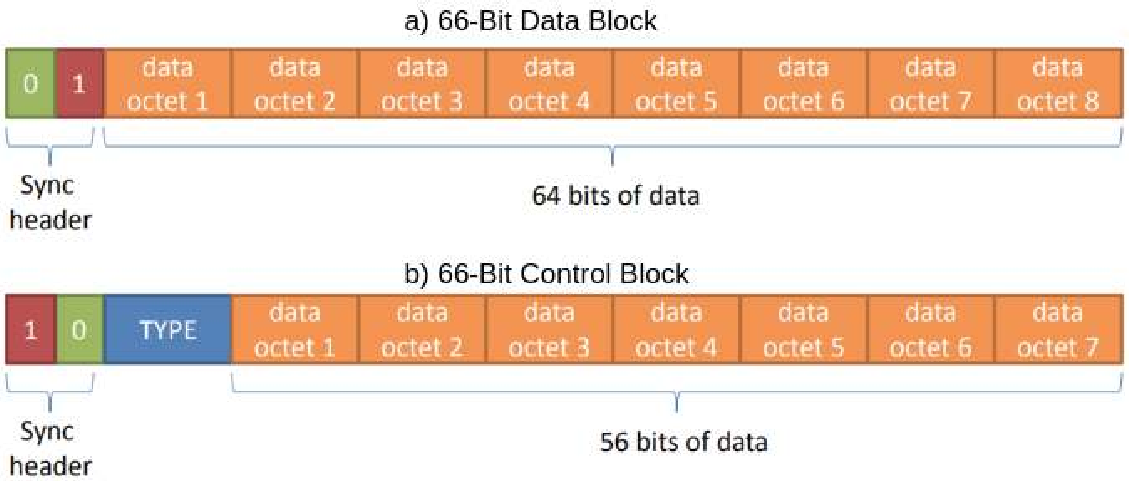


Figure 7.2: The structure of the data and control blocks of the 64b66b encoding. When the control sync bits are *0b01*, then the 64 bits of payload represent data. When the control sync bits are *0b10*, this indicates an incoming control block, with the first 8 bits defining the "Block Type".

Whenever a lane transmits a block code, the 8 octets, following the 2-bit sync header, are scrambled using a self-synchronous scrambler function. The 2-bit sync header is not scrambled. The addition of a scrambler is not intended to encrypt the data, but to ensure a relatively even distribution of 1s and 0s in the transmission line, by mixing up the data. Although the scrambler cannot guarantee that the output data will not have a long run-length, it ensures that a bit-error due to long run-lengths is extremely unlikely. It is theoretically possible for a random data pattern to align with the scrambler state and produce a long run of 64 zeroes or ones, but the probability of such an event is equal to flipping a fair coin and having it come up in the same state 64 times in a row. At 16 Gbps line rate, the expected Mean Time To Failure (MTTF) of a 66-bit block with a 65-bit run-length, assuming random data, is approximately one every 1200 years:

$$MTTF_{random} = \frac{(66bits) \times (2^{64})}{2 \times 16 \times 10^{9} bits \times sec^{-1}} = 1206.4\ years \tag{7.1}$$

Furthermore, the scrambler's output approximates a sequence of random binary bits. Passing such a sequence through an AC-coupled circuit produces a baseline wander noise that follows a Gaussian distribution, and the impact on the system error rate can be statistically quantified. In practice, a modest coupling capacitor value of $1\ nF$ in a 100Ω system, is sufficient to guarantee that a DC drift of more than 2.5% will occur less often than once per 10^{22} bits [76].

[4]Soft errors are transient, statistical errors, expected in the normal course of operation.

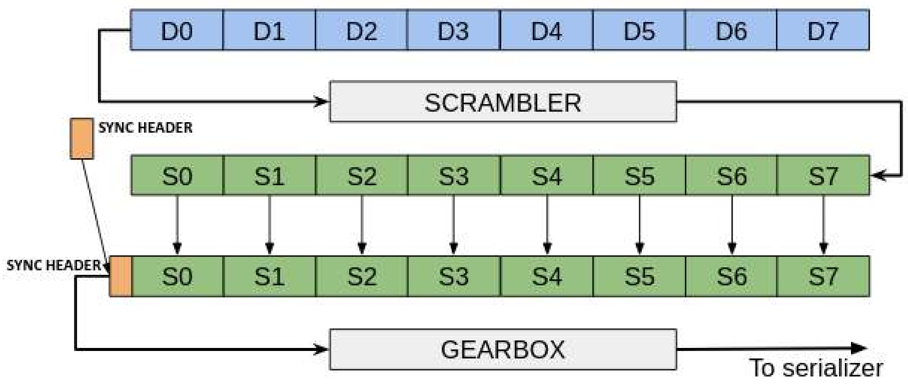


Figure 7.3: 64b66b encoding transmit chain and bit ordering.

Finally, the scrambled data and the sync header are pushed into the transceiver, where the 66-bit blocks are packed by the gearbox into 32-bit data units, and sent to PMA for serialization and transmission (Figure 7.3).

7.4.2 The *Aurora* protocol

The "Aurora 64b66" is an open protocol standard, developed by Xilinx. It is scalable, lightweight and can be used to move data point-to-point across one or more high-speed serial lanes. It provides a simple serial interface for engineers, looking for cost-optimized system connectivity solutions. A small description of the protocol is presented in this section, since the CMS *Hermes* link protocol, described below, is loosely based on the Aurora protocol.

The Aurora protocol describes the transfer of user data across an Aurora channel, consisting of one or more Aurora lanes. Each Aurora lane is a serial data connection, either full-duplex or simplex. The Aurora interface is not defined in the specification, and can be decided independently for each implementation of the protocol.

Aurora channels have the following properties:

- Data is transferred through the Aurora channel in frames.
- Frames share the channel with control information, such as flow control messages, clock compensation sequences and idles.
- Frames can be of any length, and can have any format. Only the delineation of frames is defined in the specification.
- Frames in Aurora do not have to be contiguous, and they can be interrupted at any time by flow control messages or idles.

- There is no gap required between frames in Aurora.

All transmissions in Aurora 64b66b are performed using ten types of 64-bit block codes. Only one block can be transmitted through a lane per cycle, and the blocks are prioritized, meaning that when two or more blocks need to be sent on the same cycle, on the same lane in a channel, the highest priority block is always selected. Table 7.2 summarizes the blocks available in Aurora 64b66b and their priority level.

The initialization procedure starts with each lane preparing to transmit and receive data. When ready, blocks are transmitting, while attempting to find the block boundaries of the incoming data, and align to them. The bit alignment of the incoming blocks is based on the block sync state machine, as defined in the IEEE 802.3 Standard [77]. The Aurora interface stays in the lane initialization state, until the block sync state machine receives 64 error-less blocks. A mechanism is included to reset and re-initialize the channel, if a number of errors are received in a row.

The Aurora transmit procedure starts by delineating the frames using *Separator* and *Separator-7* blocks. The next step is to encode the blocks according to the 64b66b coding, and finally the data stream is serialized and transmitted through the channel. On the receiving end, the Aurora interface first performs data de-serialization, then the 64b66b blocks are decoded, and finally all the Control blocks are stripped from the data stream. Only the data octets from the Data blocks and *Separator* blocks are delivered to the user interface as a frame. The protocol supports both simplex and full-duplex serial data connections. In full-duplex mode, Aurora interfaces can transmit and receive data from the Aurora channel. The channel control logic performs the same initialization procedure as for a simplex lane, but in both directions. Additionally, control information is used to allow each Aurora interface to detect whether the other side is prepared to receive data. In simplex mode, each interface uses a single lane, to transmit or receive from the Aurora channel. Channel control in each interface initializes the channel, passing control to the user application.

In duplex mode, the Aurora protocol also supports a flow control mechanism, allowing receivers to request from their channel partner to transmit idles instead of data. The transmitting side can issue a pause of normal data transmitting, and request up to 256 idle characters to be sent. The normal data transmitting operation continues, after all flow control messages have been sent.

Finally, the Aurora protocol includes an error handling mechanism, that defines two types of errors: hard and soft. Hard errors, such as channel disconnection, buffer overflow, or hardware failure, are irrecoverable. Soft errors are statistical errors, expected in the normal course of operation.

Aurora interfaces respond to hard errors by resetting. Soft errors do not cause a reset, but can be reported to the user application. Illegal sync header values and illegal block type field values in a control block are considered soft errors. A 32-bit CRC code is also used for error detection.

A more detailed description of the Aurora 64b66b protocol can be found in Xilinx's "Aurora 64b66b Protocol Specification" [78].

Block	**Description**	**Priority**
Clock Compensation	A special idle block that can be deleted or replicated by the Aurora interface that receives it. Clock Compensation blocks are only used for asynchronous channels.	1 (Highest)
Not Ready	Only available in Full-duplex mode, it is sent while attempting to align data from the channel and perform channel bonding.	2
Channel Bonding	They are send simutanusly and are used by the receiver to adjust their channel bonding FIFOs to receive these blocks simultaneously, thus correcting for skew between the lanes.	3
Native Flow Control	This block requests native flow control from the Aurora interface on the other side of the channel.	4
User Flow Control	This block is used to start a user flow control message and contains a field indicating how many octets that follow are a part of the message (up to 256).	5
User K-Blocks	User K-blocks are blocks that can be used to implement application-specific control functions and they are passed directly to the user. There are nine available User K-Blocks.	6
Data, Separator, Separator-7	These blocks are carrying user data. Data blocks carry eight octets of data. Separator blocks carry 0 to 6 octets of data and Separator-7 blocks are the same as Separator blocks, but always carry exactly seven valid octets of data.	7
Idle	Idle blocks are the lowest priority blocks and are transmitted whenever a higher priority block is not available for the channel. Their main function is to keep the link "alive" when nothing else is transmitted.	8 (Lowest)

Table 7.2: Aurora Code Blocks

7.4.3 Overview of the *Hermes* protocol

The *Hermes* link firmware is a lightweight, link-layer protocol, that can be used to move data point-to-point, across one or more high-speed serial lanes. It was developed to meet the needs of the CMS Level-1 trigger upgrade, and supports simplex operation with continuous data transfer. It will be used to transfer data between the trigger subsystems, at high speed, using one or many GTH or GTY transceivers. The links are asynchronous, meaning that the main algorithmic logic is clocked in a lower frequency, compared to the one of the link clock, allowing more flexibility when selecting the logic clock. This is achieved by using asynchronous FIFOs, in both the receiving and transmitting ends. To compensate for the frequency difference, padding words are being injected on the transmitting end, and are stripped away on the receiving end.

The link initialization and error handling are also based on the insertion and checking of the padding words. After initialization, applications can pass data across the channel, as frames or streams of data. The top level entity provides access to the serial links in groups of 4 links, called "quads". Data to or from a quad is presented in the form of an array of links, of type *ldata*, that is made up of the record type *lword*, containing a 64 bit data word and a data valid signal. When the valid signal, coming from the processing units, is set to "0", the protocol automatically sends idle characters in order to maintain lock and prevent excessive electromagnetic interference. The protocol was tested at a 16 Gbps line rate, with the local clock running at 240 MHz, and the link clock at 250 MHz.

The proposed asynchronous 64b66b protocol uses the standard 64b66b encoding, which is the preferred encoding scheme for 10 Gigabit Ethernet, making it compatible with many existing hardware standards for cables and back-planes. The 64b66b encoding transforms 64-bit data to 66-bit line code, to provide enough state changes. This allows reasonable clock recovery and alignment of the data stream at the receiver. The protocol overhead of 64b66b encoding is 2 coding bits for every 64 payload bits, or 3.125%. This makes the encoding considerably more efficient, compared to the 20% overhead of the previously-used 8b10b encoding scheme, which added 2 coding bits to every 8 payload bits (more details in subsection 7.4.1).

7.4.4 The protocol interface

The top level entity of the firmware provides the connectivity to the processing units and the control software. The interface is shown in Figure 7.5, and a detailed description of the input and output ports and their functionalities is given below.

TX_DATA

TX_data is the input port of the data coming from the processing units. It is an array of links of type *ldata*. *ldata* is made up of the record type *lword*, that contains a 64-bit data word, the data

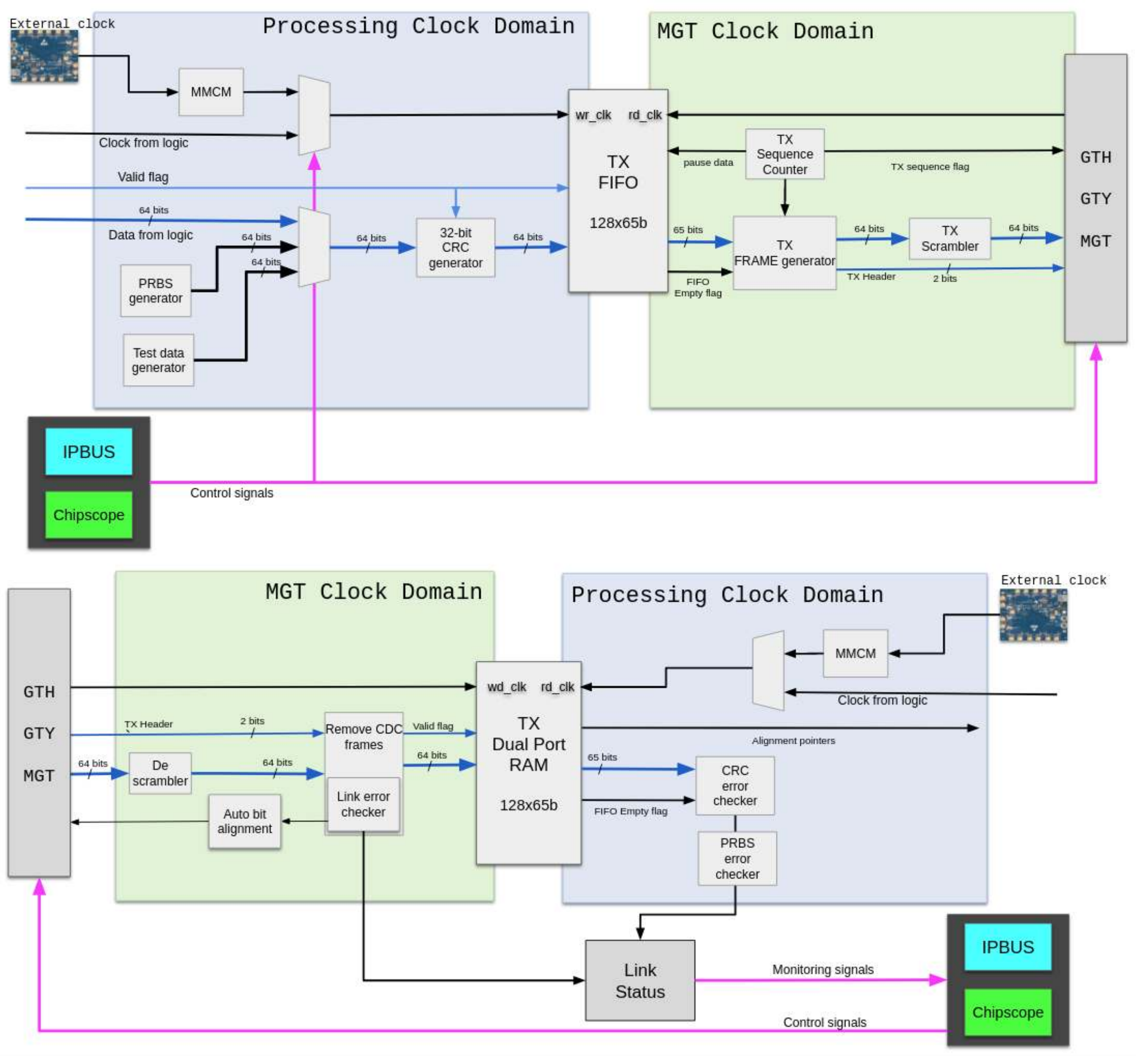


Figure 7.4: TOP: Overview of the *Hermes* framing protocol transmitter design. The TX FIFO is instantiated, to cross from the processing clock domain, to the transceiver clock domain. The frame generator injects the filler words, when the empty flag of the FIFO is asserted, and the SoF and EoF coding words, at the start and end of a data stream, respectively. BOTTOM: Overview of the *Hermes* framing protocol receiver design. The alignment and error checker module scans the incoming coding words, and the 2-bit header, for illegal values. It aligns the data words, by shifting them by one bit at a time, until no continuous errors are detected. The data validity is ensured, by generation and periodic transmission of a 32-bit CRC word. The filler words are stripped, and the RX FIFO ensures the safe crossing to the processing clock domain.

valid signal and two control signals, the start and strobe signals. The number of the channels, to be instantiated, defines the range of the array. Each *lword* corresponds to a channel.

```
-- VHDL code
type lword is
```

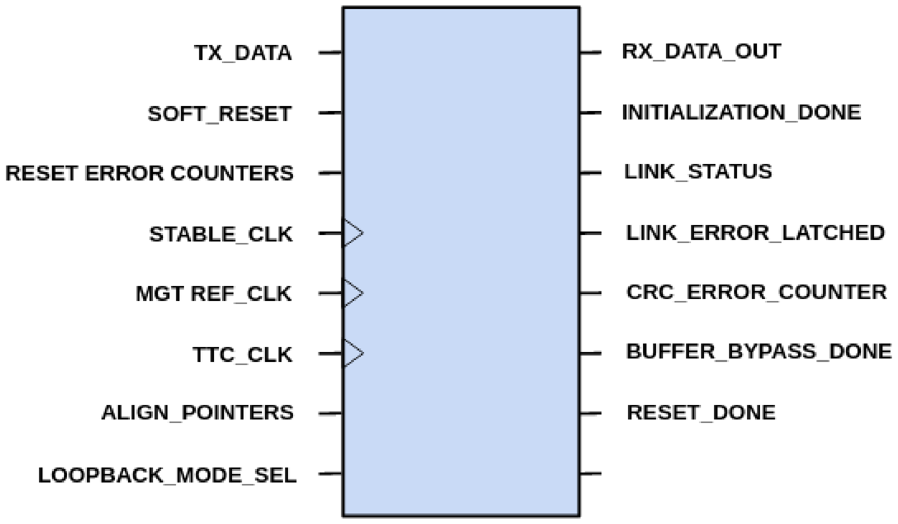


Figure 7.5: The top level entity of the firmware, that provides the connectivity to the processing units and the control software.

```
record
    data  : std_logic_vector(LWORD_WIDTH - 1 downto 0);
    valid : std_logic;
    start : std_logic;
    strobe: std_logic;
end record;
```

The data element of the record is the main data bus, with a width of 64 bits. The valid element is driven by the processing logic, and is set to “1” when the data bus is filled with data, and “0” when the data bus is empty or filled with non useful information. The start and strobe signals are used from the IPBus software.

The user can select the size of the array and the data bus width (only 64 bits for the *Hermes* protocol), by setting the value of the *N_CHANNEL* and *LWORD_WIDTH* respectively, when instantiating the top module.

RESET_ERROR_COUNTERS

In order to monitor the functionality of the links, a number of error counters have been added. One 8-bit counter counts the number of header and code block errors, for each channel, and a second 8-bit counter counts the number of CRC errors. The RESET_ERROR_COUNTERS port is controlled by the IPBus software, and it resets all error counters to “0”. It is important to reset the error counters after every (re)initialization procedure.

ALIGN_POINTERS

The ALIGN_POINTERS are comprised of 3 signals for each channel, that are used to align all the links by controlling the RX buffer. The 3 signals are:

- The *buf_ptr_inc_in*, that is used to increment the RX buffer read pointer.
- The *buf_ptr_dec_in*, that is used to decrement the RX buffer read pointer.
- The *buf_rst_in*, that is used to reset the RX buffer read and write pointers.

An alignment marker is sent to the master channel when no valid data is transmitted. Upon receipt of the alignment marker signal, from the master link, all links are checked for alignment. If a link is not aligned, the buffer read pointer is incremented. This process is repeated until all links are aligned.

LOOPBACK_MODE_SEL

The *LOOPBACK_MODE_SEL* is a 3-bit input port that selects one of the four loopback modes of the transceiver. Loopback modes are specialized configurations of the transceiver datapath, where the traffic stream is folded back to the source.

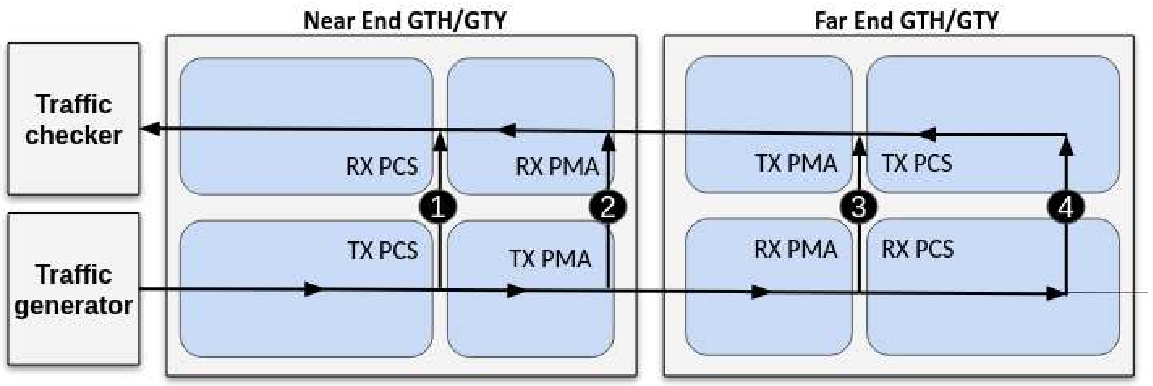


Figure 7.6: Loopback testing modes ([71]). There are two Near-end modes (1,2) and two Far-end modes (3,4).

Loopback test modes fall into two broad categories. The Near-end loopback modes transmit data back to the transceiver, closest to the traffic generator, either through PCS or from PMA. The

Far-end loopback modes loop received data back to the transceiver, at the far end of the link, either through PCS or from PMA.

Port	Description	Path
000	Normal Operation	-
001	Near-end PCS Loopback	1
010	Near-end PMA Loopback	2
011	Reserved	-
100	Far-end PMA Loopback	3
101	Reserved	-
110	Far-end PCS Loopback	4

Table 7.3: GTH/GTY Loopback selection table [71]

The loopback mode paths are illustrated in Figure 7.6, and the corresponding LOOPBACK_MODE_SEL values are listed in Table 7.3.

CLOCKS

Three clocks are required by the links in order to function properly; the **MGT's reference clock**, the **processing clock**, and a **stable clock** for the initialization of the transceivers.

The *MGT_REF_CLK* is the clock that drives the transceivers. This must be a low-jitter clock, and is used to generate and recover high-speed serial clocks in the GTH or GTY transceivers. The reference clock must be driven with high-quality clock sources, to decrease jitter and prevent bit errors. This is the main advantage of decoupling the processing clock from the MGT's reference clock.

The *TTC_CLK* is the processing clock that drives all the processing units. This clock is required due to the fact that the transceivers run asynchronously to the processing logic. It is used to clock in the data for transmission, and clock out the received data.

The *STABLE_CLK* is required, by the reset controller block, to reset the transceiver primitives, and must be present prior to the device configuration. It drives all the reset state machines and the initialization state machine. The frequency of this clock must not exceed the slowest frequency of the transceiver channel's user clock.

RX_DATA

RX_data is the output port of the data received and send to the processing units. It has the same array format as the TX_DATA port, and user can select the size of the array and the data bus width, by setting the value of the *N_CHANNEL* and *LWORD_WIDTH* respectively, when instantiating the top module.

INTTIALIZATION_DONE

The *INTTIALIZATION_DONE* signal indicates if the transmitter and the receiver of the MGT was initialized without problems. The INTTIALIZATION_DONE signal will de-assert in case of a hard link error, triggering an automated re-initialization procedure. In general, when the INTTIALIZATION_DONE indicator is high, the link is up and no hard errors are occurring. However, it does not guarantee an error-free link, since it is not affected by soft (isolated) errors. More information on assertion of the *INITIALIZATION_DONE* signal and the initialization state machine can be found in subsection 7.4.9.

LINK_STATUS and LINK_ERROR_LATCHED

The *LINK_STATUS* signal is asserted after a successful link initialization. It is an indicator of the links health, along with the error counter and the latched error signal. It will de-assert if the link error ratio is higher than 3×10^{-2}, indicating hard errors are occurring, but will re-assert if the link error ratio drops again. However, the bit error ratio acceptance of the link is set to 10^{-12}, which means that LINK_STATUS is an indicator of the link being up and running, but not necessarily error-free. A transition from “1” to “0” of this indicator will trigger a link re-initialization. More about the *LINK_STATUS* state machine in can be found in subsection 7.4.10.

The *LINK_ERROR_LATCHED* signal indicates that an error occurred during the transmission of the data. It needs to be reset after a successful link initialization. This indicator is useful to catch isolated sync header or code block errors, since it will de-assert if a single error occurs and will remain in that state until the user manually resets it.

CRC_ERROR_COUNTER

The *CRC_ERROR_COUNTER* is an 8 bit register that increments from zero to 255 when a CRC error occurs. The CRC checksum is sent at the end of a frame, or when the valid bit changes from “1” to “0”. A CRC error indicates that at least one error occurred in the previous frame. It is not affected by the sync header or the block codes, but triggers on errors in the data stream. More about the CRC checks can be found in subsection 7.4.6.

BUFFER_BYPASS_DONE

The GTH/GTY transmitter and receiver have two internal parallel clock domains; the PMA parallel clock domain and the PCS clock domain. To transmit data, all phase differences between the two domains must be resolved. For this reason, a phase adjust FIFO has been included in the GTH/GTY transmitter (TX elastic buffer) and receiver (RX elastic buffer) primitive. However, the buffers add latency to the system, and since low latency is critical for the CMS L1 trigger subsystems, the buffers were bypassed. In order to do that, Xilinx offers a helper block, called the phase alignment circuit, that adjusts the phase difference between the PMA parallel clock domain and the PCS domain. Upon successful completion of the buffer bypass and the phase alignment procedure, the BUFFER_BYPASS_DONE signal is set to “1”. If the buffer bypass fails, the transceivers need to be reset [71].

RESET_DONE

The GTH.GTY transmitter and receiver use a reset state machine, to control the reset process. Activating the TX and RX reset input triggers a full asynchronous reset of the transmitter and receiver blocks covering the entire PMA and PCS blocks. A state machine activates a series of resets and upon completion, RESET_DONE is changed from Low to High. For more details on the reset procedure, see Xilinx’s transceiver guide (*UltraScale Architecture GTH Transceivers* [71]).

7.4.5 Data Sources

There are three data sources to choose from, depending on the configuration of the transmitter (Figure 7.7). The first is a 32-bit PseudoRandom Binary Sequence (PRBS) generator, that is used for data integrity testing (i.e. BER tests). The second source is a simple counter that is used to test specific link functionality.

Finally, the third option is user data coming from the processing units. The multiplexer along with the user data generators are included in the *ultrascale_fifo_stimulus_64b66b.vhd* entity.

The PRBS-31 generator block

Transmitting PseudoRandom Binary Sequence (PRBS) patterns is a very common method to test the robustness of a link, in serial interconnect technologies. The International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Recommendation O.150 [79] defines the digital (pseudo-random) test sequences to be used for error performance measurements.

To test the link performance, the $2^{31} - 1$ pseudo-random test sequence (PRBS-31) was chosen.

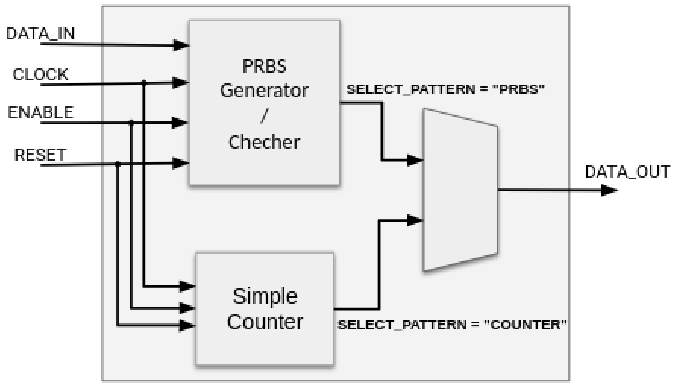


Figure 7.7: The transmitter stimulus module, implementing a PRBS test pattern generator, and a simple data source counter. The generic string "SELECT_PATTERN", is used to generate the desirable data source.

Attribute	**Description**	Value
CHK_MODE	False : Generator True : Checker	
INV_PATTERN	True : invert PRBS pattern	True
POLY_LENGHT	Length of the polynomial (number of shift register stages)	31
POLY_TAP	Intermediate stage that is XOR-ed with the last stage to generate to next PRBS bit	28
NBITS	Bus size of DATA_IN and DATA_OUT	64

Table 7.4: PRBS-31 Generator and Checker module parameters.

The 31^{st} degree polynomial selected as generator is the following :

$$g(x) = x^{31} + x^{28} + 1. \tag{7.2}$$

This trinomial[5] guarantees that the maximum period length of $2^{31} - 1$ is obtained, with the minimum number of feedback connections. All trinomials tested and recommended by the ITU-T are displayed in Table 7.5.

In practice, the PRBS bit-pattern is generated in a linear feed-back shift-register with a XOR-ed feedback of the output-values of specific flip-flops to the input of the first flip-flop. The sequence is generated in a twenty-nine-stage shift register, whose 28^{th} and 31^{st} stage outputs are added in a modulo-two addition stage, and the result is fed back to the input of the first stage. For the implementation of the PRBS generator, the parallel, programmable PRBS generator/checker circuit, designed by Xilinx [80], was used. In Table 7.4, there is a list of the

[5]A trinomial is a polynomial consisting of three terms or monomials.

Polynomial	POLY_LENGTH	POLY_TAP	INV_PATTERN
$x^7 + x^6 + 1$	7	6	TRUE
$x^9 + x^5 + 1$	9	5	FALSE
$x^{11} + x^9 + 1$	11	9	FALSE
$x^{15} + x^{14} + 1$	15	14	TRUE
$x^{17} + x^{14} + 1$	17	14	FALSE
$x^{20} + x^3 + 1$	20	3	FALSE
$x^{23} + x^{18} + 1$	23	18	TRUE
$x^{29} + x^{27} + 1$	29	27	TRUE
$x^{31} + x^{28} + 1$	31	28	TRUE

Table 7.5: Configuration for PRBS Polynomials Most Used to Test Serial Lines ([80])

module attributes, used to program the generator and the values chosen to test the links. If we want to select a different polynomial generator, Table 7.5 lists the recommended polynomials and the corresponding attribute values.

A simple counter

As an alternative data source, for testing purposes, a 16-bit counter was implemented. In order to form the 64-bit words, the 16-bit counter values and it's complement, was used twice. Using the complement ensures an equal number of ones and zeros in the word. Figure 7.8 shows the format of the 64-bit words when using the counter as source.

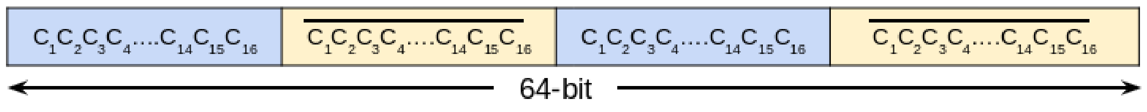

$C_1C_2C_3C_4....C_{14}C_{15}C_{16}$
$\overline{C_1C_2C_3C_4....C_{14}C_{15}C_{16}}$
$C_1C_2C_3C_4....C_{14}C_{15}C_{16}$
$\overline{C_1C_2C_3C_4....C_{14}C_{15}C_{16}}$
64-bit

Figure 7.8: A simple 16-bit counter can be selected as data source for testing purposes. The 64 bit data words are formed by using the 16-bit counter word and its complementary word twice.

User Data

In normal operation, the data source is the input from the processing units. The inputs consist of a 64-bit payload and the valid bit. As described in detail in subsection 7.4.6 and subsection 7.4.8, the valid bit defines the START and END of the frame, and triggers the insertion of various control signals and CRC checksum words.

7.4.6 Error Control - The Cyclic Redundancy Check generator and checker

In computer science and telecommunications, error detection and error control techniques are necessary for reliable delivery of digital data over unreliable communication channels. Most communication channels are subject to channel noise, and thus errors may be introduced during transmission, from a source to a receiver. These techniques allow the detection of such errors, while error correction enables reconstruction of the original data.

The Cyclic Redundancy Check (CRC), is a technique that allows the error detection, but not correction of such errors in digital data. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ensure, with a certain probability, whether or not an error occurred during transmission.

The CRC is based on polynomial arithmetics, and computes the remainder of dividing one polynomial in GF(2) (Galois field with two elements)[6] by another. Addition and subtraction are done modulo 2, that is, they are both the same as the XOR operator. It is like treating the message as a very large binary number, dividing it by a fairly large prime and computing the remainder.

For example, the message "11001001", where the order of transmission is from left to right (110...), is a representation of the polynomial $x^7 + x^6 + x^3 + 1$. The sender and receiver agree on a certain fixed polynomial, called the generator polynomial. To compute an r-bit CRC checksum, the generator polynomial must be of degree r.

The sender appends r zero-bits to the m-bit message, and divides the resulting polynomial of degree $r + m - 1$ by the generator polynomial. This produces a remainder polynomial of degree $r - 1$ (or less). The remainder polynomial has r coefficients, which are the checksum. The quotient polynomial is discarded. The data transmitted (the code vector) is the original m-bit message followed by the r-bit checksum. The receiver can divide all the received bits by the generator polynomial and check if the r-bit remainder is zero [81].

The selection of a "good" polynomial generator, is the most important part of implementing the CRC algorithm. The polynomial must be chosen to maximize the error-detecting capabilities. The most important attribute of the polynomial is its length, because of its direct influence on the length of the computed check value. Numerous varieties of cyclic redundancy checks have been incorporated into technical standards, with the most commonly used polynomial lengths being CRC-8, CRC-16, CRC-32 and CRC-64.

For the asynchronous links, the CRC-32 polynomial was chosen, as defined in IEEE 802.3 (ETHERNET) standard. This polynomial has excellent error detection performance [82].

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

[6]A polynomial in GF(2) is a polynomial in a single variable x whose coefficients are 0 or 1.

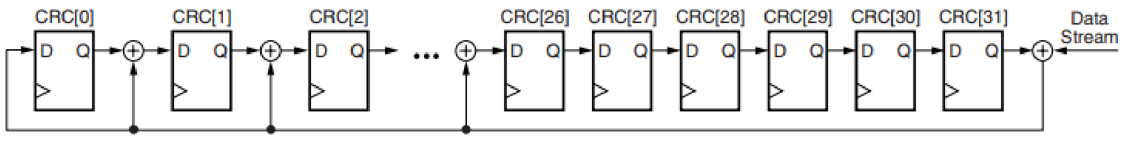


Figure 7.9: Serial Linear Feedback Shift Register Implementation of CRC-32 (register bits "3" through "25" are left out of the figure to simplify the drawing).

Name	Type	Range	Value	Description
POLYNOMIAL	vector	4-32 bits	0x04C11DB7	The CRC generator polynomial
INIT_VALUE	vector	4-32 bits	0xFFFFFFFF	Initialization value for the CRC register
DATA_WIDTH	integer	2-256 bits	64 bits	Width of the data input port
SYNC_RESET	integer	1	0b1	0 : Use asynchronous reset 1 : Use synchronous reset

Table 7.6: Generics defining the operation of the parallel CRC generator/checker module.

In hardware, a serial calculation of CRC-32 is implemented with a Linear Feedback Shift Register (LSFR). The CRC-32 LSFR is illustrated in Figure 7.9. The serial implementation, however, is not optimal for our 64-bit data-bus width, since it would require pipe-lining the data for 64 clock cycles, radically increasing the latency of the system. For this reason a parallel implementation was chosen, capable of calculating the CRC-32 checksum in 2 clock cycles, with a trade-off of increased resources used.

For the parallel implementation of the CRC generator and checker, an open source IP core was used, based on the proposed solution by E. Staninov [83]. The logic is generated during synthesis, from generics that specify the properties of the CRC (Table 7.6).

The module needs to be reset, before a CRC calculation initiates. After resetting, the clock enable signal must be asserted, before the first 64-bit data of the frame are pushed in. As long as the clock enable is high, a CRC checksum is calculated recursively. Clock enable must be de-asserted, at the last input data word of the frame, and after one clock cycle, the CRC checksum, corresponding to all frame bits, is pulled out from CRC_out port (see Figure 7.10). The CRC generation and checking are basically identical operations. If the CRC is clocked into the core after the data, the CRC register is set to zero, when no errors exist, and the crc_match_out is set to "High".

As can be seen in Figure 7.16, the generated CRC is inserted in the data stream with one clock delay, as required, after the last data word of the frame.

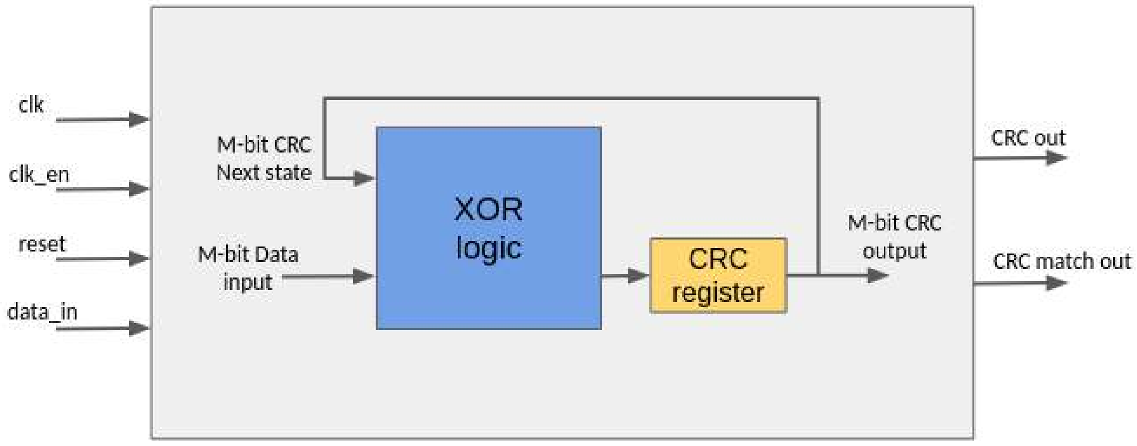


Figure 7.10: A simplified illustration of the CRC architecture.

7.4.7 Crossing from the Processing clock domain to the Link clock domain

As already mentioned, one of the prerequisites for a stable CMS trigger link is the use of a low jitter, stable, reference clock, to drive the the GTH/GTY transceivers. This implies that we need a local reference clock source, to drive the clock domain of the transceivers, while the processing clock domain can be driven by the LHC clock.

The interaction of the two asynchronous clock domains is called Clock Domain Crossing (CDC), and it is a quite common in multiple clock FPGA designs. The consequence of CDC is a metastability effect, which leads to either unpredictability of downstream data or data incoherency. These CDC issues can cause a significant number of failures in FPGA devices.

There are two basic methods for transferring data signals across clock domain boundaries. The first is based on enable-controlled data capture in the receiving domain. When the data are stable on the transmitting clock domain, the enable signal is asserted, informing the receiving clock domain that the data are ready to be captured. The stability of all data bits, during received data capture, guarantees the absence of the metastability effect. This method, however, has limited bandwidth. The second method is based on sequential writing and reading of data, using a dual-port FIFO, and can increase bandwidth across the interface, while maintaining reliable communication.

The dual-FIFO solution was chosen, because it provides a fast, reliable and high-bandwidth synchroniser. The synchronising circuit consists of a dual-FIFO IP core, the packet builder entity and the synchronous Gearbox sequence counter. The FIFO's data input is a 65-bit bus, consisting of the 64-bit data word and one data valid bit (MSB). The data are written in the FIFO at the processing clock frequency, after reset is de-asserted. The frequency of the processing clock must be lower than that of the link clock. When there are no more new data to be read in the FIFO, the empty flag is asserted, and no data are read for 1 link clock period. The packet builder state machine injects a padding word, upon detecting the empty flag signal assertion.

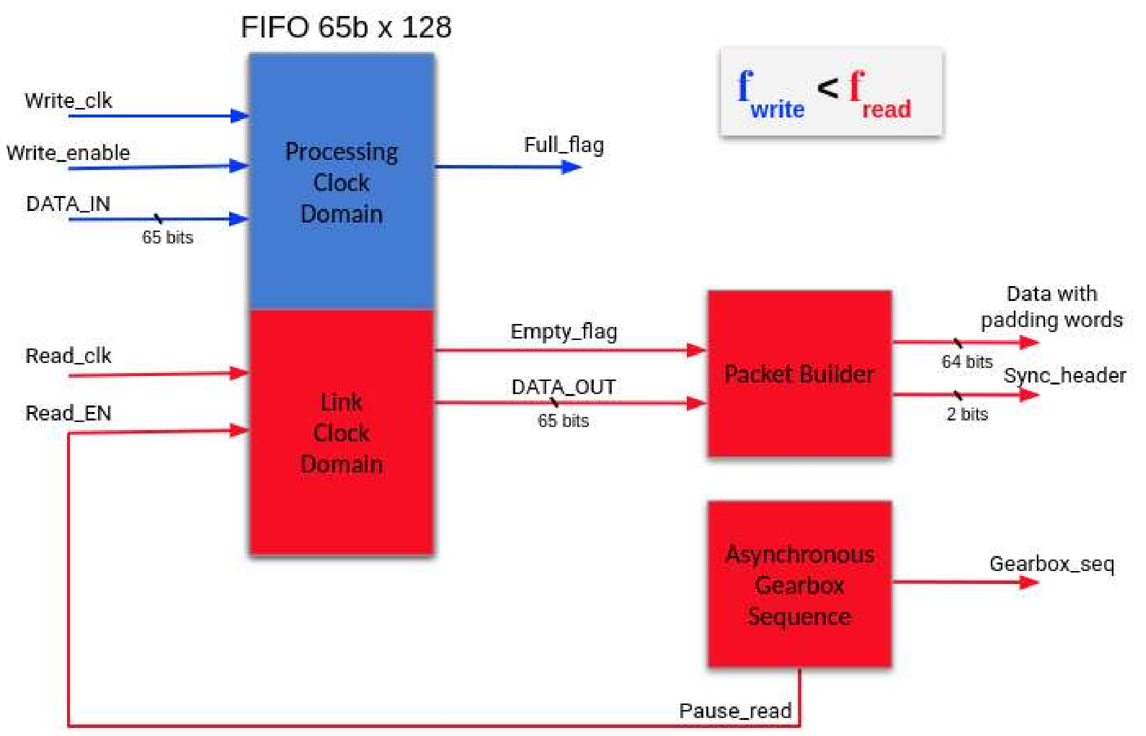


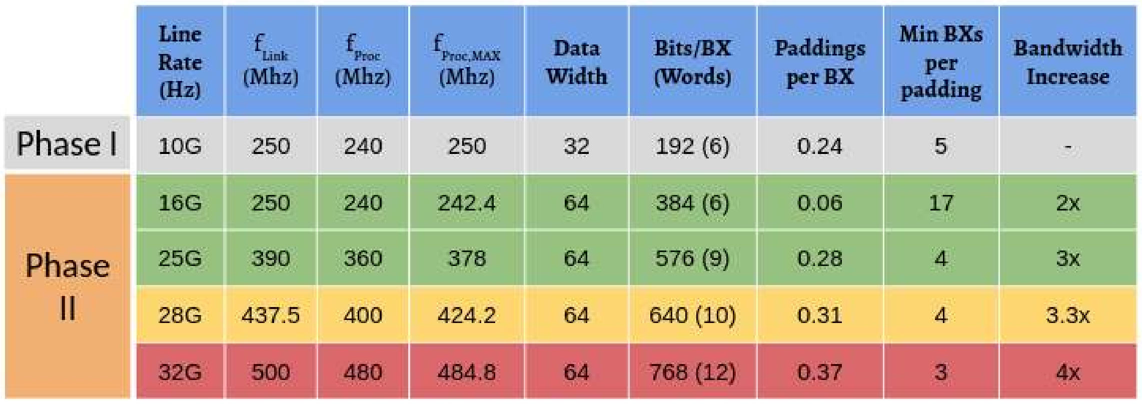
Figure 7.11: The TX synchronising circuit. It consists of a dual-FIFO IP core, the packet builder entity and the synchronous Gearbox sequence counter. The frequency of the processing clock is always lower than that of the link clock. When the empty flag of the FIFO is asserted, no data are read for one link clock period, and the packet builder state machine injects a padding word.

As mentioned, the processing clock is generated from the LHC clock, and its frequency will be a multiple of the LHC clock frequency, $f_{LHC} = 40.08MHz$.

This results in sending an integer number of parallel data words per bunch crossing. For example, the processing clock frequency for the 10 Gbps asynchronous links, used during Phase-1, was 240 MHz, allowing the transmission of six 32-bit words of data per bunch crossing. In Figure 7.12, the maximum processing clock frequencies are presented, for all different line rates supported by the optical transceiver modules for Phase-2.

In order to calculate the link clock frequency, when using the synchronous gearbox, we divide the line rate with the data bus width:

$$f_{Link} = \frac{Line\ Rate}{Data\ Bus\ Width} \tag{7.3}$$



	Line Rate (Hz)	f_{Link} (Mhz)	f_{Proc} (Mhz)	$f_{Proc,MAX}$ (Mhz)	Data Width	Bits/BX (Words)	Paddings per BX	Min BXs per padding	Bandwidth Increase
Phase I	10G	250	240	250	32	192 (6)	0.24	5	-
Phase II	16G	250	240	242.4	64	384 (6)	0.06	17	2x
	25G	390	360	378	64	576 (9)	0.28	4	3x
	28G	437.5	400	424.2	64	640 (10)	0.31	4	3.3x
	32G	500	480	484.8	64	768 (12)	0.37	3	4x

Figure 7.12: Possible line rates for Phase-2 and corresponding processing clock and link clock frequencies, along with the frequency of the padding word insertion.

for example for 16Gbps line rate and 64 bit data bus width:

$$f_{Link,16G} = \frac{16 \times 10^9 bits/sec}{64 bits} = 250MHz. \tag{7.4}$$

However, this calculation does not take into account the 2-bit header. The 2-bit header is combined with the 64-bit data in the transceiver core. For this reason, the Synchronous Gearbox requires the interruption of the reading sequence of the FIFO, once every 32 clock cycles. This has to be taken into account, to calculate the maximum frequency of the processing clock. Adding the synchronous gearbox interruption, we end up with:

$$f_{proc,max} = \frac{Line\ Rate}{66} \tag{7.5}$$

The average period of the padding words injection, expressed in frames, can be calculated using the frequency difference, between the link clock and the processing clock. For example, to estimate the average period of padding words, in the data stream, for the 16 Gbps line rate link:

$$f_{padding\ words} = \frac{f_{link} - f_{proc}}{f_{proc}} = \frac{242.42Mhz - 240Mhz}{242.42Mhz} = 0.0099pads/frame \tag{7.6}$$

$$T_{padding\ words} = \frac{1}{f_{padding\ words}} = 100.2. \tag{7.7}$$

Considering that a bunch crossing has a period of 25 ns, running the processing clock at 240 Mhz clock, allows us to send 6 frames per bunch crossing. We can calculate the average number of padding words per bunch crossing:

$$Average\ padding\ words\ per\ BX = 0.0099 \times \frac{240Mhz}{40Mhz} = 0.0594\ pad/BX \qquad (7.8)$$

or 1 padding word per 16.83 BXs.

A simulation waveform, of the interrupt flag, issued every 32 clocks by the synchronous gearbox, and the padding word flag, triggering the insertion of a padding word when the FIFO is empty, can be seen in Figure 7.13.

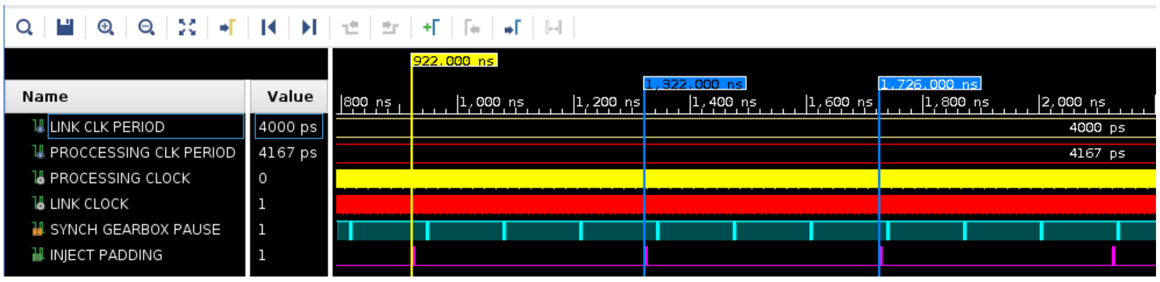


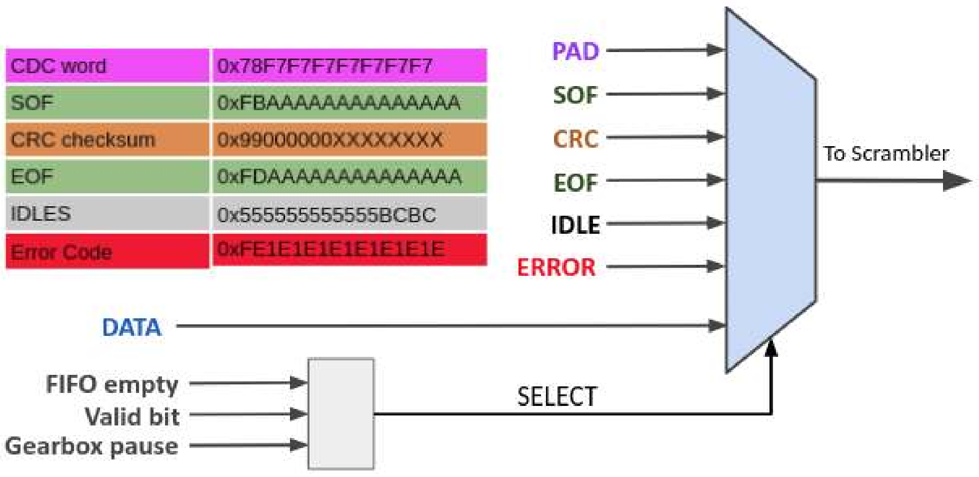
Figure 7.13: A simulation waveform depicting the behavior of the CDC circuit. The *"INJECT PADDING"* flag is indicating that the CDC FIFO is empty, and a filler (padding) word must be injected in the data stream. The period of the padding word insertion of $\sim$ 100 words (for 16 Gbps line rate) is in agreement with the value calculated in Equation 7.7. The *"SYNCH GEARBOX PAUSE"* flag indicates the interruption of data streaming, once every 32 link clock cycles, as required by the Synchronous Gearbox.

7.4.8 The Packet Builder

The packet builder module is responsible for adding the padding words, and the extra protocol coding word, to the data stream at the link clock domain. The selection of the coding word, to be injected, is triggered by the incoming data valid bit. The encoding words of the protocol, and a simplified schematic of the packet builder circuit, can be seen in Figure 7.14.

A transition of the data valid bit from "LOW" to "HIGH", marks the start of valid data stream. On the rising edge of the data valid bit, the data stream is delayed by one clock cycle, and the Start of Frame (SoF) coding word is injected. When the valid bit is high, the valid data stream is only interrupted, when the Synchronous Gearbox flag, or the TX FIFO's empty flag, is asserted. The SoF injection timing diagram can be seen in Figure 7.15.

When the valid bit is de-asserted, an IDLE coding word is injected, while waiting for the calculation of the CRC. The CRC coding word follows, and the End of Frame (EoF) coding word



CDC word	0x78F7F7F7F7F7F7F7
SOF	0xFBAAAAAAAAAAAAAA
CRC checksum	0x99000000XXXXXXXX
EOF	0xFDAAAAAAAAAAAAAA
IDLES	0x555555555555BCBC
Error Code	0xFE1E1E1E1E1E1E1E

Figure 7.14: The six coding words of the *Hermes* protocol: Padding (CDC), Start of Frame (SoF), CRC, End of Frame (EoF), Idle, Error, and a simplified schematic of the injection circuit.

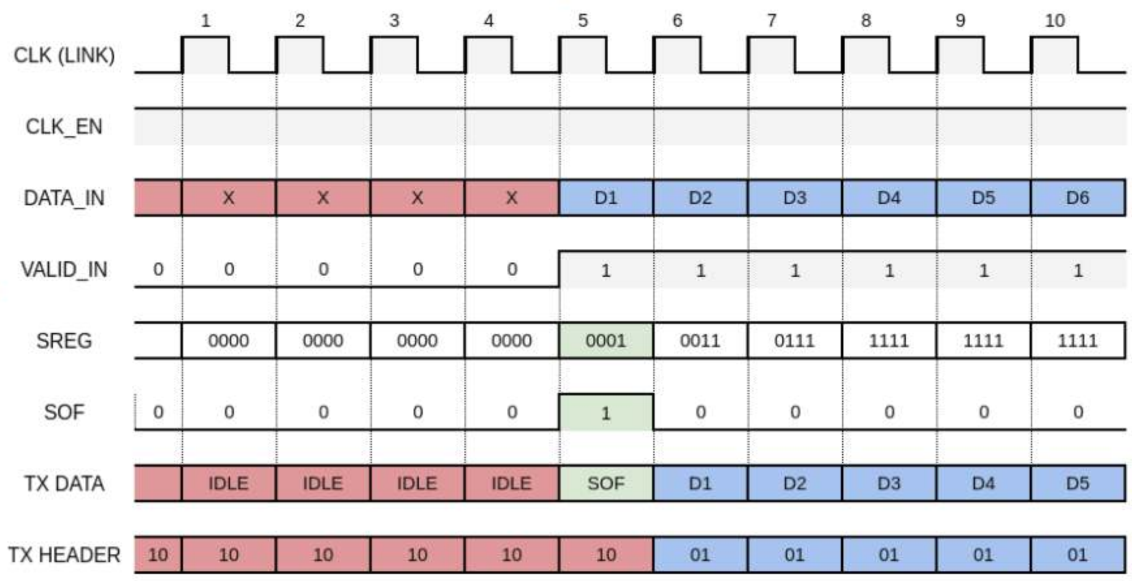


Figure 7.15: The SoF coding word marks the beginning of a valid data stream. The injection of the SoF is triggered by a rising edge of the data valid bit. The data are delayed by one clock cycle and the SoF word is injected. All invalid data between the EoF and the next SoF are replaced by IDLE words.

marks the end of a valid data stream. After the insertion of the EoF, all invalid data are replaced with IDLE coding words. A stream of invalid data is usually transmitted during the abort gap of the LHC. The CRC and EoF injection timing diagram can be seen in Figure 7.16.

The packet builder consists of a simple multiplexer, that adds only one clock of latency.

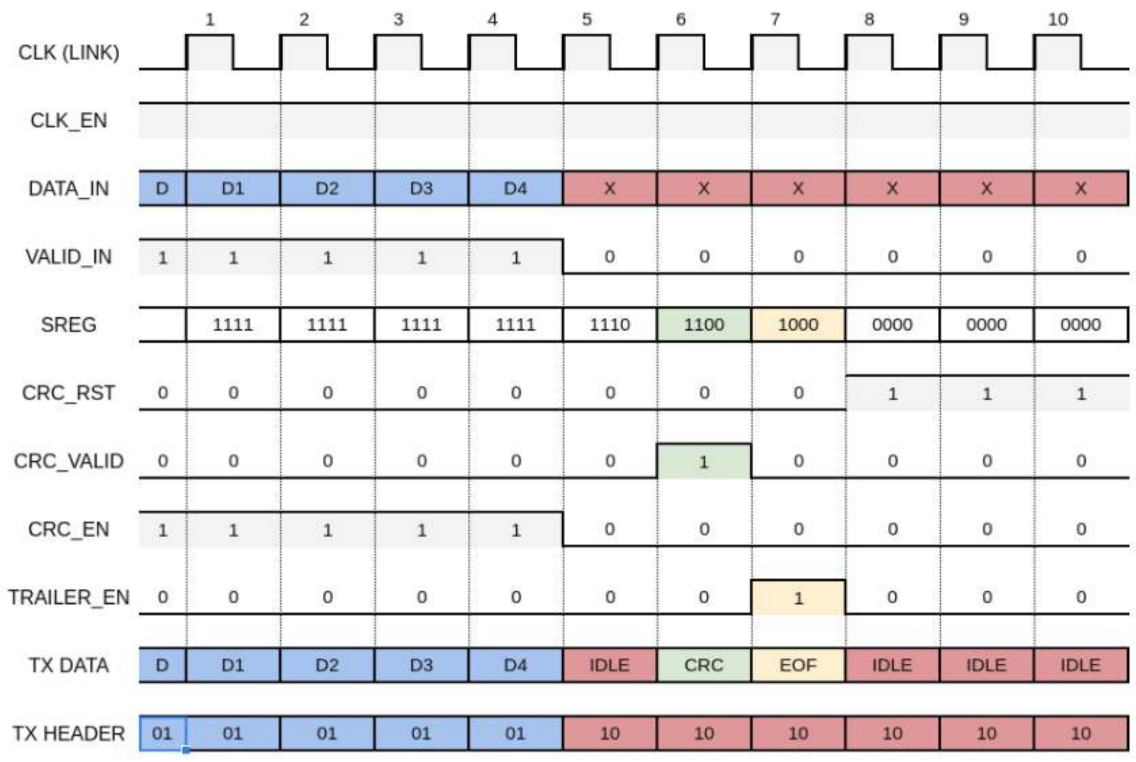


Figure 7.16: The CRC and EoF words are transmitted at the end of a valid data stream. A falling edge of the data valid bit, triggers the injection of the CRC word, followed by the EoF word. An additional IDLE character is transmitted, while the CRC is being calculated.

Padding words are injected when the FIFO is empty, with the highest priority. When the Sync Gearbox pause flag is set, nothing is inserted to the data stream, and the previous word is send again. The valid bit signal is the only control bit coming from the processing logic, and the one used to select between transmitting one of the four valid coding words (SoF, CRC, EoF or IDLE), or regular data. The selection is made by using a 4-bit shift register for the valid bit. In Table 7.7, the valid bit shift register values and the corresponding transmitted words are listed.

7.4.9 Initialization

The cores initialize automatically after power-up, upon reset, or when hard errors are detected (more on hard errors in subsection 7.4.10). During the initialization procedure, both the transmitter and the receiver of the transceivers are reset. Since the links run in simplex mode, there is no side-band connection, and a timer needs to be instantiated, to declare that the partner is out of initialization, and ready for data transfer. The total time for the transceiver to initialize and bypass the elastic buffers is 8.8 μs and 10.4 μs respectively, for the TX and RX. On the transmitting side, after resetting the transceivers, the transmitter waits for at least 20 μs, before any valid data can

SREG	CODE	Transmitted Word	Sync Header
0b0000	IDLE	0x555555555555bcbc	0b10
0b0001	SOF	0xfbaaaaaaaaaaaaaa	0b10
0b0011	DATA	-	0b01
0b0111	DATA	-	0b01
0b1111	DATA	-	0b01
0b1110	IDLE	0x555555555555bcbc	0b10
0b1100	CRC	0x99000000xxxxxx	0b10
0b1000	EOF	0xfdaaaaaaaaaaaaaa	0b10
OTHER	ERROR	0xfe1e1e1e1e1e1e1e	0b10

Table 7.7: The data valid bits are shifted in a 4-bit shift register. The value of the shift register triggers the transmission of the corresponding codding word and sync header.

be transmitted.

Upon completion of the reset procedure, data transmission is initiated. In order to align the incoming words, no special training patterns are required. Instead, the receiving initialization machine is checking the 2-bit header, and the padding words, for illegal values. If a stream of 16 consecutive errors is detected, the *gearbox_slip* bit signal is asserted, and all data are shifted by 1 bit. This is repeated, until no streaming errors appear, which means the received words are aligned.

The same initialization procedure is triggered during normal operation, if a stream of more than 16 consecutive errors are received. Since the padding words and the header are always transmitted, no change is required on the transmitting end. The receiver initialization logic will automatically re-align the links.

The initialization waveform of the links, with all the related signals, is depicted in Figure 7.17. After de-asserting the general reset, the transmitter reset procedure is completed, followed by the receiver reset procedure. The elastic buffers are bypassed, in the same order, and the transceiver alignment procedure starts. The *gearbox_slip* bit is asserted, when a stream of 16 consecutive errors are captured. The data are shifted, one bit at a time, until data alignment is achieved. The *link_up* indicator is asserted, after receiving 64 words without errors.

The error checking mechanism uses the special coding words of the protocol, and the 2-bit header, to check for errors in the channel. This makes the link bring-up and error checking mechanism independent of the actual data that are being streamed through the channel. Finally, if the *Error* coding word is received, an error must have occurred on the transmitting side, which is not an error in the actual communication.

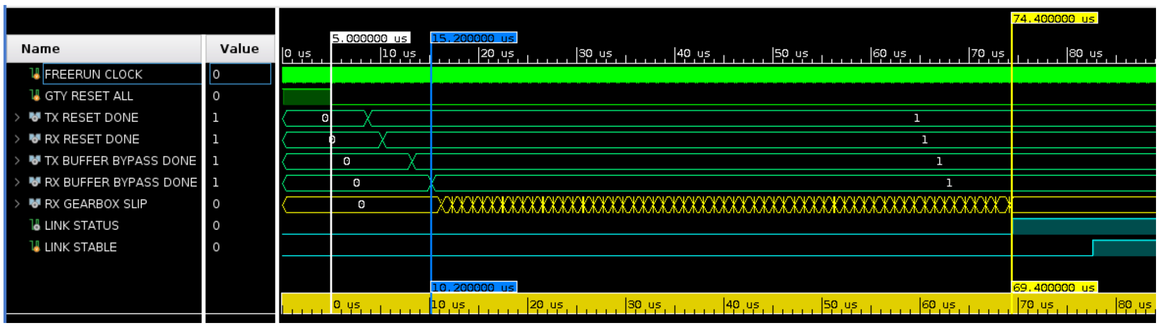


Figure 7.17: This simulation waveform shows the initialization and alignment mechanism of the *Hermes* protocol. First the TX and then the RX are reset, and their elastic buffers bypassed. The error checking mechanism shifts the data, one bit at a time, by setting the *gearbox_slip* bit, until there are no errors. After receiving 64 sequential error-free words, the *link_up* indicator is asserted.

7.4.10 Auto-link recovery

The auto link recovery mechanism is based on the same error checking circuit, used during initialization. After the successful initialization of the links, a counter is incremented by one, for every error-free data word received. When the counter reaches 64, the link is considered stable. However, upon reception of an error, this counter is decremented by 33. If we receive more than 2 errors for every 64 checks, the counter reaches zero and a reset and re-initialization procedure will start on the receiving side. The initialization and auto-align state machine diagrams are depicted in Figure 7.18.

7.4.11 Crossing from the Link clock domain to the Processing clock domain.

On the receiving end, in compliance with the transmitting side, the data must cross from the fast link clock domain to the slower processing clock domain. In order to achieve this, a similar CDC mechanism has been designed. This time, the only difference is the use of a Dual-BRAM instead of a FIFO. The main reason is that the Dual-BRAM allows control of the read/write pointers, that are required to align all channels (see Figure 7.19).

At the link clock domain, the incoming padding words are removed from the data stream. This is achieved by de-asserting the write enable of the Dual-BRAM, when a padding word is received. On the processing clock domain, the data, stripped from the extra padding words, are pulled out from the BRAM uninterrupted. During initialization, however, corrupt padding words can flood the BRAM. To avoid this, and optimize for latency, a reset o the BRAM read/write pointers is required, after the link is stable.

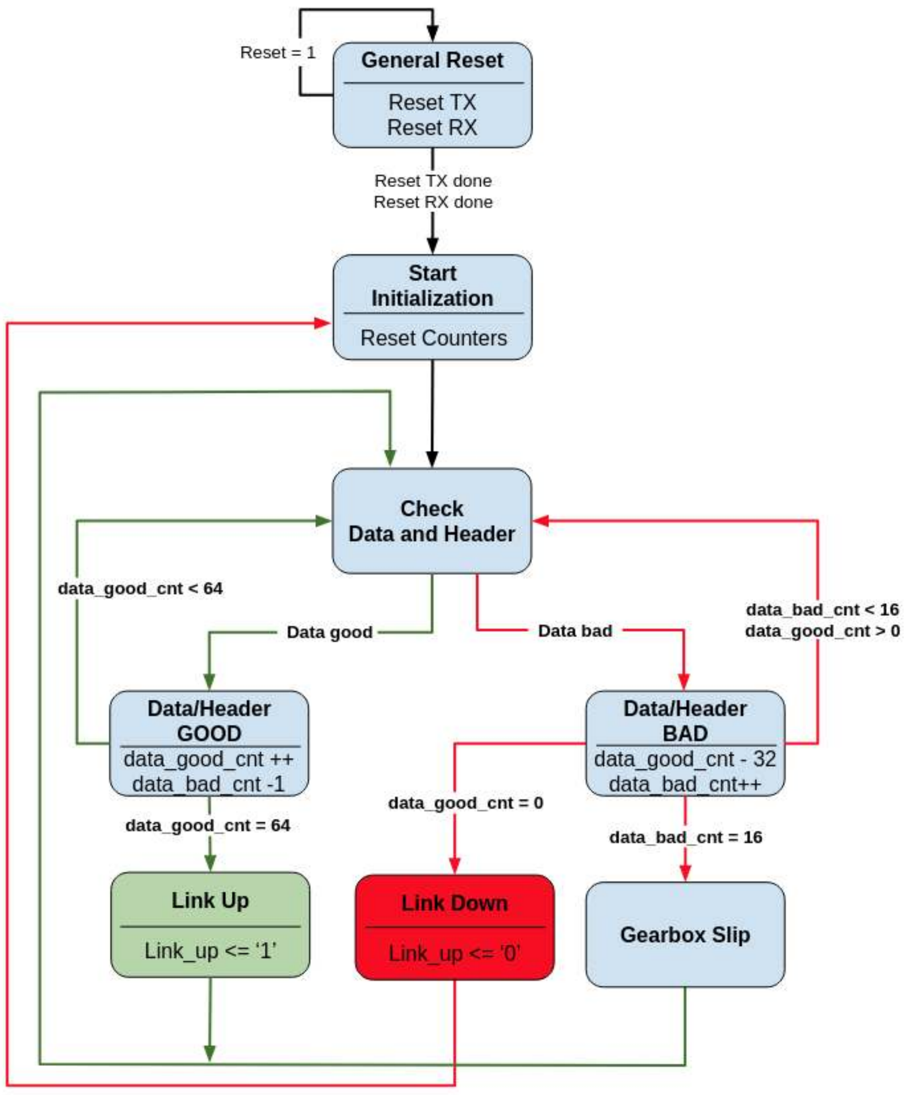


Figure 7.18: The initialization procedure starts after both transmitting and receiving reset sequences are completed successfully. The coding words and the sync header are checked for errors, and a 4-bit *data_bad* counter and a 6-bit *data_good* counter are increased or decreased respectively. If the *data_bad* counter reaches its maximum value, the *gearbox_slip* bit is set, and the data are shifted by one bit. The procedure repeats, until bit alignment is achieved. The *data_good* counter is increased by 1 for every data word received correctly. The link is considered stable, when the counter reaches its maximum value. While the link is in stable condition, the data good counter will decrease by 32, when an error is received. If the counter reaches 0, a new initialization procedure is triggered.

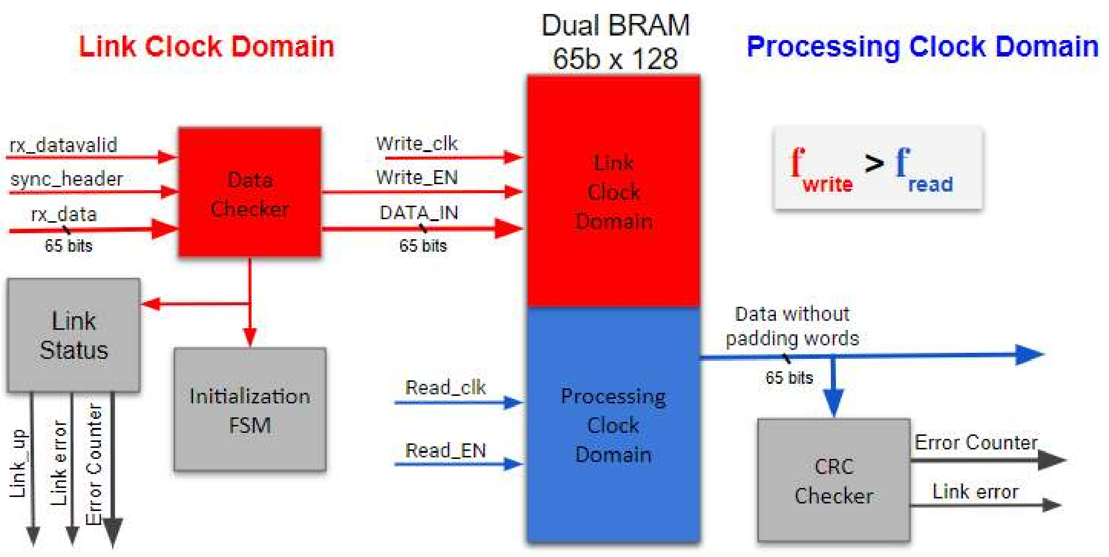


Figure 7.19: A simple illustration of the clock domain crossing circuit. An asynchronous dual port BRAM is used, with data being pushed in the RAM at link clock frequency, and pulled out at processing clock frequency. The padding words that were added, on the transmitting side, are not pushed to the BRAM, to compensate for the clock difference between the two domains.

7.4.12 CRC checksum error detection

On the receiver's processing clock domain, the data, stripped from the padding words, are forwarded to the processing units and the CRC checker module. The CRC checker module functionality was described in subsection 7.4.6. The CRC error detection circuit uses the valid bit, to detect the end of the frame. One clock cycle after the last data word of the frame, the local CRC checksum is pulled out from the CRC output port, and compared to the received CRC. If the two checksums do not match, the CRC error counter is incremented by one (see Figure 7.20). After the *link_stable* signal is asserted, the CRC counters must be reset, to remove any errors occurred during the link initialization.

7.4.13 The ONE_WORD simple link protocol

The *Hermes* Framing protocol is a general purpose protocol, offering a simple way to send data, grouped in packets (frames). However, one might want to stream data without the need of advanced framing features, like EoF or SoF. An alternative protocol was designed, to meet the needs of subsystems only streaming data, by adding just one extra coding word or SoF, containing the CRC code.

In addition, in order to avoid adding extra words as a payload, the periodical injection of the filler words, required to cross between processing and link clock domains, were used. The

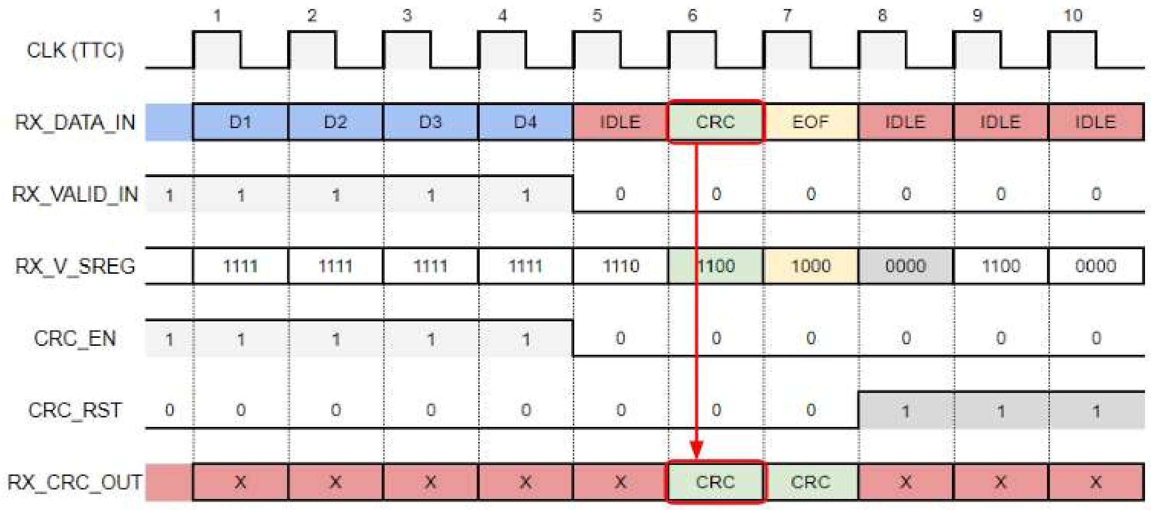


Figure 7.20: The CRC checksum is calculated at the end of a valid data stream. A falling edge of the data valid bit triggers the calculation of the CRC word. The locally calculated and the transmitted CRC checksums are compared; if no match is found, the CRC error counter is increased by 1.

CRC/SoF word were injected instead of the filler word. Special care must be taken, however, to inject the CRC/EoF word at a frequency shorter than the maximum filler rate. You can refer to Figure 7.12, column 4, for the minimum number of data words between fillers.

For this simple protocol, the CRC checksum must be generated and checked at the link clock domain, where the padding words are injected at the transmitter. At the receiving end, the CRC/EoF padding words are first checked for errors, and then stripped from the data stream, before crossing to the processing clock domain (see Figure 7.21).

7.5 Design Verification and Tests

The links performance was tested in different hardware configuration, at 16 Gbps and 25 Gbps. The development and initial testing of the links were realized using the KCU105 development board by Xilinx. Further testing was performed using the Ultrascale+ Xilinx development board, and the Barrel Muon Demonstrator board, that was designed by the Greek CMS group, to accommodate the efforts of Firmware and Hardware development for the Barrel Muon Trigger System upgrade for Phase-2 (see subsection 6.3.1). The results for the latency and the BER measurements are presented. All tests were repeated, for various different frame sizes, using simulation and hardware implementation with debugging logic cores and the IPBus software.

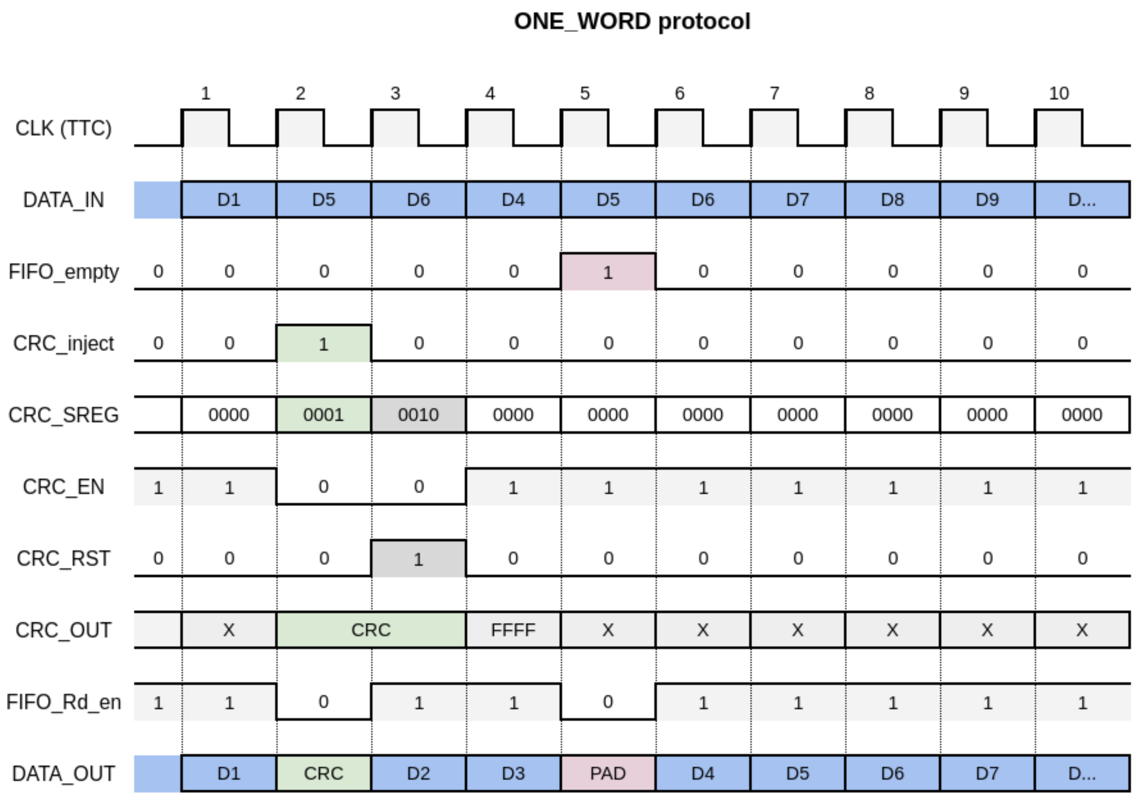


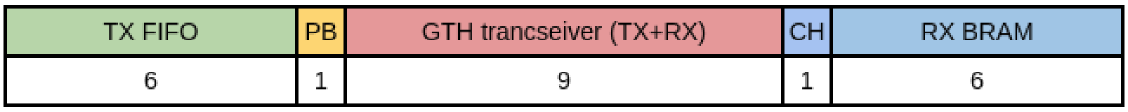
Figure 7.21: In the One Word protocol, only one extra coding word is added as a filler (TX FIFO read is paused), marking the start of a new frame and containing the CRC code.

7.5.1 Latency measurements

A breakdown of the expected latency of the links is given in Figure 7.22. The latency of the links was measured using the Xilinx ChipScope Analyzer. The transceivers were configured in "Loop-back mode", in order to measure the latency of the links, without the signal propagation time through the optical fiber (see ??). A snapshot of the waveform captured with the Chipsope software can be seen in Figure 7.23.

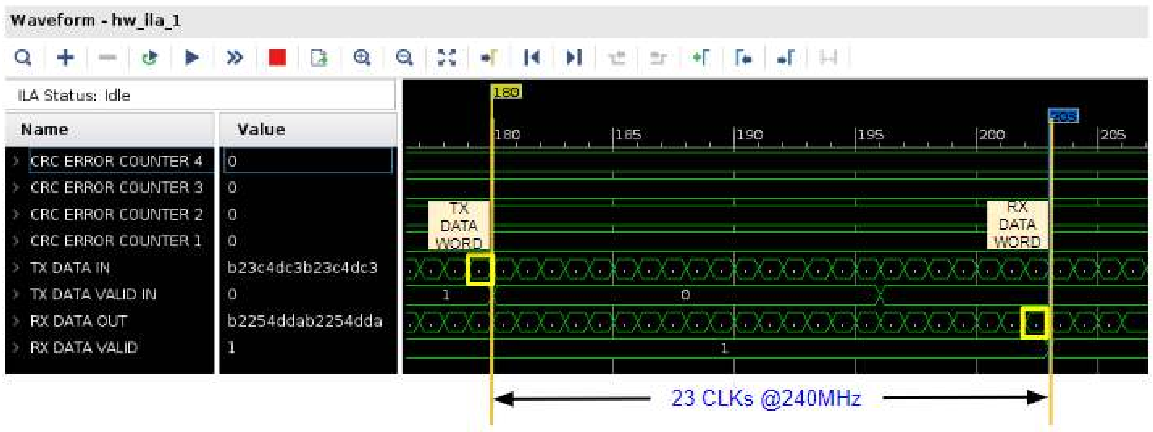
The measurement was verified, by an alternative measuring method, using the IPBus control software, to inject a pattern into the transmitter spy buffer, and detect them by capturing the receivers spy buffer. The offset between the TX and RX pattern is equal to the total link latency, measured in periods of the processing clock. The data path in the FPGA, along with a snapshot of the spy buffer content, as captured by the software, is shown in Figure 7.24. The offset was found to be 23 processing clocks. The processing clock frequency was 240 MHz, resulting in a period of 4.167 ns. The total link latency is:

$$Latency_{(16G@240MHz)} = 23clocks \times 4.167\ ns = 95\ ns\ or\ 3.8\ BXs.$$



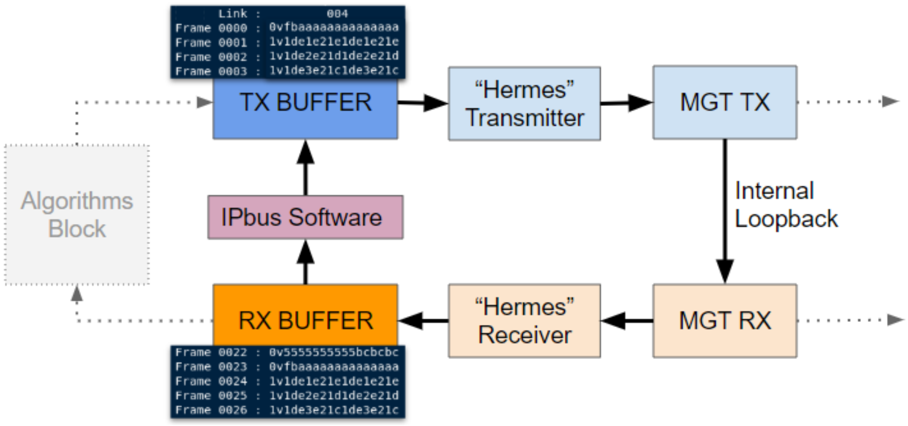
TX FIFO	PB	GTH trancseiver (TX+RX)	CH	RX BRAM
6	1	9	1	6

Figure 7.22: The latency of the links in periods of the link clock. The FIFO requires 6 clocks while the Packet Builder logic requires 1 clock to inject the SoF. The transceivers themselves (both transmitter and receiver) require 9 extra clocks in total. On the receiving end, the error checker and the padding word stripping require one more cycle and finally the RX BRAM add 6 more clocks. In summary, the total latency of the links is 23 clocks.



Name	Value
CRC ERROR COUNTER 4	0
CRC ERROR COUNTER 3	0
CRC ERROR COUNTER 2	0
CRC ERROR COUNTER 1	0
TX DATA IN	b23c4dc3b23c4dc3
TX DATA VALID IN	0
RX DATA OUT	b2254ddab2254dda
RX DATA VALID	1

Figure 7.23: In this example we captured the last data word of a frame (notice the valid bit de-assertion). The last word was captured at the RX FIFO output after 23 clocks.



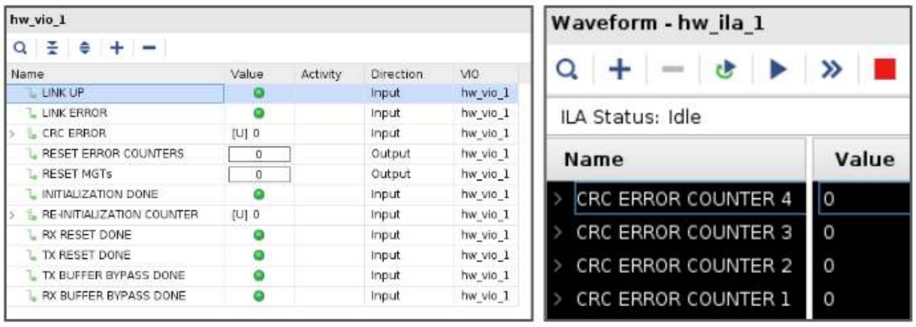
```
           Link :          004
Frame 0000 : 0vfbaaaaaaaaaaaaaa
Frame 0001 : 1v1de1e21e1de1e21e
Frame 0002 : 1v1de2e21d1de2e21d
Frame 0003 : 1v1de3e21c1de3e21c
```

```
Frame 0022 : 0v5555555555bcbcbc
Frame 0023 : 0vfbaaaaaaaaaaaaaa
Frame 0024 : 1v1de1e21e1de1e21e
Frame 0025 : 1v1de2e21d1de2e21d
Frame 0026 : 1v1de3e21c1de3e21c
```

Figure 7.24: A pattern was injected in the TX spy buffer and run through the internal loopback of the links and captured by the receiving spy buffer. Since the rx data are clocked in the spy buffer with the processing clock, by multiplying the 23 frames offset with the clock period we get a measurement of the total link latency.

7.5.2 Bit Error Ratio measurements

The links performance was tested mainly with the KCU105 board. Both link protocol designs were tested, for more than 72 hours, with all link error indicators and the CRC error counters monitored, using the Chipscope VIO cores and the IPBus software.


hw_vio_1

Name	Value	Activity	Direction	VIO
LINK UP	●		Input	hw_vio_1
LINK ERROR	●		Input	hw_vio_1
CRC ERROR	[U] 0		Input	hw_vio_1
RESET ERROR COUNTERS	0		Output	hw_vio_1
RESET MGTs	0		Output	hw_vio_1
INITIALIZATION DONE	●		Input	hw_vio_1
RE-INITIALIZATION COUNTER	[U] 0		Input	hw_vio_1
RX RESET DONE	●		Input	hw_vio_1
TX RESET DONE	●		Input	hw_vio_1
TX BUFFER BYPASS DONE	●		Input	hw_vio_1
RX BUFFER BYPASS DONE	●		Input	hw_vio_1

Waveform - hw_ila_1

ILA Status: Idle

Name	Value
CRC ERROR COUNTER 4	0
CRC ERROR COUNTER 3	0
CRC ERROR COUNTER 2	0
CRC ERROR COUNTER 1	0

Figure 7.25: Monitoring the links for errors with the Xilinx ChipScope Analyzer. All debugging internal signals for the links were probed and monitored for the duration of the BER tests.

The calculated maximum BER for 72 hours of running at 16 Gbps line rate is:

$$BER^{copper}_{max,16Gbps} = \frac{Number\ of\ errors}{Number\ of\ bits\ send} = \frac{1}{(72h \times 3600s/h) \times (16 \times 10^9 bits/s)} = 2.41 \times 10^{-16}$$

and for 25 Gbps line rate:

$$BER^{copper}_{max,25Gbps} = \frac{Number\ of\ errors}{Number\ of\ bits\ send} = \frac{1}{(72h \times 3600s/h) \times (25 \times 10^9 bits/s)} = 1.54 \times 10^{-16}$$

The first tests were in “Loopback” mode, to test the behaviour of the link logic. Then, a loopback copper test was performed, with no errors detected.

For the optical loopback tests, we used the L1-BMT demonstrator board, and initialized 3 quads. The transceivers were connected to the 12 optical firefly modules and, through a 10 m long OM3 optical cable, were looped back to the receivers. The configuration can be seen in Figure 7.26.

Running for 72 hours, in a more realistic configuration this time, produced no errors, resulting in a maximum BER of:

$$BER^{optical}_{max,16Gbps} = \frac{1}{(72h \times 3600s/h) \times (16 \times 10^9 bits/s)} = 2.41 \times 10^{-16}$$

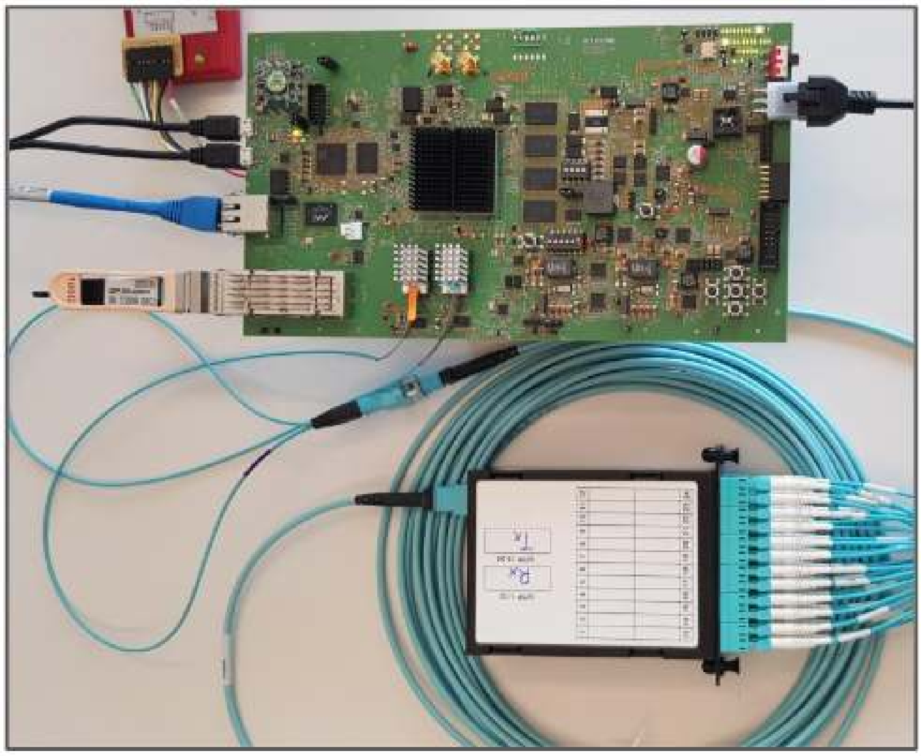


Figure 7.26: The optical loopback test configuration used to test the *Hermes* links. Three MGT quads (12 transceivers) are routed to the Firefly optical modules. The 12 optical transmitter fibers and 12 optical receiver fibers, coming from the Firefly optical engines, are merged in a 10 m long MTP cable with 24 fibers. An MTP-to-LC cassette is used, to split the fibers and forward the transmitters to the corresponding receivers, creating a 20 m optical loopback path.

Chapter 8

Conclusions

Following the major upgrade of the LHC during Long Shutdown 2 (2024 - 2027), the detector electronics and the trigger system of the CMS experiment will be replaced. In the present dissertation, the development of a new high speed protocol for the trigger system, the debugging and commissioning of a new muon tracking algorithm, and the design and testing of a demonstrator board for the barrel muon trigger upgrade were presented.

The CMS Trigger groups identified 16.000 Gbps and 25.78125 Gbps line rates with 64b66b encoding for wide use in Phase-2 trigger applications. The trigger protocol and the amount of physics payload carried over the links have important implications towards algorithm logic and algorithm developers. The link protocols running during Phase-1 will be replaced to accommodate the needs of the CMS Level 1 trigger. The *Hermes* protocol was designed and tailored to meet the specified requirements for Phase 2.

Hermes utilizes the latest state-of-the art GTH/GTY Xilinx transceivers and incorporates the 64b66b encoding. The benefit of using the 64b66b encoding scheme is the use of only 2 bits of overhead for every 64 bits of data, resulting in 96.875% efficiency – an improvement by $\sim$16%, compared to the 80% efficiency of the previously used 8b10b encoding. The use of a separate clock to drive the transceivers was verified during Phase 1 using the 10 Gbps asynchronous links. A similar method was used for the *Hermes* links, to provide a low-jitter local clock source for the transceivers and to ensure stability.

Furthermore, the firmware was designed to meet the specific needs of the CMS trigger, and only the essential components were included, making it a very lightweight design. Extra care was taken to reduce the latency by removing the GTH/GTYs' internal elastic buffer, on both the transmitter and receiver blocks. The latency of the links was measured to be 95 ns (or 3.8 BXs) for the 16 Gbps links, and 75 ns (or 3 BXs) for the 25 Gbps links. For comparison, the latency for the 10 Gbps asynchronous links was 85 ns (or 3.5 BXs).

In addition, three layers of error detection mechanisms were implemented. The first and second layers are used to check for illegal header values and illegal coding word values respectively, and are also used for word alignment. The third layer is based on the CRC checksum method, and it is used to verify the contents of the payload. A link auto-recovery mechanism was implemented to bring up the link after any interruption in the transmission. Finally, a light version of the links was developed for streaming applications that do not require framing of the data.

The protocol was integrated in various development boards (KCU105 and KCU116), and Phase-2 demonstrator boards (L1-BMT, SERENITY and OCEAN). The firmware was exhaustively tested with the results being excellent, transmitting data over a 20 m optical loopback configuration, running for several days without errors. The calculated bit error ratio is less than 10^{-15}. The latency was measured using two different methods, implementing Xilinx logic analyser cores and injecting/capturing data with spy buffers. The firmware design and the test results were presented to the CMS trigger groups, within collaboration meetings and workshops. A poster, titled "A BMT Layer-1 technology demonstrator card - Hardware and Firmware", was presented at the 2018 Topical Workshop on Electronics for Particle Physics (TWEPP) [84]. CMS trigger groups have shown interest in testing the protocol. Two of them, the Imperial College group and the UCLA group, have already adopted and integrated the firmware in their core frameworks.

The Barrel Muon Track Finding algorithm, commissioned during the Year-End Technical Stop of 2015 displayed an excellent performance, with muon-detecting efficiency of more than 97%. However, its approach of using Look-up tables implemented with B RAMs, provides only a limited amount of address space. This resulted in the use of only two DT stations for the assignment of the transverse momentum. The use of an extra constraint, that assumes that the muon originated from the center of the detector, greatly improves the momentum resolution, but diminishes the efficiency of the algorithm in detecting displaced muons.

The new algorithm, presented in this dissertation, uses an approximate Kalman filter and exploits the measurements in all four muon stations. This allows the assignment of transverse momentum with good resolution, without the use of a vertex constraint. The Kalman filter algorithm was commissioned during the 2018 run, with cosmics, proton-proton and heavy ion collisions. The efficiency of detecting displaced muons was greatly improved, especially at distances of 40 to 100 cm from the primary vertex, where the efficiency doubled or tripled.

The commissioned firmware implements both the legacy and the Kalman Filter algorithms in the same FPGA, occupying $\sim$70% of the total resources. The final version of the firmware was in 99% agreement with the emulated one and is going to be the standalone BMTF algorithm during the data taking period of Run 3 (2021-2023).

HLS studies have shown that the Kalman Filter firmware performs best, in terms of latency and resource utilization, at 200 MHz. The legacy firmware could only run at frequencies up to 160 MHz, which made the use of a higher frequency clock impossible. However, the future removal of the legacy firmware will allow the clock frequency to go up to 200 MHz, and further reduce the total latency.

During the Phase-2 upgrade, all CMS detector and trigger electronics will be replaced, to exploit new available technologies. The Layer-1 Barrel Muon Demonstrator board was designed to provide the hardware environment, for developing and evaluating new Level-1 Trigger muon designs and technologies. It also provides an excellent way to test and extend our capabilities on FPGA board designing. For the PCB design, sophisticated techniques were adopted (i.e. serpentine routing, back-drilling), to achieve best performance and eradicate noise and jitter. The board includes a modern 20-nm technology FPGA, a Zynq SoC controller and twelve state-of-the-art optical transceivers.

All different components of the board were tested extensively, with emphasis on the high speed optical links. The FPGA was powered on and programmed through JTAG, and all 12 optical links were tested in different copper and optical loopback configurations, for different line rates. The links performance was excellent and the bit error ratio was better than the 10^{-15} limit.

The successful design and implementation of the trigger demonstration board, together with the valuable experience gained, are highly beneficial for the future design of a more elaborate ATCA compatible board, that will be used for the Layer-1 Barrel Muon Trigger during Phase-2. Furthermore, the current board could be used to implement and test new algorithms and new link protocols, like *Hermes*. Finally, the board will be used in future slice tests, for system performance measurements.

Bibliography

[1] Andrew Rose. *Serenity - An ATCA prototyping platform for CMS Phase-2*. Tech. rep. CMS-CR-2018-327. Geneva: CERN, Oct. 2018. URL: https://cds.cern.ch/record/2646388.

[2] Michail Bachtis. *Upgrade of the CMS Barrel Muon Track Finder for HL-LHC featuring a Kalman Filter algorithm and an ATCA Host Processor with Ultrascale+ FPGAs*. Tech. rep. 2018.

[3] *Vivado HLS Optimization Methodology Guide*. UG1270. v2018.1. Xilinx Incorporated. Apr. 2018. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug1270-vivado-hls-opt-methodology-guide.pdf.

[4] N A Tahir et al. "The CERN Super Proton Synchrotron as a tool to study high energy density physics". In: *New Journal of Physics* 10.7 (2008), p. 073028. URL: http://stacks.iop.org/1367-2630/10/i=7/a=073028.

[5] Karlheinz Schindl. *The injector chain for the LHC*. Tech. rep. CERN, 1999.

[6] Oliver Brüning et al. "LHC design report(Volume I, The@ LHC main ring)". In: *Reports-CERN* (2004).

[7] Kenneth Aamodt et al. "The ALICE experiment at the CERN LHC". In: *Journal of Instrumentation* 3.08 (2008), S08002.

[8] LHCb Collaboration. "The LHCb Detector at the LHC". In: *JINST* 3.LHCb-DP-2008-001. CERN-LHCb-DP-2008-001 (2008). Also published by CERN Geneva in 2010, S08005. URL: https://cds.cern.ch/record/1129809.

[9] Giovanni Anelli et al. "The totem experiment at the cern large hadron collider". In: *Journal of Instrumentation* 3.08 (2008), S08007.

[10] Alessia Tricomi et al. "The LHCf experiment: modelling cosmic rays at LHC". In: *Journal of Physics: Conference Series*. Vol. 110. IOP Publishing. 2008, p. 062026.

[11] MoEDAL Collaboration et al. "Technical design report of the MoEDAL experiment". In: *CERN Preprint, CERN-LHC-2009-006, MoEDAL-TDR-1.1* (2009).

[12] G Apollinari et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report*. CERN, 2017.

[13] *The High-Luminosity LHC Project. 298th Meeting of Scientific Policy Committee*. Tech. rep. CERN/SPC/1068. CERN/FC/6014. CERN/3255. June 2016. URL: https://cds.cern.ch/record/2199189.

[14] Tai Sakuma and Thomas McCauley. “Detector and event visualization with sketchup at the cms experiment”. In: *Journal of Physics: Conference Series*. Vol. 513. 2. IOP Publishing. 2014, p. 022032.

[15] CMS collaboration et al. “Precise mapping of the magnetic field in the CMS barrel yoke using cosmic rays”. In: *Journal of Instrumentation* 5.03 (2010), T03021.

[16] A Hervé et al. “Status of the construction of the CMS magnet”. In: *IEEE Trans. Appl. Supercond.* 14 (2004), pp. 542–547.

[17] LL Jones et al. “The APV25 deep submicron readont chip for CMS detectors”. In: (1999).

[18] Frank Hartmann. *Evolution of silicon sensor technology in particle physics.* Vol. 275. Springer, 2017.

[19] CMS Collaboration et al. “The CMS experiment at the CERN LHC”. In: *Jinst* 3 (2008), S08004.

[20] CMS collaboration et al. “Commissioning and performance of the CMS pixel tracker with cosmic ray muons”. In: *Journal of Instrumentation* 5.03 (2010), T03007.

[21] H Chr Kästli et al. “CMS barrel pixel detector overview”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 582.3 (2007), pp. 724–727.

[22] Sudhir Malik, CMS Forward Pixel Collaboration, et al. “The CMS forward pixel detector”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 572.1 (2007), pp. 87–89.

[23] Viktor Veszpremi. “Performance verification of the CMS Phase-1 Upgrade Pixel detector”. In: *Journal of Instrumentation* 12.12 (2017), p. C12010.

[24] A Dominguez et al. *CMS technical design report for the pixel detector upgrade.* Tech. rep. Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), 2012.

[25] Serguei Chatrchyan et al. “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 30–61.

[26] Andrea Benaglia. “The CMS ECAL performance with examples”. In: *Journal of Instrumentation* 9.02 (2014), p. C02008.

[27] B Borgia et al. “The Electromagnetic Calorimeter Technical Design Report”. In: *CERN/LHCC CMS TDR* 4 (2006).

[28] Edwige Tournefier. “The preshower detector of CMS at LHC”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 461.1-3 (2001), pp. 355–360.

[29] CMS collaboration et al. *Properties of the Higgs-like boson in the decay H⟶ ZZ⟶ 4l in pp collisions at√ s= 7 and 8 TeV.* Tech. rep. CMS-PAS-HIG-13-002, 2013.

[30] Giovanni Abbiendi et al. “The CMS muon system in Run2: preparation, status and first results”. In: *arXiv preprint arXiv:1510.05424* (2015).

[31] CMS collaboration, CMS Collaboration, et al. “The muon project technical design report”. In: *CERN/LHCC* 32.404 (1997), p. 1997.

[32] Serguei Chatrchyan et al. “Performance of the CMS drift tube chambers with cosmic rays”. In: *Journal of Instrumentation* 5 (2010).

[33] Francesca Romana Cavallo et al. *Test of CMS Muon Barrel Drift Chambers with Cosmic Rays.* Tech. rep. CERN-CMS-NOTE-2003-017, 2003.

[34] CMS collaboration et al. “The performance of the CMS muon detector in proton-proton collisions at√ s= 7 TeV at the LHC”. In: *Journal of Instrumentation* 8.11 (2013), P11002.

[35] Pierluigi Paolucci. “The CMS Muon System”. In: *Astroparticle, Particle And Space Physics, Detectors And Medical Physics Applications.* World Scientific, 2006, pp. 605–615.

[36] Luigi Guiducci. *Upgrades of the CMS muon system in preparation of HL-LHC.* Tech. rep. 2018.

[37] CMS Collaboration et al. *Technical proposal for the phase-II upgrade of the Compact Muon Solenoid, CMS Technical proposal CERNLHCC-2015-010.* Tech. rep. CMS-TDR-15-02, CERN, 2015. https://cds. cern. ch/record/2020886, 2015.

[38] Luciano Musa. “FPGAS in high energy physics experiments at CERN”. In: *2008 International Conference on Field Programmable Logic and Applications.* IEEE. 2008, pp. 2–2.

[39] Clive Maxfield. *The Design Warrior’s Guide to FPGAs: Devices, Tools and Flows.* 1st. Newton, MA, USA: Newnes, 2004. ISBN: 9780750676045.

[40] I.A. Grout. “Digital Systems Design with FPGAs and CPLDs”. In: *Digital Systems Design with FPGAs and CPLDs* (Jan. 2008), p. 724. DOI: 10.1016/B978-0-7506-8397-5.X0001-3.

[41] Doulos Ltd. *VHDL Golden reference guide.* 4th. Doulos Ltd, 2015. ISBN: 978-1-910062-05-0.

[42] Peter Ashenden. *The Designer’s Guide to VHDL.* 3rd. Morgan Kaufmann Publishers, 2008. ISBN: 978-81-312-1855-6.

[43] Donald G Bailey. “The advantages and limitations of high level synthesis for FPGA based image processing”. In: *Proceedings of the 9th International Conference on Distributed Smart Cameras.* ACM. 2015, pp. 134–139.

[44] *Intel HLS Compiler:Best Practices Guide.* UG-20107. 19.1. Intel Incorporated. Apr. 2019. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/hls/ug-hls-best-practices.pdf.

[45] Emmanuel Casseau et al. “C-based rapid prototyping for digital signal processing”. In: *2005 13th European Signal Processing Conference.* IEEE. 2005, pp. 1–4.

[46] CMS collaboration et al. “CMS TriDAS project: technical design report, volume 1: The trigger systems”. In: (2000).

[47] A Rose. *The Level-1 Trigger of the CMS experiment at the LHC and the Super-LHC.* Tech. rep. 2009.

[48] CMS collaboration et al. “CMS technical design report for the level-1 trigger upgrade”. In: *CMS Technical Design Report CERN-LHCC-2013-011. CMS-TDR-12* (2013).

[49] *7 Series FPGAs overview. Advance product specification.* DS180. Rev. 2.6. Xilinx Incorporated. Feb. 2018. URL: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

[50] A Svetek et al. “The Calorimeter Trigger Processor Card: the next generation of high speed algorithmic data processing at CMS”. In: *Journal of Instrumentation* 11.02 (2016), p. C02011.

[51] K Compton et al. “The MP7 and CTP-6: multi-hundred Gbps processing boards for calorimeter trigger upgrades at CMS”. In: *Journal of Instrumentation* 7.12 (2012), p. C12024.

[52] A. Madorsky. "Electronics for CMS Endcap Muon Level-1 Trigger System Phase-1 and HL LHC upgrades". In: *Journal of Instrumentation* 12.07 (July 2017), pp. C07010–C07010. DOI: 10.1088/1748-0221/12/07/c07010. URL: https://doi.org/10.1088%2F1748-0221%2F12%2F07%2Fc07010.

[53] E Hazen et al. "The AMC13XG: a new generation clock/timing/DAQ module for CMS MicroTCA". In: *Journal of Instrumentation* 8.12 (2013), p. C12036.

[54] Andrea Triossi et al. "A New Data Concentrator for the CMS Muon Barrel Track Finder-Phase I Upgrade". In: *Technology and Instrumentation in Particle Physics 2014*. Vol. 213. SISSA Medialab. 2015, p. 412.

[55] Chun-Jie Wang et al. "Design of a high throughput electronics module for high energy physics experiments". In: *Chinese Physics C* 40.6 (2016), p. 066102.

[56] Geoffrey Hall et al. "A time-multiplexed track-trigger architecture for CMS". In: *Journal of Instrumentation* 9.10 (2014), p. C10034.

[57] M Baber et al. "Development and testing of an upgrade to the CMS level-1 calorimeter trigger". In: *Journal of Instrumentation* 9.01 (2014), p. C01006.

[58] G Iles, A Rose, et al. *A time-multiplexed calorimeter trigger for CMS with addendum*. Tech. rep. CMS-IN-2011-008, 2011.

[59] Alexandre Zabi et al. "The CMS Level-1 Calorimeter Trigger for the LHC Run II". In: *Journal of Instrumentation* 12.01 (2017), p. C01065.

[60] Andrea Triossi et al. "The CMS barrel muon trigger upgrade". In: *Journal of Instrumentation* 12.01 (2017), p. C01095.

[61] Ioanna Papavergou. "The CMS Level-1 muon triggers for the LHC Run II". In: *PoS* LHCP2018 (2018), p. 070. DOI: 10.22323/1.321.0070.

[62] WM Zabolotny and A Byszuk. "Algorithm and implementation of muon trigger and data transmission system for barrel-endcap overlap region of the CMS detector". In: *Journal of Instrumentation* 11.03 (2016), p. C03004.

[63] Dinyar Rabady. "Upgrade of the Global Muon Trigger for the Compact Muon Solenoid experiment at CERN". PhD thesis. Vienna U.

[64] Johannes Wittmann et al. "The upgrade of the CMS Global Trigger". In: *Journal of Instrumentation* 11.02 (2016), p. C02029.

[65] Andrea Perrotta. "Performance of the CMS high level trigger". In: *Journal of Physics: Conference Series*. Vol. 664. 8. IOP Publishing. 2015, p. 082044.

[66] CMS Collaboration. *The Phase-2 Upgrade of the CMS Level-1 Trigger*. Tech. rep. CMS-TDR-19-002. In preparation. CERN.

[67] Greg Welch and Gary Bishop. "An Introduction to the Kalman Filter". In: (2006).

[68] Rudolf Frühwirth. "Application of Kalman filtering to track and vertex fitting". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 262.2-3 (1987), pp. 444–450.

[69] *Vivado Suite: Design Analysis and Closure Techniques*. UG906. 2018.2. Xilinx Incorporated. June 2018. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug906-vivado-design-analysis.pdf.

[70] Nick Mehta. “Xilinx ultrascale architecture for high-performance, smarter systems”. In: *Xilinx White Paper WP434* (2013).

[71] *UltraScale Architecture GTH Transceivers*. UG576. Rev. 1.5. Xilinx Incorporated. July 2017. URL: http://xxcp05.apwindows.net/support/documentation/user_guides/ug576-ultrascale-gth-transceivers.pdf.

[72] Wikipedia contributors. *Hermes*. [Online; accessed 30 October 2019]. 2004. URL: https://en.wikipedia.org/wiki/Hermes.

[73] A.X. Widmer. “8b/10b encoding and decoding for high speed applications”. US Patent 6,977,599. Dec. 2005.

[74] High-Energy Physics (HEP) group, Imperial College, London and Iceberg Technology, Cornwall. *User guide for the MP7 Master Processor*. http://www.hep.ph.ic.ac.uk/mp7/documents/UserGuideForTheMP7.latest.pdf. 2014.

[75] American National Standard for Information Technology. “FIBRE CHANNEL Physical Interface-5”. In: (Sept. 2010), p. 116.

[76] Richard Dugan Rick Walker. “64b66b low-overhead coding proposal for serial links”. In: (2000), p. 17.

[77] “IEEE Standard for Physical Coding Sublayer (PCS) for 64B/66B, type 10GBASE-R”. In: *IEEE Std. Std 802.3 Section 4* (2005), pp. 389–424.

[78] *Aurora 64B/66B Protocol Specification*. SP011. Rev. 1.3. Xilinx Incorporated. Oct. 2014. URL: https://www.xilinx.com/support/documentation/ip_documentation/aurora_64b66b_protocol_spec_sp011.pdf.

[79] ITU-T. *GENERAL REQUIREMENTS FOR INSTRUMENTATION FOR PERFORMANCE MEASUREMENTS ON DIGITAL TRANSMISSION EQUIPMENT*. Tech. rep. INTERNATIONAL TELECOMMUNICATION UNION, 1996. URL: https://www.itu.int/rec/T-REC-O.150-199605-I/en.

[80] *An Attribute-Programmable PRBS Generator and Checker*. XAPP884. Rev. 1.0. Xilinx Incorporated. Jan. 2011. URL: https://www.xilinx.com/support/documentation/application_notes/xapp884_PRBS_GeneratorChecker.pdf.

[81] Henry S Warren. *Hacker’s delight; 2nd ed.* Upper Saddle River NJ: Addison-Wesley, 2013. Chap. 14. URL: https://cds.cern.ch/record/1538686.

[82] Philip Koopman and Tridib Chakravarty. “Cyclic redundancy code (CRC) polynomial selection for embedded networks”. In: *International Conference on Dependable Systems and Networks, 2004*. IEEE. 2004, pp. 145–154.

[83] Evgeni Stavinov. “A practical parallel CRC generation method”. In: *Circuit Cellar-The Magazine For Computer Applications* 31.234 (2010), p. 38. URL: http://outputlogic.com/my-stuff/circuit-cellar-january-2010-crc.pdf.

[84] Stavros Mallios. "A BMT Layer-1 technology demonstrator card - Hardware and Firmware". In: 2018. URL: https://indico.cern.ch/event/697988/contributions/3056078/attachments/1725944/2788040/Twepp_2018_poster_smallios.pdf. Poster presented at TWEPP workshop, Antwerp, Belgium.

[85] C Ghabrous Larrea et al. "IPbus: a flexible Ethernet-based control system for xTCA hardware". In: *Journal of Instrumentation* 10.02 (2015), p. C02019.

List of Figures

Appendix A

Appendix A: HDL code

A.1 *Hermes* protocol frame generator code

The Frame Generator Module is the “heart” of the *Hermes* protocol. It is responsible for the insertion of all the coding words in the data stream. The VHDL code that follows describes both *Framing* and *Single Word* versions of the protocol.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
library work;
use work.ucrc_pkg.all;

entity add_padding_and_prbs_data is
  generic(  WORD_WIDTH : natural := 64;
            PROTOCOL_SEL : string := "ONE_WORD"   -- "FRAMING" or "ONE_WORD"
         );
 port(
    clk                    : in  std_logic; -- link clock
    data_in                : in  std_logic_vector(WORD_WIDTH-1 downto 0);
    data_valid_in          : in  std_logic; -- Valid data flag
    ttc_insertion          : in  std_logic; -- User controlled pad insertion flag
    pad_in                 : in  std_logic; -- RX FIFO empty flag
    pause_read_in          : in  std_logic; -- Sync Gearbox pause flag
    inject_sof_padding     : out std_logic; -- FIFO read disable flag during SoF insertion
    data_out               : out std_logic_vector(WORD_WIDTH downto 0); -- 64-bit data out
    header_out             : out std_logic_vector(5 downto 0) -- 2-bit header out (4 MSBs reserved)
    );
end add_padding_and_prbs_data;


architecture BEHAVIORAL of add_padding_and_prbs_data IS

    signal SEL                  : std_logic_vector(4 downto 0);
    signal trailer              : std_logic;
    signal crc_valid            : std_logic;
    signal crc_reset            : std_logic:='1';
    signal crc_word             : std_logic_vector(31 downto 0);
    signal data_valid_shreg     : std_logic_vector(3 downto 0):="0000";
    signal start_of_packet      : std_logic;
    signal data_valid_shreg2    : std_logic_vector(3 downto 0):="0000";
    signal pause_read_in_reg    : std_logic;
    signal pause_read_in_reg2   : std_logic;
    signal ttc_insertion_int    : std_logic;
    signal simple_counter       : UNSIGNED(7 downto 0):=x"00";
    signal ttc_insertion_i      : std_logic;

```

```

BEGIN


--=========================================================================================
--================================= Framing Protocol ======================================
--=========================================================================================

----------------------------------------------------------------------------------------|
-- The Framing Protocol sends a Start of Frame word before the first valid data word and |
-- a CRC word followed by a End of Frame word after the last valid data word. A Padding  |
-- words are injected when the FIFO is empty to compensate for the frequency difference  |
-- of the two clock domains                                                              |
----------------------------------------------------------------------------------------|
framing_protocol:if PROTOCOL_SEL = "FRAMING" GENERATE

data_valid_shreg(0) <= data_valid_in;

-- Shift data valid register only on FIFO read
process(clk)
  begin
    if rising_edge(clk) then
        if pause_read_in = '0' and pad_in = '0' then
            data_valid_shreg(3 downto 1)  <= data_valid_shreg(2 downto 0);
        end if;
    end if;
end process;


crc_valid       <= '1' when data_valid_shreg = "1110" else '0'; -- inject CRC at the end of frame
trailer         <= '1' when data_valid_shreg = "1100" else '0'; -- inject end of packet at the End of frame
start_of_packet <= '1' when data_valid_shreg = "0001" else '0'; -- inject Start of packet at the Start of frame


-- Pause FIFO read to inject Start of Frame
inject_sof_padding <= start_of_packet;


SEL <= data_valid_in & pad_in & crc_valid & trailer & start_of_packet;

-- This process shifts the data
-- when pad_in is '1' we inject a padding word for clock compensation
-- else when data valid bit is '0' we send IDLEs
--      when data valid bit is '1' we send DATA
--      we inject the SoF word before the first DATA word
--      and the CRC and EoF after the last DATA word
process(SEL,data_in)
begin
    case SEL is
        when "01000"|"11000"|"01100"|"01010"|"11100"|"11010" => -- send CDC PAD word (1st priotity)
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= x"78F7F7F7F7F7F7F7";
        when "00000" => -- send IDLEs
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= x"5555555555BCBCBC";
        when "00001"|"01001"|"11001"|"10001" => -- send Start Of Frame
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= x"FBaaaaaaaaaaaaaa";
        when "00100" => -- send CRC
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0) <= data_in(WORD_WIDTH-1 DOWNTO 0);
        when "00010" => -- send End of Frame (Trailer)
                header_out <= "000010";
                data_out(WORD_WIDTH-1 DOWNTO 0) <= x"FDaaaaaaaaaaaaaa";
        when "10000" => -- send DATA
            header_out <= "000001";
            data_out(WORD_WIDTH-1 DOWNTO 0) <= data_in(WORD_WIDTH-1 DOWNTO 0);
        when others => -- send Error Code
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= x"FE1e1e1e1e1e1e1e";
    end case;
end process;


data_out(WORD_WIDTH) <= data_valid_in;

END GENERATE;

--================================ END FRAMING Generator =====================================


--===========================================================================================
--================================ SINGLE WORD Protocol =====================================
--===========================================================================================

one_word:if PROTOCOL_SEL = "ONE_WORD" GENERATE
-----------------------------------------------------------------------------------------
```

```
-- The one word protocol sends only two kinds of words. The Padding word when the FIFO  |
-- is empty and the TTC word containing the CRC checksum when required. The TTC word is |
-- sent as a PAD word but the frequency is controlled by the user. Special care must be |
-- taken to avoid sending more TTC/PAD words than required by the domains frequency  |
-- difference.                                                                          |
-----------------------------------------------------------------------------------------

-- Emulate the user driven TTC injection signal using a simple counter and changing the
-- reset value and the counter width register.
process (clk)
begin
    if rising_edge(clk) then
      if ( to_integer(unsigned(simple_counter)) > 127 ) then
          simple_counter <= (others => '0');
      else
          simple_counter <= simple_counter + 1;
      end if;
    end if;
end process;

data_valid_shreg(0) <= simple_counter(7);

-- Shift data valid register only on FIFO read
process(clk)
  begin
    if rising_edge(clk) then
        if pause_read_in = '0' and pad_in = '0' then -- ONLY shift on FIFO read
            data_valid_shreg(3 downto 1)  <= data_valid_shreg(2 downto 0);
        end if;
    end if;
end process;

-- Generate and inject the CRC checksum when TTC inject is active
ucrc_inst: entity work.ucrc_par
    generic map (
      POLYNOMIAL => "00000100110000010001110110110111",
      INIT_VALUE => "11111111111111111111111111111111",
      DATA_WIDTH => 64,
      SYNC_RESET => 1)
    port map(
      clk_i     => clk,
      rst_i     => crc_reset,
      clken_i   => ttc_insertion_int,
      data_i    => data_in,
      match_o   => open,
      crc_o     => crc_word);


ttc_insertion_int <= '0' when data_valid_shreg = "0001" else '1';
crc_reset <= '1' when data_valid_shreg = "0010" else '0';


SEL(1 downto 0) <= pad_in & not ttc_insertion_int;

process(SEL,data_in)
begin
    case SEL(1 downto 0) is
        when "00" => -- send data
            header_out <= "000001";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= data_in(WORD_WIDTH-1 DOWNTO 0);
        when "10" => -- send CDC Padding Word
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= x"78F7F7F7F7F7F7F7";
        when "01"|"11" => -- send CRC + Header as Padding
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= x"FB000000" & crc_word;
        when others => -- send Error Code
            header_out <= "000010";
            data_out(WORD_WIDTH-1 DOWNTO 0)<= x"FE1e1e1e1e1e1e1e";
    end case;
end process;


inject_sof_padding   <= not ttc_insertion_int;
data_out(WORD_WIDTH) <= data_valid_in;

END GENERATE;

END BEHAVIORAL;

--================================ END SINGLE WORD Generator ===================================
```

A.2 Hermes protocol error checker and alignment code

The “*data_quality_with_idle*” module is responsible to capture errors during transmission. The header and the coding words are checked for illegal values. This module contains also the word alignment logic by sending a bit shift signal when a stream of more then 16 errors is captured.

```
-- ============================================================================================|
-- This module checks both the header and the coding words for errors and aligns the incoming    |
-- words. The alignment is achieved by shifting the incoming data words when we receive a stream |
-- of errors.                                                                                    |
-- ============================================================================================|

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.std_logic_unsigned.all;
library UNISIM;
use UNISIM.VComponents.all;
USE IEEE.std_logic_misc.all; -- contains OR_REDUCE & AND_REDUCE functions among others

entity data_quality_with_idle is
  generic ( WORD_WIDTH   : natural := 64;
            PROTOCOL_SEL : string:="ONE_WORD"); -- "FRAMING" or "ONE_WORD"
  Port (
         reset_all_in       : in STD_LOGIC;
         rx_usrclk2_in      : in STD_LOGIC;
         rx_active_in       : in STD_LOGIC;
         rxdatavalid_in     : in  STD_LOGIC_VECTOR(1 DOWNTO 0);
         rxdata_in          : in  STD_LOGIC_VECTOR(WORD_WIDTH-1 DOWNTO 0);
         rxdata_header_in   : in STD_LOGIC_VECTOR(5 DOWNTO 0);
         rxgearboxslip_out  : out STD_LOGIC:='0';
         prbs_match_out     : out STD_LOGIC:='0';
         data_good          : out STD_LOGIC:='0'
       );

end data_quality_with_idle;


architecture RTL of data_quality_with_idle is

component reset_synchronizer
  Port(
        clk_in      : in STD_LOGIC;
        rst_in      : in STD_LOGIC;
        rst_out     : out STD_LOGIC
    );
end component;

component gtwizard_ultrascale_0_prbs_any
  generic (
        CHK_MODE    : INTEGER := 1;
        INV_PATTERN : INTEGER := 1;
        POLY_LENGHT : INTEGER := 31;
        POLY_TAP    : INTEGER := 28;
        NBITS       : INTEGER := 64
    );
  port (
        RST         : in  STD_LOGIC;
        CLK         : in  STD_LOGIC;
        DATA_IN     : in  STD_LOGIC_VECTOR(WORD_WIDTH-1 DOWNTO 0);
        EN          : in  STD_LOGIC;
        DATA_OUT    : out STD_LOGIC_VECTOR(WORD_WIDTH-1 DOWNTO 0)
    );
end component;


-- This function reverses the bits of a vector
FUNCTION bit_reverse(s1:std_logic_vector) return std_logic_vector is
  variable rr : std_logic_vector(s1'high downto s1'low);
  begin
    for ii in s1'high downto s1'low loop
        rr(ii) := s1(s1'high-ii);
    end loop;
  return rr;
end bit_reverse;


signal reset                    : STD_LOGIC;
```

```vhdl
signal reset_sync               : STD_LOGIC;
signal rxdata_int               : STD_LOGIC_VECTOR (WORD_WIDTH-1 DOWNTO 0);
signal data_check               : STD_LOGIC;
signal crc_en                   : STD_LOGIC;
signal header_match_i           : STD_LOGIC;
signal rxdata_header_i          : STD_LOGIC_VECTOR(5 DOWNTO 0);
signal data_good_i              : STD_LOGIC;
signal prbs_checker_enable      : STD_LOGIC;
signal rxgearboxslip_ctr_int    : STD_LOGIC_VECTOR (3 DOWNTO 0) := (OTHERS=>'0');
signal prbs_error_vector        : STD_LOGIC_VECTOR (WORD_WIDTH-1 DOWNTO 0);
signal prbs_match_out_i         : STD_LOGIC;
signal padding_word_flag        : STD_LOGIC;
signal rxdatavalid_in_l1        : STD_LOGIC;
signal prbs_error_vector_test   : STD_LOGIC_VECTOR (WORD_WIDTH-1 DOWNTO 0);



BEGIN


-- Synchronize the "reset" signal into the txusrclk2 clock domain
reset <= reset_all_in OR (NOT rx_active_in);

example_checking_reset_synchronizer_inst: reset_synchronizer
  PORT MAP(
            clk_in  => rx_usrclk2_in,
            rst_in  => reset,
            rst_out => reset_sync
        );


rxdata_header_i <= rxdata_header_in;


-- Bit-reverse the incoming data, since gearbox modes transmit data MSb first.
rxdata_int <= bit_reverse(rxdata_in);


-- This process uses the error checker indicator as feedback to periodically pulse the "rxgearboxslip"
-- until word alignment is achieved and no errors are received.
process(rx_usrclk2_in, reset_sync)
begin
    if reset_sync = '1' then
        rxgearboxslip_ctr_int <= (OTHERS=>'0');
        rxgearboxslip_out     <= '0';
    elsif rising_edge(rx_usrclk2_in) then
        if data_check = '1' then
            if data_good_i = '0' then
                rxgearboxslip_ctr_int <= rxgearboxslip_ctr_int + '1';
                rxgearboxslip_out     <= and_reduce(rxgearboxslip_ctr_int);
            else
                if rxgearboxslip_ctr_int /= "0000" then
                    rxgearboxslip_ctr_int <= rxgearboxslip_ctr_int - '1';
                end if;
                rxgearboxslip_out <= '0';
            end if;
        end if;
    end if;
end process;


-- The error checking process always monitors the incoming data for illegal
-- header values. Furthermore, for both "FRAMING" and "ONE_WORD" protocols, it
-- checks for the correct transmition of the coding words.

--===================================== FRAMING Checker =====================================

framing_protocol: if PROTOCOL_SEL = "FRAMING" GENERATE

process(rx_usrclk2_in, reset_sync)
    begin
        if reset_sync = '1' then
            data_good_i <= '0';
            data_check  <= '0';
            crc_en      <= '0';
        elsif rising_edge(rx_usrclk2_in) then
            if rxdata_header_i(1 DOWNTO 0) = "01" then
                data_check  <= '0';
                data_good_i <= '0';
                crc_en      <= '1';
            elsif rxdata_header_i = "10"  and rxdatavalid_in(0) = '1' then
                data_check <= '1';
                if rxdata_int = x"78F7F7F7F7F7F7F7" then -- Padding
                    data_good_i <= '1';
                    crc_en <= '0';
                elsif rxdata_int = x"5555555555BCBCBC" then -- IDLE
                    data_good_i <= '1';
                    crc_en <= '0';
                elsif rxdata_int = x"FBaaaaaaaaaaaaaa" then -- SOF
                        data_good_i <= '1';
                        crc_en <= '0';
                elsif rxdata_int = x"FDaaaaaaaaaaaaaa" then -- EOF
                        data_good_i <= '1';
                        crc_en <= '0';
                elsif rxdata_int(63 downto 32) = x"99000000" then -- CRC
                    data_good_i <= '1';
                    crc_en <= '1';
                else
                    data_good_i <= '0';
                    crc_en <= '0';
                end if;
            else
                data_check  <= '1';
                data_good_i <= '0';
            end if;
        end if;
end process;

padding_word_flag <= '1' when (rxdata_int = x"78F7F7F7F7F7F7F7") and (rxdata_header_i = "10") else '0';

END GENERATE;
--================================ END of FRAMING Checker ====================================


--==================================== ONE_WORD Checker ====================================

one_word: if PROTOCOL_SEL = "ONE_WORD" GENERATE

process(rx_usrclk2_in, reset_sync)
    begin
        if reset_sync = '1' then
            data_good_i <= '0';
            data_check  <= '0';
            crc_en      <= '0';
        elsif rising_edge(rx_usrclk2_in) then
            if rxdata_header_i(1 DOWNTO 0) = "01" then
                data_check  <= '0';
                data_good_i <= '0';
                crc_en      <= '1';
            elsif rxdata_header_i = "10"  and rxdatavalid_in(0) = '1' then
                data_check   <= '1';
                if rxdata_int = x"78F7F7F7F7F7F7F7" then -- Padding
                    data_good_i <= '1';
                elsif rxdata_int(63 downto 56) = x"FB" then -- CRC
                    data_good_i <= '1';
                    crc_en      <= '1';
                else
                    data_good_i <= '0';
                    crc_en      <= '0';
                end if;
            else
                data_check  <= '1';
                data_good_i <= '0';
            end if;
        end if;
end process;

padding_word_flag <= '1' when (rxdata_int = x"78F7F7F7F7F7F7F7" or rxdata_int(63 downto 56) = x"FB") and rxdata_header_i = "10" else '0';

END GENERATE;

--================================ END of ONE_WORD Checker ====================================


process (rx_usrclk2_in)
    begin
        if rising_edge(rx_usrclk2_in) then
             rxdatavalid_in_l1 <=  rxdatavalid_in(0);
        end if;
end process;

data_good <= data_good_i or not(data_check) or not(rxdatavalid_in_l1);

end RTL;
```

A.3 uGMT new ghost-busting code

The modified version of the uGMT ghost-busting module for the Kalman track address encoding.

```vhdl
library IEEE;
use IEEE.NUMERIC_STD.all;
use IEEE.STD_LOGIC_1164.all;
use work.GMTTypes.all;

entity GhostCheckerUnit_BMTF is
  port (mu1    : in  TBMTFTrackAddress;
        qual1  : in  unsigned(3 downto 0);
        mu2    : in  TBMTFTrackAddress;
        qual2  : in  unsigned(3 downto 0);
        ghost1 : out std_logic;
        ghost2 : out std_logic;
        clk    : in  std_logic);
end GhostCheckerUnit_BMTF;

architecture Behavioral of GhostCheckerUnit_BMTF is

BEGIN

P : process(mu1, mu2, qual1, qual2)
    variable matchedStation : boolean := false;  -- whether a track segment was shared between two tracks
    begin

    matchedStation := false;

    for station in 0 to 2 loop
-- If candidates are in same wheel on same side
-- (modified for Kalman address scheme).
      if ((mu2.detectorSide = mu1.detectorSide) and (mu2.wheelNo = mu1.wheelNo)) then
        if (mu2.stationAddresses(station) = X"A" and mu1.stationAddresses(station) = X"8") or
           (mu2.stationAddresses(station) = X"B" and mu1.stationAddresses(station) = X"9") or
           (mu2.stationAddresses(station) = X"8" and mu1.stationAddresses(station) = X"C") or
           (mu2.stationAddresses(station) = X"9" and mu1.stationAddresses(station) = X"D") or
           (mu2.stationAddresses(station) = X"2" and mu1.stationAddresses(station) = X"0") or
           (mu2.stationAddresses(station) = X"3" and mu1.stationAddresses(station) = X"1") or
           (mu2.stationAddresses(station) = X"0" and mu1.stationAddresses(station) = X"4") or
           (mu2.stationAddresses(station) = X"1" and mu1.stationAddresses(station) = X"5") then
          matchedStation := true;
        end if;

-- If candidates are in same side and candidate 2 is one wheel in front of candidate 1.
-- (modified for Kalman address scheme).
      elsif (mu2.detectorSide = mu1.detectorSide) and
            ((mu2.wheelNo = 0 and mu1.wheelNo = 1) or
             (mu2.wheelNo = 1 and mu1.wheelNo = 2)) then
        if (mu2.stationAddresses(station) = X"A" and mu1.stationAddresses(station) = X"0") or
           (mu2.stationAddresses(station) = X"B" and mu1.stationAddresses(station) = X"1") or
           (mu2.stationAddresses(station) = X"8" and mu1.stationAddresses(station) = X"4") or
           (mu2.stationAddresses(station) = X"9" and mu1.stationAddresses(station) = X"5") then
          matchedStation := true;
        end if;

-- If candidates are in same side and candidate 2 is one wheel behind candidate 1.
-- (modified for Kalman address scheme).
      elsif (mu2.detectorSide = mu1.detectorSide) and
            ((mu2.wheelNo = 1 and mu1.wheelNo = 0) or
             (mu2.wheelNo = 2 and mu1.wheelNo = 1)) then
        if (mu2.stationAddresses(station) = X"2" and mu1.stationAddresses(station) = X"8") or
           (mu2.stationAddresses(station) = X"3" and mu1.stationAddresses(station) = X"9") or
           (mu2.stationAddresses(station) = X"0" and mu1.stationAddresses(station) = X"C") or
           (mu2.stationAddresses(station) = X"1" and mu1.stationAddresses(station) = X"D") then
          matchedStation := true;
        end if;

--  If one muon in 0+ and one muon in 0- (0+ and 0- are physically the same
--  wheel) (only legacy BMTF - wheel 0 is not split in Kalman algorithm).
--      elsif (mu2.detectorSide /= mu1.detectorSide) and
--            (mu2.wheelNo = 0 and mu1.wheelNo = 0) then
--          if (mu2.stationAddresses(station) = X"8" and mu1.stationAddresses(station) = X"A") or
--             (mu2.stationAddresses(station) = X"9" and mu1.stationAddresses(station) = X"B") or
--             (mu2.stationAddresses(station) = X"C" and mu1.stationAddresses(station) = X"8") or
--             (mu2.stationAddresses(station) = X"D" and mu1.stationAddresses(station) = X"9")then
--             matchedStation := true;
--          end if;

--  If candidate 1 is at -1 and candidate 2 is at 0
-- (only Kalman).
      elsif (mu2.detectorSide /= mu1.detectorSide) and
                (mu2.wheelNo = 0 and mu1.wheelNo = 1) then
              if (mu2.stationAddresses(station) = X"A" and mu1.stationAddresses(station) = X"0") or
                 (mu2.stationAddresses(station) = X"B" and mu1.stationAddresses(station) = X"1") or
                 (mu2.stationAddresses(station) = X"8" and mu1.stationAddresses(station) = X"4") or
                 (mu2.stationAddresses(station) = X"9" and mu1.stationAddresses(station) = X"5")then
                 matchedStation := true;
              end if;

-- If candidate 1 is at 0 and candidate 2 is at -1
-- (only Kalman).
      elsif (mu2.detectorSide /= mu1.detectorSide) and
                (mu2.wheelNo = 1 and mu1.wheelNo = 0) then
              if (mu2.stationAddresses(station) = X"2" and mu1.stationAddresses(station) = X"8") or
                 (mu2.stationAddresses(station) = X"3" and mu1.stationAddresses(station) = X"9") or
                 (mu2.stationAddresses(station) = X"0" and mu1.stationAddresses(station) = X"C") or
                 (mu2.stationAddresses(station) = X"1" and mu1.stationAddresses(station) = X"D")then
                 matchedStation := true;
              end if;
      end if;
    end loop;

-- If the muons are 'far enough' apart we don't check the LUT output.
    if matchedStation = true then
      if qual1 > qual2 then
        ghost1 <= '0';
        ghost2 <= '1';
      else
        ghost1 <= '1';
        ghost2 <= '0';
      end if;
    else
      ghost1 <= '0';
      ghost2 <= '0';
    end if;
  end process;

END BEHAVIORAL;
```

Appendix B

Appendix B: The *IPBus* control system for xTCA hardware

This appendix describes the IP-based protocol. The IPbus protocol is a simple packet-based control protocol for reading and modifying memory-mapped resources within FPGA-based IPaware hardware. A tightly-integrated suite of IPbus software and firmware components which can be used to construct reliable, scalable, high-performance control systems for controlling hardware devices. It assumes the existence of a virtual bus with 32-bit word addressing and 32-bit data transfer. The choice of 32-bit data width is fixed in this protocol. This IPbus suite is used to control the xTCA off-detector electronics in the upgrades for Run 2 and Run 3 of the CMS experiment [85].

The IPbus protocol defines the following operations:

- **Read**: A read of user-definable depth. Two types are defined: address-incrementing (for multiple continuous registers in the address space) and non-address-incrementing (for a port or FIFO).
- **Write**: A write of user-definable depth. As with reads, two types of write are defined: incrementing and non-incrementing.
- **Read-Modify-Write bits (RMWbits)**: An atomic bit-masked write, defined as X := (X&A)|B. This allows one to efficiently set/clear a subset of bits within a 32-bit register.
- **Read-Modify-Write sum (RMWsum)**: An atomic increment operation, defined as X := X + A, which is useful for adding values to a register (or subtracting, using two's complement).

The protocol is transactional — for each read, write or RMW operation, the IPbus client (typically software) sends a request to the IPbus device; the device then sends back a response message containing an error code (equal to 0 for a successful transaction), followed by return data in case of reads. In order to minimise latency, multiple transactions can be concatenated into a single IPbus packet. Since the IPbus device implementation must have a low FPGA resource usage, UDP has been chosen as the transport protocol and a reliability mechanism over UDP was included, through which the client can correct for any packet loss, duplication or reordering.

The IPbus software and firmware suite consists of the IPbus firmware, that implements the IPbus protocol within end-user hardware, the Control Hub Software application that mediates simultaneous hardware access from multiple μHAL clients, and implements the IPbus reliability mechanism over UDP and the μHAL, C++ and Python end-user programming interface for writes, reads and RMW operations (see ??).

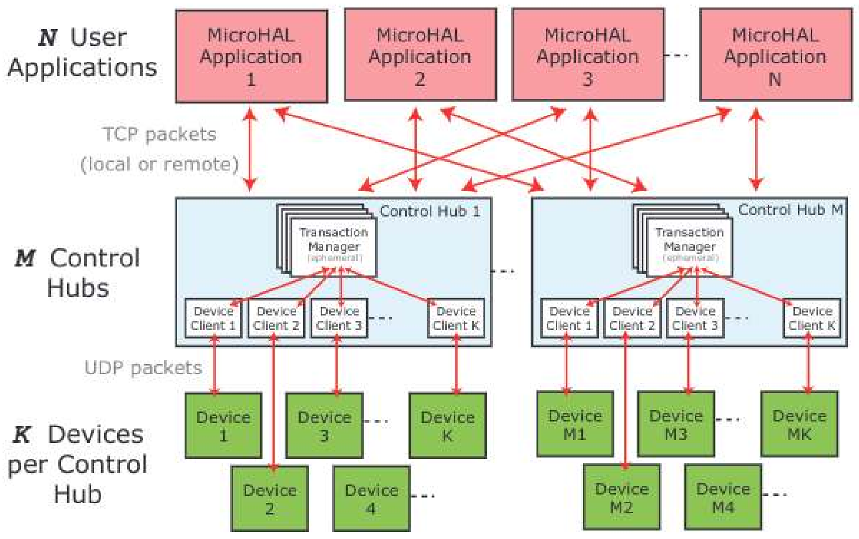


Figure B.0.1: Example of a large-scale IPbus system, with multiple MicroHAL end-users applications communicating with many crates of hardware via their respective Control Hubs [85].

The IPbus firmware is implemented in VHDL, and is well modularised, with a clean separation between the Ethernet interface, protocol decoders, and bus master. This allows any of these components, to be easily swapped out if desired; for instance, if a different transport protocol other than UDP was preferred. Furthermore, a general-purpose interface to the bus master, allows arbitrated control of the IPbus, from a number of sources other than the usual Ethernet/UDP route. This allows the IPbus to be controlled by a microcontroller or external SPI/I2C interface, providing the convenience of being able to set the IP address of any individual IPbus host via IPMI.

Control Hub Control is a software instance that forms a single point of contact with what may be one or more crates of IPbus-enabled hardware. It mediates transaction requests coming from several end-user software processes into orderly queues for dispatch to the individual hardware devices, and then passes the transaction responses back to the originator. Users software processes can be on the same physical host as the Control Hub, or on remote machines communicating across a user network, with the Control Hub acting as a middle-man between these processes and the hardware devices connected on a private hardware network. It is largely analogous to, but much more flexible than a, VME crate controller and its associated driver software

μHAL is the Hardware Access Library (HAL) providing an end-user C++/Python API for IPbus reads, writes and RMW transactions. In μHAL each device's register layout is specified by XML files. Each node of the XML tree represents either a single register, block RAM, FIFO, or a collection of these; the nodes in one file can reference other address files, such that the interfaces to repeated instances of a firmware module can be generated with minimal copy-paste of address file contents. This enables the user to write control software in a manner that intuitively mirrors the modular, hierarchical structure of large firmware designs. The μHAL interface can communicate with the hardware either directly over UDP or via a Control Hub.

MicroHAL is a C++ hardware access library (HAL) for IPbus, allowing end-users to write applications to control their IPbus-enabled hardware. It provides two main packages; the IPbus Client and Redwood. The IPbus Client code encapsulates the implementation of the protocol,whilst Redwood provides higher-level functionality that enables the end-user to write software in a manner that logically and intuitively mirrors the structure of the firmware.

The IPbus-specific information is contained within the payload of the UDP packet, which in turn is wrapped within an IP packet and an Ethernet packet (see Figure B.0.2).

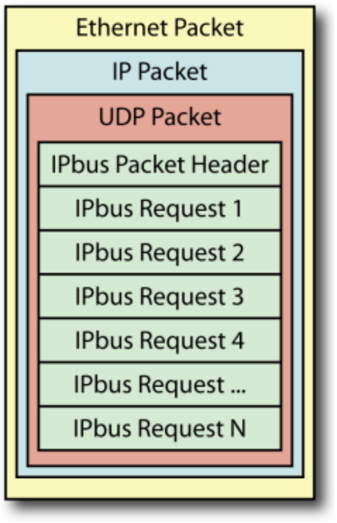


Figure B.0.2: The IPBus packet. All IPbus requests are contained within a the payload of a UDP packet, which in turn, is wrapped within an IP packet and an Ethernet packet.

The request and reply of an IPbus packet always begins with a 32-bit header followed either a Control, a Status, or a Re-send packet. The Control packet is the concatenation of a control packet header and one or many IPbus transactions (Read, Write, Read-Modify-Write bit, Read-Modify-Write sum). The Status packet is necessary for the reliability mechanism, as well as being useful for debugging the target's recent activity. Finally, the re-send request packet is necessary for the reliability mechanism, and its purpose is to trigger a re-send of one of the target's recent outbound control packets [85].

Acronyms

ALICE A Large Ion Collider Experiment. 35, 36

AMC Advanced Mezzanine Card. 76–79, 90

AMC13 Advanced Mezzanine Card 13. 78

ASIC Application Specific Integrated Circuit. 66, 69, 73, 93, 123

ATLAS A Toroidal LHC ApparatuS. 34–36, 175

BER Bit Error Ratio. 64, 126, 127, 132, 144, 159, 162, 183, 186

BMTF Barrel Muon Track Finder. 78, 84–87, 96–100, 103, 106, 107, 109–114, 117, 118, 121, 122, 166, 179–182

BPix Barrel Pixel. 48

BRAM Block Random Access Memory. 63, 109, 111, 156, 158, 185

CaloL1 Calorimeter Trigger Layer-1. 80, 81

CaloL2 Calorimeter Trigger Layer-2. 78, 81, 82, 84, 179

CCC Cern Control Center. 32, 33

CCU Communication and Control Units. 47

CDAQ Central Data Acquisition. 78

CDC Clock Domain Crossing. 149, 152, 156, 185

CDR Clock and data recovery. 126–128, 183

CERN Conseil Européen pour la Recherche Nucléaire. 32, 34, 37, 41, 61, 175

CLB Configurable Logic Block. 61–63, 66, 177

CMOS Complementary Metal-Oxide-Semiconductor. 48, 49

CMS Compact Muon Solenoid. 27, 28, 34–36, 41–47, 49, 51–56, 61, 63, 73–80, 82, 83, 90, 93–95, 97–99, 102, 107, 109, 111, 119–121, 129–132, 135, 138, 144, 149, 159, 165–167, 175–180, 182, 183

COMPASS Common Muon and Proton Apparatus for Structure and Spectroscopy. 32

CPLD Complex Programmable Logic Device. 66

CPPF Concentration Pre-Processing and Fan-out. 79, 84, 87, 89, 179

CPU Central Processing Unit. 62

CRC Cyclic Redundancy Check. 129, 131, 136, 139, 140, 143, 146–149, 152–154, 158–160, 184–186

CSC Cathode Strip Chamber. 53, 54, 56–58, 84, 87, 88, 176, 177, 179

CTP7 Calorimeter Trigger Processor 7. 77, 81, 83, 178

CU Cooling Unit. 76

CuOF Copper-to-Optical Fiber. 85

DAQ Data Acquisition. 78, 93, 96, 111, 112, 181

DCM Digital Clock Manager. 62, 63, 177

DSP Digital Signal Processing. 62, 63, 66, 67, 109–111, 123, 124, 177

DT Drift Tube. 53–55, 57, 58, 84–86, 88, 89, 97–100, 102–104, 106, 122, 166, 176, 177, 179, 180

EB ECAL Barrel. 50

ECAL Electromagnetic Calorimeter. 49–52, 80, 82, 176, 178

EMTF Endcap Muon Track finder. 78, 79, 84, 87, 88, 96, 179

EoF End of Frame. 139, 152–154, 158, 159, 184, 185

ETTF Eta Track Finder. 103–105

FEC Front-End Controllers. 47

FED Front-End Driver. 47

FPGA Field-Programmable Gate Array. 27, 28, 61–68, 71, 73, 76–79, 86, 90, 91, 93, 94, 100, 103, 109–113, 115, 123–125, 132, 149, 160, 166, 167, 177, 181, 183, 195, 196

FPix Forward Pixel. 48

GbE Gigabit-Ethernet. 78

GEM Gas Electron Multiplier. 58, 59, 177

GMT Global Muon Trigger. 96, 122

GOL Gigabit Optical Link. 79

GP Golden Pattern. 89

GTT Global Track Trigger. 95, 96

HB HCAL Barrel. 52, 80, 176

HCAL Hadronic Calorimeter. 52, 53, 80, 82, 178

HDL Hardware Description Language. 67–69, 71, 177

HDLs Hardware Description Languages. 68, 69

HE HCAL Endcap. 52, 80, 176

HF HCAL forward. 52, 80, 176

HG-CAL High Granularity Endcap Calorimeter. 94

HL-LHC High Luminosity LHC. 27, 28, 37, 39, 58, 76, 94, 106, 107, 129, 175

HLS High Level Synthesis. 28, 70, 71, 107, 114, 166, 181

HLT High Level Trigger. 27, 47, 73, 74, 90, 91, 93, 94, 179

HO HCAL Outer. 52, 85, 86, 176, 179

IBS Intrabeam Scattering. 35

IC Integrated Circuit. 61

IEEE Institute of Electrical and Electronics Engineers. 69, 133, 136, 147

IP Interaction Point. 33, 41, 46, 48, 51, 52, 149, 150, 184

iRPC improved RPC. 58, 59, 177

L1-BMT Layer-1 Barrel Muon Trigger. 123, 124, 183

L1A Level-1 Accept. 74, 75, 78, 96, 178

L1T Level-1 Trigger. 27, 28, 73–77, 79, 90, 91, 93–95, 178, 180

LC Lucent Connector. 163, 186

LCDS Low Current Differential Signal. 49

LCT Local Charged Track. 87, 88

LEIR Low Energy Ion Ring. 32, 175

LEP Large Electron-Positron Collider. 31

LHC Large Hadron Collider. 27, 28, 31–38, 41–43, 46, 49, 51, 53, 58, 73, 75, 77, 80, 81, 89, 90, 130–132, 149, 150, 153, 165, 175, 178

LHCb Large Hadron Collider beauty. 35, 36

LHCf Large Hadron Collider forward. 35, 36

LINAC2 Linear accelerator 2. 31, 32, 175

LINAC3 Linear accelerator 3. 32

LSFR Linear Feedback Shift Register. 148

LUT Look-Up Table. 62, 63, 86, 87, 100, 103, 111

LVDS Low Voltage Differential Signal. 49, 90

MB Muon Barrel. 53

MCH uTCA Carrier Hub. 76–78

ME Muon Endcap. 53

MGT Multi-Gigabit Transceiver. 64, 65, 123–125, 130, 131, 142, 143

MMC Module Management Controller. 78

MoEDAL Monopole and Exotics Detector at the LHC. 35, 36

MP7 Master Processor 7. 78, 82, 86, 89–91, 111, 112, 181

MTF7 Modular Track Finder 7. 78, 87, 88

MTP Multi-fiber Termination Push-on. 163, 186

MTTF Mean Time To Failure. 134

OMTF Overlap Muon Track Finder. 78, 79, 84–86, 88, 89, 96, 103, 117, 121, 179

P5 Point 5. 41, 110, 111, 118

PCS Physical Coding Sublayer. 64, 65

PF Particle Flow. 93, 95

PHTF Phi Track Finder. 99, 104, 105, 180, 181

PLD Programmable Logic Devices. 69

PLL Phase-locked loop. 78

PM Power Module. 76

PMA Physical Medium Attachment. 65, 135

PRBS PseudoRandom Binary Sequence. 144, 145, 184

PS Proton Synchrotron. 32, 175

PSB Proton Synchrotron Booster. 31, 32, 175

RAM Read Access Memory. 62, 63

RF Radio Frequency. 35, 175

ROC Read-Out Chip. 48, 49

RPC Resistive Plate Chamber. 53–55, 57, 58, 79, 84–89, 97, 122, 176, 177, 179

RTL Register Transfer Level. 67, 71

SC Super-Crystals. 51

SoC System-on-Chip. 77, 123, 183

SoF Start of Frame. 139, 152–154, 158, 159, 161, 184–186

SPL Software Programming Language. 67

SPS Super Proton Synchrotron. 32, 175

SST Silicon Strip Tracker. 46, 47, 176

TBM Token Bit Manager. 48, 49

TCDS Timing and Control Distribution System. 78, 96

TEC Tracker End-Cap. 46

TIB Tracker Inner Barrel. 46

TMT Time-Multiplexed Trigger. 80–83, 178

TOB Tracker Outer Barrel. 46

TOTEM TOTal cross section, Elastic scattering and diffraction dissociation Measurement at the LHC. 35, 36

TP Trigger Primitive. 84, 85, 97

TPG Trigger Primitive Generator. 74, 80

uGMT Global Muon Trigger. 78, 83–86, 88–91, 103, 106, 107, 111, 117–119, 179, 181, 182

uGT Global Trigger. 78, 82, 84, 89–93, 179

uTCA Micro Telecommunications Computing Architecture. 76–79, 81, 86, 90, 178

VHDCI Very High Density Cable Interconnect. 90

VHDL Very-high-speed-integrated-circuit Hardware Description Language. 28, 69, 70, 91, 113, 114, 118, 177, 181

VLSI Very Large Scale Integration. 68, 69

WBM Web-Based Monitoring. 119, 182