Evaluation of an Associative Memory and FPGA-based System for the Track Trigger of the CMS-Detector

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der Fakultät
Elektrotechnik und Informationstechnik
des Karlsruher Institut für Technologie (KIT)
genehmigte

DISSERTATION

von

M.Sc. Christian Amstutz

geb. in: Baden, Schweiz

Tag der mündlichen Prüfung:	22. Juli 2016
Hauptreferent:	Prof. Dr. rer. nat. Marc Weber
Korreferent:	Prof. Dr.-Ing. Dr. h. c. Jürgen Becker

Abstract

By 2025, the luminosity of the particle beam at the largest and most powerful particle accelerator in the world—the Large Hadron Collider (LHC)—will be increased even further. The increased luminosity will cause more concurrently colliding particles at the center of the Compact Muon Solenoid (CMS) experiment at the LHC. To cope with the new conditions, a track trigger will be integrated to the CMS experiment. The CMS track trigger preprocesses data from the outer silicon tracker and delivers the track parameters of the particles to the first trigger level. A track trigger at the first trigger level has never been included in any particle physics experiment before, as the requirements for such an electronics system are enormous. In particular, the input data rate of the CMS track trigger will be nearly 100 Tbit/s and the total processing time must not exceed 4 µs. To fulfill these extraordinary requirements, a unique, heterogeneous embedded system is required.

In this thesis, a concept for a system-level simulation of the CMS track trigger is presented. The system simulation facilitates the evaluation the CMS track trigger electronics as a whole. It includes everything from the electronics of the detector modules recording the particle tracks up to the processing board implementing the track processing algorithms. The system simulation provides the system designer with three features: (1) Figures of merit—such as latencies, bandwidths and buffer sizes—can be estimated. (2) Different system architectures can be evaluated quickly. (3) The system simulation can serve as a test bench for algorithms and Field-Programmable Gate Arrays (FPGA) firmware. To achieve realistic results, the input data are taken from a simulation of the CMS experiment detector.

One of the investigated concepts for the CMS track trigger consists of up to 48 large crates with hundreds of electronic boards. The actual processing is performed on a board with FPGAs and Associative Memory (AM) chips specifically designed for track finding. Some of these boards were produced and tested at Karlsruhe Institute of Technology. One of the independent processors of the CMS track trigger based on AMs has been modeled according to the system simulation concepts. It is shown that the system simulation is able to simulate an essential part of the proposed CMS track trigger system.

Many components are instantiated multiple times within the simulation of the CMS track trigger. Usually, the components are arranged in regular structures, e.g. two- or three-dimensional arrays. To support the modeling and configuration of such regular structures, a SystemC library has been developed.

Additionally, an independent cost estimate shows that the CMS track trigger based on AMs can realistically be built for the 11.9 million euro that are budgeted. Extrapolated to the year 2022, when the CMS track trigger will be built, even lower costs can be expected.

Zusammenfassung

Im Jahr 2025 wird die Luminosität des Teilchenstrahls am Large Hadron Collider (LHC), dem größten Teilchenbeschleuniger der Welt mit den höchsten Energien, weiter erhöht. Dadurch werden noch mehr Teilchen gleichzeitig im Zentrum des Compact Muon Solenoid (CMS) Experimentes kollidieren. Um unter diesen neuen Bedingungen verwertbare Daten zu liefern, wird erstmals ein Spurtrigger für CMS entwickelt. Dieser verarbeitet die Daten des äußeren Spurdetektors und liefert die Parameter der Teilchenspuren an die erste Triggerstufe von CMS. Da die technischen Anforderungen an ein solches Spurtriggersystem enorm sind, wurde bisher noch nie ein Spurtrigger auf der ersten Triggerstufe eines Teilchenphysikexperimentes eingesetzt. Die Datenrate am Eingang des CMS-Spurtriggers wird beinahe 100 Tbit/s betragen und die Verarbeitungszeit darf 4 µs nicht überschreiten. Um diese außergewöhnlichen Anforderungen zu erfüllen, ist ein einzigartiges, heterogenes eingebettetes System erforderlich.

Diese Dissertation präsentiert eine neu konzeptionierte Simulationsumgebung auf Systemebene für den CMS-Spurtrigger. Die Simulationsumgebung ermöglicht die Evaluation der CMS-Spurtriggerelektronik als Ganzes: von den Modulen mit den Siliziumdetektoren bis zu den Komponenten, welche die Algorithmen zur Spurerkennung ausführen. Die Simulation stellt dem Systementwickler drei Funktionen zur Verfügung: Erstens können Systemeigenschaften wie Latenz, Bandbreite und benötigte Puffergrößen abgeschätzt werden. Zweitens können verschiedene Systemarchitekturen miteinander verglichen werden. Drittens dient die Simulationsumgebung als Testumgebung für Algorithmen und Code, welcher in Field-Programmable Gate Arrays (FPGA) implementiert wird. Um realistische Ergebnisse zu erhalten, werden Daten einer Simulation des CMS-Experimentes als Eingangsdaten der Simulationsumgebung verwendet.

Eines der untersuchten Konzepte für den CMS-Spurtrigger besteht aus bis zu 48 großen Baugruppenträgern mit Hunderten von Platinen. Zur Verarbeitung der Daten werden FPGAs und eigens für die Suche von Teilchenspuren entwickelte Assoziativspeicher genutzt. Prototypen einer Platine mit FPGAs und Assoziativspeicher Chips wurden am Karlsruher Institut für Technologie produziert und getestet. Zusätzlich wurde ein essenzieller Teil des CMS-Spurtriggers mithilfe der neuen Simulationsumgebung simuliert. Durch diese Implementierung wurde aufgezeigt, dass es möglich ist, ein solch großes System in der Simulationsumgebung zu simulieren.

Innerhalb der Simulation werden viele Elemente des CMS-Spurttriggers vielfach instantiiert. Dabei sind die Elemente oft in regelmäßigen Strukturen wie zum Bei-

spiel zwei- oder dreidimensionalen Rastern angeordnet. Eine SystemC-Bibliothek wurde entwickelt, um das Modellieren und Konfigurieren solcher Strukturen zu vereinfachen.

Außerdem wurde eine unabhängige Kostenabschätzung des CMS-Spurtriggers durchgeführt. Diese zeigt, dass die veranschlagten 11,9 Millionen Euro ausreichen, um den auf Assoziativspeicher basierende CMS-Spurtrigger zu bauen. Werden die Werte anhand des Technologiefortschritts auf das Jahr 2022 hochgerechnet, kann sogar mit deutlich niedrigeren Kosten gerechnet werden.

Acknowledgements

First of all, I would like to thank Prof. Dr. Marc Weber for the opportunity to join the Institute for Data Processing and Electronics (IPE) for my PhD research. I also appreciate his efforts to introduce me to the fascinating topics around high-energy physics and CERN. I would also like to thank Prof. Dr.-Ing. Jürgen Becker for being my second supervisor. I thank Matthias Balzer for his supervision as my group leader and the support that I always received for my work.

In general, I would like to thank the people at IPE for their support during my three and a half years at the institute. With many at IPE, I had fruitful discussions about various topics in the field of research. I would like to thank some people especially. Oliver Sander, for his great support in the final phase of this thesis and our discussions about the CMS track trigger that helped me to understand it more deeply. I also thank Denis Tcherniakhovski for his support with practical questions about FPGAs, hardware design and testing. Thomas Schuh and Benedikt Zimmermann, I would like to thank for the time that they spent to explain me some of the secrets in particle physics. With Tanja Harbaum, I liked to share a technical view on the CMS track trigger. At IPE, I also met Matthias Vogelgesang, Lorenzo Rota, Nicholas Tan Jerome and Uroš Stevanović with who I had not only interesting discussions about science and research, but I also enjoyed many hours in my spare time.

I would also like to thank the people from the different groups involved in the CMS track trigger project for providing me information about their work on the CMS track trigger. Especially, I thank Guido Magazzù from INFN Pisa for the collaboration with the simulation framework. It was always a pleasure to meet in Pisa or at CERN.

In the context of this thesis, I would like to thank also the following persons for their review and input: Theodor Stana, Moysis Tsamsakizoglou, Djorn Karnick, Benjamin Oldenburg and Franziska Vogel.

I acknowledge the support by the DFG-funded Doctoral School "Karlsruhe School of Elementary and Astroparticle Physics: Science and Technology (KSETA)."

I would also like to thank my family and Christian Jenni who always supported me and continued to do so while I was in Karlsruhe. Last but not least, I would like to thank Salomé for giving me this great support during these intense years. I also appreciate her patience with me during the writing of this thesis.
Thank you!

Contents

List of Figures

List of Tables

List of Acronyms

2S	Strip-Strip
ALICE	A Large Ion Collider Experiment
AM	Associative Memory
AM Chip	Associative Memory Chip
ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
ATCA	Advanced Telecommunications Computing Architecture
ATLAS	A Toroidal LHC ApparatuS
BU	Builder Unit
C++ STL	C++ Standard Template Library
CAM	Content-Addressable Memory
CBC	CMS Binary Chip
CDF	Collider Detector at Fermilab
CERN	European Organization for Nuclear Research
CIC	Concentrator Integrated Circuit
CMS	Compact Muon Solenoid
CMSSW	CMS Software
CPU	Central Processing Unit
CSC	Cathode Strip Chamber
CTC	Central Tracking Chamber
DAQ	Data Acquisition
DC	Don't Care
DDR2	Double Data Rate version 2
DELPHI	Detector with Lepton, Photon and Hadron Identification
DESY	Deutsches Elektronen-Synchrotron
DO	Data Organizer
DRAM	Dynamic Random-Access Memory
DSP	Digital Signal Processing

DT	Drift Tube
DTC	Data, Trigger and Control Board
ECAL	Electromagnetic Calorimeter
EDM	Event Data Model
FCC	Future Circular Collider
FIFO	First-In, First-Out
FMC	FPGA Mezzanine Card
FPGA	Field-Programmable Gate Array
FTK	Fast TracKer
FU	Filter Unit
GBT	GigaBit Transceiver
HCAL	Hadron Calorimeter
HDL	Hardware Description Language
HE-LHC	High Energy Large Hadron Collider
HL-LHC	High-Luminosity Large Hadron Collider
HLT	High-Level Trigger
HSSL	High-Speed Serial Link
IDB	Intelligent Data Buffer
INFN	Istituto Nazionale di Fisica Nucleare
IP	Internet Protocol
IP	Intellectual Property
IPMC	Intelligent Platform Management Controller
JTAG	Joint Test Action Group
KIT	Karlsruhe Institute of Technology
L1 trigger	Level-1 trigger
L2 trigger	Level-2 trigger
LAN	Local Area Network
LEP	Large Electron-Positron Collider
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LHCf	Large Hadron Collider forward
LUT	Lookup Table

LVDS	Low-Voltage Differential Signaling
MoEDAL	Monopole and Exotics Detector at the Large Hadron Collider
MP7	Imperial Master Processor Virtex-7
MPA	Macro-Pixel ASIC
MTCA	Micro Telecommunications Computing Architecture
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PLA	Programmable Logic Array
PRM	Pattern Recognition Mezzanine
PS	Pixel-Strip
QSFP+	Quad Small Form-factor Pluggable
RAM	Random Access Memory
RF	Radio Frequency
RLDRAM	Reduced Latency DRAM
ROD	Read-Out Driver
RPC	Resistive Plate Chamber
RTL	Register-Transfer Level
RTM	Rear Transition Module
SFP	Small Form-factor Pluggable
SRAM	Static Random-Access Memory
SSA	Strip Sensor ASIC
SVT	Secondary Vertex Trigger
SVXII	Silicon VerteX detector for run II
TCP	Transmission Control Protocol
TMTT	Time Multiplexing Track Trigger
TOTEM	TOTal Elastic and diffractive cross section Measurement
TSMC	Taiwan Semiconductor Manufacturing Company
UDP	User Datagram Protocol
UML	Unified Modeling Language
VHDL	Very High Speed Integrated Circuit Hardware Description Language
XFT	eXtremely Fast Tracker

1. Introduction

In 2012, when the Higgs boson was discovered the whole world looked at the European Organization for Nuclear Research (CERN) in Geneva, Switzerland. Two of the experiments at the Large Hadron Collider (LHC)—the accelerator ring at CERN—proved the existence of the Higgs boson [1, 2]. The LHC is the largest and most powerful particle accelerator in the world and builds the heart of the research at CERN [3]. By 2025, the LHC will be upgraded, and the luminosity of the accelerated particle beams will be increased by a factor of three compared with today [4]. An increased luminosity connotes that more particles are going to collide at the center of the experiments placed at the LHC within the same time span. From this upgraded LHC, which is also called the High-Luminosity Large Hadron Collider (HL-LHC), the physicists expect new discoveries that are not possible with today's accelerators. By increasing the luminosity, the number of particles that are created in particle collisions when the beams are crossed also increase by a factor of three. Therefore, the experiments at LHC that record the results of the colliding particles also needs to be upgraded to cope with the new conditions [5, 6].

One of these experiments is the Compact Muon Solenoid (CMS) [7]. CMS is one of the two large general-purpose detectors at the LHC that were involved in the discovery of the Higgs boson. Particles collide every 25 ns at the center of the CMS experiment. New particles are produced in these collisions and are then detected by the different detectors of the CMS experiment. The amounts of data, which are produced by the high collision rates and the high resolution of the detectors, are enormous. However, interesting physics processes occur in only few collision events, and it is enough to store and analyze such events. Therefore, two trigger stages are part of the readout chain of CMS. These trigger stages process part of the data by fast algorithms and select interesting events [8]. Currently, the first trigger stage selects only 1 in 400 events, and the rest are discarded.

At the HL-LHC, the current first level trigger will not be sufficient, and the data from the silicon tracker has to be included. The silicon tracker is the innermost detector of CMS and delivers data of charged particles with high momentum

and spatial resolution [7]. To preprocess the data from the silicon tracker, a track trigger will be added to the trigger system of CMS [9]. A track trigger reconstructs the tracks of the charged particles and provides their parameters to the first level trigger. Such a track trigger at the first trigger level has never before been incorporated at a particle collider experiment, but it will be of exceptional importance for the future operation of the CMS experiment.

The requirements of the CMS track trigger are exceptionally high. New data arrive at a rate of 40 MHz, and the total input data rate is close to 100 Tbit/s. This corresponds to roughly half the bandwidth of the internet traffic in 2015 estimated by Cisco [10]. Additionally, the processed track data must arrive not later than 5 µs after the collision of the particles at the first level trigger. As it already takes 1 µs to deliver the data to the CMS track trigger, the latency of the CMS track trigger must be lower than 4 µs.

Although the CMS track trigger will be just a small piece of a big puzzle, it will be a unique heterogeneous embedded system by itself. Due to the low latency that is required, the data are mainly processed by Field-Programmable Gate Arrays (FPGAs), which are mounted on hundreds of electronic boards. One of the currently discussed concepts for the CMS track trigger even includes Application-Specific Integrated Circuits (ASICs), which are specialized in finding tracks of particles, to reduce the latency of the CMS track trigger [11].

1.1. Motivation

Due to the complexity of the system, many different groups are involved in the research on the CMS track trigger. In regular meetings, new findings are presented and discussed within the CMS track trigger community. By far the most presentations concern individual parts of the CMS track trigger, e.g. one hardware board, an algorithm for one task or its implementation within an FPGA. When large parts of the system are discussed, the discussion only takes place at an algorithmic level. About ten years before the CMS track trigger will be operational, the discussions should not only cover the separate components of the system but also include considerations on the system as a whole. For instance: how do the components influence the system properties or how do the components influence each other?

A simulation framework for the CMS detector exists—the CMS Software (CMSSW) application framework [12]. It allows physicists to simulate many aspects of CMS from the detection of the particles to the processing of the

data. However, the simulation model of CMSSW is not suited for the simulation of hardware effects because the simulation steps are fixed to the frequency at which particles collide at the experiment. Additionally, CMSSW does not natively support data types that suit hardware modeling.

The missing part in the design process of the CMS track trigger is a framework that provides a global view on the electronic system. The framework shall combine the developments of the individual components and allow the system designers to evaluate their impact to the entire CMS track trigger system. The goals of such a framework are: (1) The evaluation of figures of merit, e.g. latency of components and the whole system, communication bandwidths and buffer sizes. (2) The evaluation of different system architectures and the allocation of hardware resources to algorithms, i.e. design space exploration. (3) The possibility to test both ideas for algorithms and code developed for FPGAs within the context of the CMS track trigger system. As the input data to the CMS track trigger is non-deterministic, the framework also has to include data from simulations of the detector.

1.2. Contributions

To address the need for a framework for the evaluation of the CMS track trigger hardware, a concept for a system simulation of the CMS track trigger was developed [13]. SystemC was chosen as the modeling language of the system simulation. SystemC is well suited for such a simulation, as it facilitates to model hardware both at a high abstraction level and with enough details to determine the properties of the final system. On the one hand, the possibility to model at a high abstraction level allows a large part of the CMS track trigger to be simulated. On the other hand, the possibility to include hardware details into the simulation enables the evaluation of system properties. The question which of the two options is the best choice for a specific situation is part of the concept of the system simulation.

The CMS track trigger consists of many components that are often arranged in regular structures, e.g. the silicon detectors are read out by front-end chips arranged in a 2×8 array. A library for SystemC has been developed to ease the modeling and configuration of such structures [14].

The CMS track trigger is a very complex system, consisting of many different parts that are developed by several research groups. To create a system simulation of the CMS track trigger, detailed knowledge of every single part

is necessary. Therefore, the thesis starts with an extensive theory part that describes the CMS track trigger with the focus on a concept for the CMS track trigger that includes an Associative Memory Chip (AM Chip) specialized in track finding. This information has been used to implement a system simulation of the CMS track trigger. The simulation covers all parts of the data processing chain from the readout chips of the detector up to board running the track reconstruction algorithms. By simulating the implemented CMS track trigger with realistic input data, it has been shown that it is feasible to simulate such a large system with SystemC and getting results within reasonable time.

Some of the prototypes of the central processing boards of the associative-memory-based CMS track trigger were produced at the Karlsruhe Institute of Technology (KIT). Beside the system simulation these boards were tested, and the test procedure is described in this thesis. Furthermore, a cost estimate of the CMS track trigger has been carried out. The results of the cost estimate are extrapolated to the year 2022 when the CMS track trigger will be built.

2. High-Energy Physics and the CMS Experiment

The very fundamental questions of nature are the driving force behind high-energy physics experiments. For example: How is matter composed? What happened after the big bang? Or, why has the universe developed the way we know it today? Although physics research made big progress since the beginning of the twentieth century, many questions remain unanswered, and new questions arise constantly inspired by new findings. To address these questions, ever more powerful experiments are built typically within large international collaborations, for example, the CMS experiment at CERN.

2.1. In the beginning there was physics

In ancient times, Greek and Indian philosophers already thought that matter is composed of small basic building blocks. The name atom originates from the Greek word *atomos*, which means indivisible [15]. Research in the nineteenth century also seemed to give evidence for the theory that matter consists of atoms—discrete, indivisible particles. In 1897, when Joseph John Thomson discovered the electron, it became clear that atoms have an inner structure. In 1913, Ernest Rutherford discovered the nucleus of the atom and in this context the proton, as the nucleus of the hydrogen atom. By the discovery of the neutron by James Chadwick in 1932, an explanation was found for the faster increase of the atomic mass than its nuclear charge upward through the periodic table [16].

Over the following decades, more and more particles were discovered or theoretically proposed. Some of these particles are composite and some, at least by the current state of knowledge, elementary; i.e. they do not have an inner structure and are infinitesimally small. In the 1970s, physics theorists combined multiple theories on elementary particles in a single model—the standard model

of particle physics [15]. Figure 2.1 shows the particles of the standard model, ordered by families of similar particles.

Quarks are particles that interact through the strong nuclear force and form composite particles, which are called hadrons. The most common hadrons are the proton and the neutron, which consist of three quarks each. In contrast to the hadrons, charged leptons interact through the electromagnetic and the weak nuclear force, but not with the strong one. Whereas, the neutral leptons, the neutrinos, only interact through the weak nuclear force. Furthermore, they do not form composite particles. The electron belongs to the leptons. Additionally, an antiparticle exists for each of these matter particles. Antiparticles behave the same way as their corresponding particle, have the same mass but opposite charge.

The gauge bosons are force carriers of the fundamental forces. Particles that carry three of the four fundamental forces—the electromagnetic force, the strong force, the weak force—have been discovered. For the gravitational force, a particle called graviton has been proposed, but it has not been discovered yet. The graviton is not part of the standard model. The latest discovered particle of the standard model is the Higgs boson. It explains why the other particles, except the photon and the gluon, have mass.

However, not all observed phenomena are explainable by the standard model. For example, dark matter is a proposed type of matter that has mass but does not emit or interact with electromagnetic radiation [15]. Observations of galaxies showed that the rotational velocity in the outer areas of a galaxy does not decrease according to the distribution of the mass as predicted by the laws of Newton. Instead, the rotational velocity remains constant [15]. Therefore, a hypothetical halo of dark matter around galaxies has been suggested. Combining all observations, only 4 percent of the mass/energy in the universe consists of visible matter, and 22 percent is dark matter [15]. The remaining 74 percent are supposed to be contributed by the also undiscovered dark energy, which is responsible for the acceleration of the universe [18].

To investigate this kind of new physics and also to refine the parameters of the standard model, ever more sophisticated experiments are designed. Particle colliders are one type of experiments for this purpose. In collider experiments, two beams of particles are accelerated to speeds very close to the speed of light. At dedicated interaction points the two beams are crossed which causes particles in the beam to collide. Through the collision, high amounts of energy

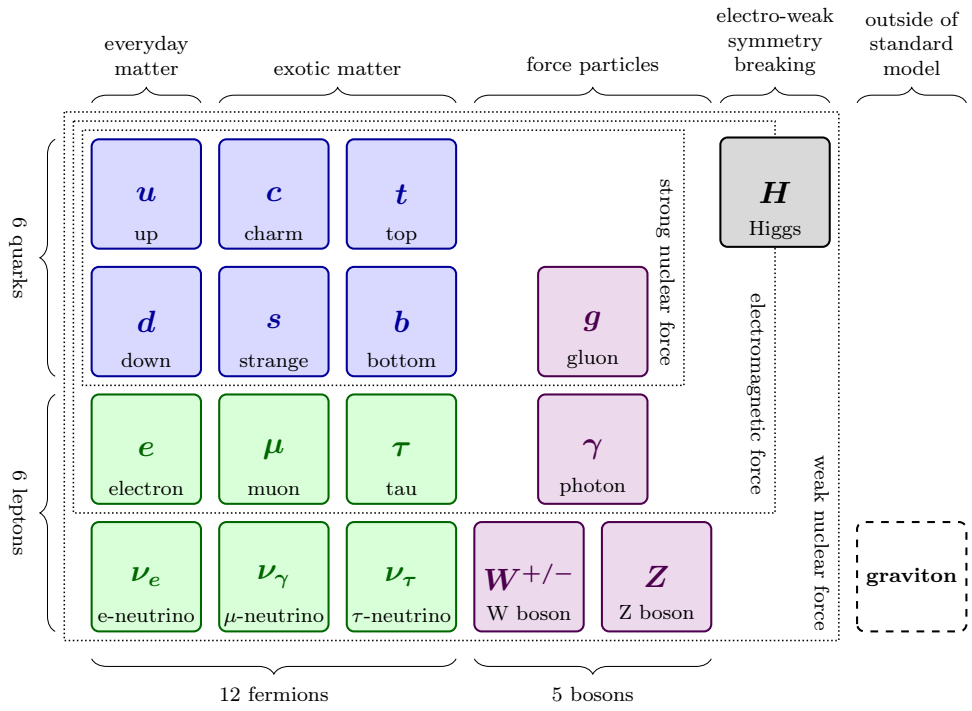


Figure 2.1.: Schematic illustration of the standard model of particle physics. Source: [17]

are released, and new particles are created with a certain probability. According to Einstein's famous equation energy (E) and mass (m) are related by

$$E = mc^2 \tag{2.1}$$

where c is the speed of light [15]. Thus, the higher the energies of the colliding particles are, the heavier the newly created particles may be. For instance, the Higgs boson with a mass of $125\,\mathrm{GeV/c^2}$ needs the same amount of energy to be produced. Therefore, new colliders that provide higher collision energies are designed to discover new particles. Particles generated in the collisions or their decay products can be observed by detectors arranged around the collision point. Such a collision point is also called interaction point. Usually, many different detector systems are necessary that are specialized in the detection of specific particles or physical quantities.

Sophisticated algorithms based on theoretical physics are applied to the detector data to reconstruct the processes and involved particles. By means of physics

conservation laws, it is possible to draw conclusions from the recorded situation to the processes that happened just after the collision. The conservation of energy and momentum should be mentioned here as an example. The transverse momentum (p_T) is the momentum of the particle perpendicular to the beam. As it is zero before a collision, it must also be zero afterward. If the transverse momentum of all detected particles is summed up and some transverse momentum is missing, there must have been particles that were not detected. These could be neutrinos, which are not easily detectable because they interact only weakly, or an indication of previously unknown particles—probably dark matter?

2.2. The Large Hadron Collider (LHC)—a success story

After finishing the experiments in 2000, the predecessor of the LHC, the Large Electron-Positron Collider (LEP) [19] at CERN had to be replaced in order to continue cutting-edge research on new physics. In 1994, the CERN Council decided to build a new accelerator with new experiments. The result was the LHC [3], whose construction started in 1998 and went operational in 2008. After just four years of operation, the scientists of the two largest experiments at the LHC announced the discovery of the Higgs boson in [1, 2]. It is still the most important finding of the LHC so far and the culmination of more than five decades of experimental search for the Higgs boson. The discovery smoothed the way for the award of the Nobel Prize in Physics to François Englert and Peter W. Higgs in 2013 [20]. They received the prize for the theoretical prediction of the Higgs particle in [21, 22].

The LHC is located in a tunnel with a circumference of about 27 km on the border between Switzerland and France at a depth of 100 m on average. Thereby, the tunnel that was originally dug for LEP between 1984–1989 is reused. The LHC accelerates hadrons in two rings with counter-rotating beams. These beams are crossed at four interaction points. As the accelerated hadrons are composite particles, not the hadrons themselves collide but the quarks and gluons, which they are built from. Different experiments are located at these interaction points whose positions are sketched in Figure 2.2.

Most of the time the particles collided at the LHC are protons. For some fraction of the time, the LHC is also used to collide lead ions. The particles for the LHC beams are produced and pre-accelerated by the CERN accelerator

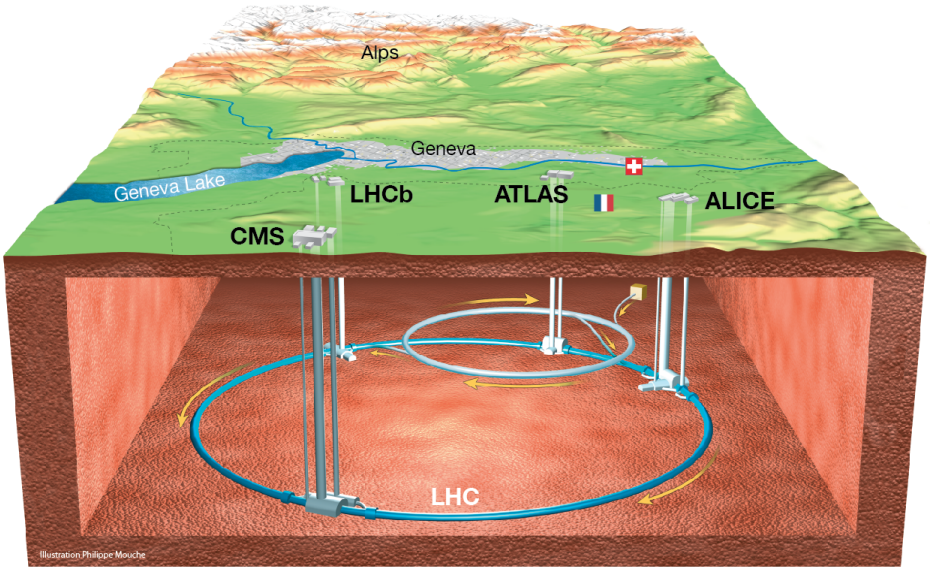


Figure 2.2.: LHC with the interaction points where the four largest experiments are located. The SPS ring, which is illustrated in the back, is now used as a pre-accelerator. Courtesy of CERN.

facility [23]. This is a system of several accelerators and storage rings that accelerate the particles step-by-step until they are fed into the LHC rings with an energy of 450 GeV. At the start of the LHC program, protons were then further accelerated by the LHC up to a collision energy of 8 TeV. The collisions energy has been increased step-by-step and is at 13 TeV since 2015. Ultimately, the collision energy will reach 14 TeV after the Phase-I upgrade, as described in Section 2.2.2. At top energy, the protons move at 99.999 999 1 percent the speed of light.

Protons are so small that it is impracticable to align the paths of two single protons exact enough so that they collide. Therefore, not single protons but many protons in so-called bunches are collided by the LHC. The bunches are spaced in the ring by 25 ns, which correspond to a distance of 7.5 m. In total, 2808 bunches are concurrently rotating in each ring. One bunch contains up to 115 billion protons right after filling the ring with protons. At the interaction points, the beam is focused to a transverse size of only 16.7 µm [23]. Today, in average only 20 of the protons in a bunch collide each bunch crossing.

Through the spacing of the bunches by 25 ns, the collision rate of the bunches is 40 MHz.

Within the tunnels, the particle bunches run in a vacuum pipe with a diameter of 56 mm. 1232 superconducting dipole bending magnets are installed in the tunnel to keep the particles on the circular track. Each of them is cooled down to 1.9 K and produces a magnetic field of 8.33 T. Different magnet systems are arranged around the ring to keep the beam focused, and 16 superconducting Radio Frequency (RF) cavities accelerate the particles.

2.2.1. Experiments at the LHC

Seven experiments are located at the four interaction points of the LHC. The two largest ones are CMS [7], described closer in Section 2.3, and "A Toroidal LHC ApparatuS" (ATLAS) [24]. Both are general purpose particle detectors, and their data are used to research many different questions of fundamental physics. They follow the same goals by applying different technologies and thereby complement each other. The other experiments pursue more specific research programs.

The "A Large Ion Collider Experiment" (ALICE) experiment [25] employs lead ion collisions to investigate quark-gluon plasma. Quark-gluon plasma is a state of matter where the quarks and gluons are free, i.e. they are not bound together and do not build larger particles, such as protons or neutrons. These are the conditions that existed a few microseconds after the big bang. During proton-proton runs, ALICE contributes to the results of CMS and ATLAS.

The main research program of the "Large Hadron Collider beauty" (LHCb) experiment [26] targets the matter-antimatter asymmetry in the universe. Although the big bang should have produced the same amount of matter and antimatter, there is more matter than antimatter in the observable universe today. LHCb tries to find reasons for this. Additionally, LHCb supports CMS and ATLAS in the search for unknown particles.

Located at the same interaction point as CMS, the "TOTal Elastic and diffractive cross section Measurement" (TOTEM) experiment [27] performs measurements of scattering effects in proton-proton collisions. This allows the physicists to measure the size of the proton with very high precision.

The "Large Hadron Collider forward" (LHCf) experiment [28] shares the interaction point with the ATLAS experiment. It measures the numbers and energy

of neutral pions, particles consisting of a quark and an antiquark. The measurements may explain the origin of ultra-high-energy particles from cosmic rays discovered by the Pierre Auger Observatory [29]. At the observatory, particles with energies above 10^{19} eV were discovered—energies million times higher than the ones of the particles at the LHC.

The "Monopole and Exotics Detector at the Large Hadron Collider" (MoEDAL) [30] is the smallest experiment at the LHC and located in the same cavern as the LHCb experiment. Its purpose is to search for theoretically proposed but as yet undiscovered exotic particles. For instance, MoEDAL searches for magnetic monopoles, i.e. particles with magnetic charge. The experiment also searches for dark matter and hints of extra dimensions.

2.2.2. Upgrades at the LHC

With the advance of technology, the LHC and the experiments may constantly be upgraded. The increased performance facilitates the search for physics processes, which were inaccessible before. There are two parameters that can be tuned: the collision energy and the luminosity.

The increase of the collision energy would give access to physics processes that do not happen at lower energies. However, the maximum energy that can be achieved in a circular storage ring depends on the radius of the ring and the magnetic field of the bending magnets. Obviously, the tunnel of the LHC is fixed. Also, the technology for stronger magnets is not mature yet. Therefore, the collision energy at the LHC will remain at 14 TeV in the near future.

The luminosity is a measure of the number of collisions over a particular time in a particular area and is measured in $\mathrm{cm}^{-2}\mathrm{s}^{-1}$. It may be defined by

$$L = \gamma \frac{n_b N^2 f_{rev}}{\sigma} \tag{2.2}$$

where γ is the energy of the particle beam, n_b the number of particles in a bunch, N the number of bunches in the ring, f_{rev} the revolution frequency of the bunches and σ a parameter describing the cross section of the beam [4]. As already discussed, the beam energy cannot be increased further. Likewise, the revolution frequency is predetermined by the length of the tunnel. An increase in the number of bunches would increase the collision rate, which would

need a replacement of many parts of the experiments. Therefore, the number of particles in a bunch and the cross section of the beam remain as the tunable parameters at the LHC. In contrast to an increase in energy, with an increased luminosity no new processes will occur that did not occur before. However, processes with low probability happen more often at higher luminosities. Therefore, more statistical evidence can be collected in the same amount of time.

The measure to specify the productivity of an experiment is the luminosity integrated over time, i.e. the number of collisions that occurred at an experiment.

$$L_{int} = \int L \, \mathrm{d}t \tag{2.3}$$

The unit of the integrated luminosity is measured in inverse femtobarns (fb^{-1}), whereas 1 fb corresponds an area of $10^{-43}\,\mathrm{m}^2$. An exact definition of inverse femtobarn goes beyond the scope of this thesis. Some numbers should help to establish an understanding of the scale of the unit. Before the LHC, all hadron colliders together accumulated a luminosity of about $10\,\mathrm{fb}^{-1}$. This value has already been exceeded by the LHC within the first three years of operation (2010–2012) [31] The goal is to reach $300\,\mathrm{fb}^{-1}$ by the end of 2022. Figure 2.3 shows the progress of integrated luminosity at the LHC and HL-LHC. In the HL-LHC phase after 2025, each year an additional $300\,\mathrm{fb}^{-1}$ will be collected, and a total of $3000\,\mathrm{fb}^{-1}$ is targeted by the end of 2036.

For the upgrade of the detectors, the accelerator has to be shut down, and the experiments have to be opened which takes several weeks. Therefore, the upgrade and maintenance shutdowns are synchronized among the experiments. Figure 2.3 shows the plan for the long shutdowns scheduled for upgrades between 2010 and 2035. The figure also shows the trend of the peak luminosity and the integrated luminosity.

2.2.3. The High-Luminosity LHC and beyond

Around 2025 after the long shutdown 3 with the Phase-II upgrade, the luminosity will be increased by a factor of five in relation to the initial luminosity when the LHC went operational. This phase is called High-Luminosity Large Hadron

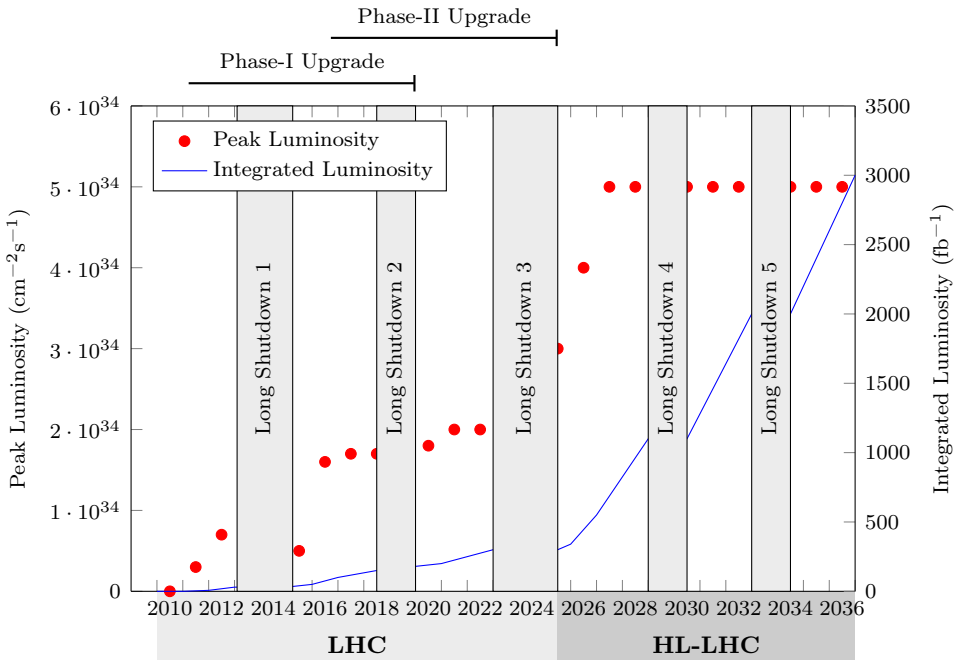


Figure 2.3.: Schedule for the long shutdowns at the LHC. Source: [4, 31, 32]

Collider (HL-LHC) [4]. The HL-LHC has been approved by the CERN Council in 2013 and facilitates to continue research beyond the limits of the original LHC.

To achieve the higher luminosity, the particle beam has to become more intense (more particles in a bunch or more bunches in the accelerator ring) and more focused (smaller beam area). The beam intensity is increased by upgrading the injection system to the LHC. To improve the focusing of the beam, several magnet systems in the tunnel have to be exchanged. Especially, the focusing magnets in front of the CMS and ATLAS experiments that focus the beam at the interaction point. Furthermore, upgrades at all large experiments will be necessary to cope with the new conditions. The upgrades needed at CMS are described in Section 2.3.5.

Whereas the luminosity will be increased at the HL-LHC, the rate at which the proton bunch collides will stay the same. Consequently, the number of simultaneous proton collisions in one bunch crossing will increase. The number of simultaneous collisions is also called pile-up. Figure 2.4 shows two example collisions to visualize the pile-up under LHC and HL-LHC conditions. In 2012,

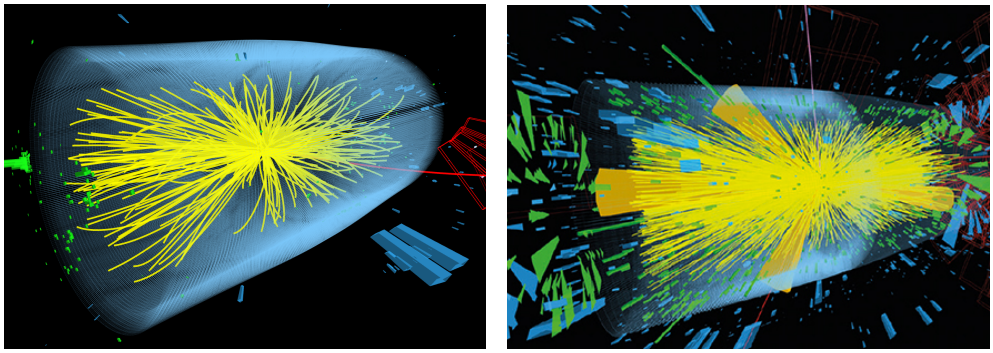
the average pile-up was 20, and the maximum pile-up was 40. At the HL-LHC, the pile-up will increase to 140 in average and may be as large as 200. The number of produced secondary particles will increase accordingly.

The experiments were designed with conditions at the original LHC in mind. Under the HL-LHC conditions, there are two challenges they have to face. Firstly, the high number of particles produced in collisions requires detectors with high enough spatial resolution to distinguish between the separate particles. Secondly, the radiation damage to the detectors increases proportionally to the increase in integrated luminosity. The detectors must be carefully designed with respect to that.

Which findings do the physicists expect from the HL-LHC? With each new high physics experiment, the properties of the known particles and underlying processes are determined more precisely. Thereby, the existing models can be refined. The research on the Higgs boson will continue. Besides the precise measurement of its properties, also decays and production processes of the Higgs boson will be investigated. The search for undiscovered particles will continue, and if they exist they are expected to have a mass of $3\,\mathrm{TeV/c^2}$ or more; for comparison, the mass of the Higgs boson is $125\,\mathrm{GeV/c^2}$. With the higher luminosity, the production of such particles occurs more often and, thus, they could be detected. Another candidate for a discovery at the HL-LHC are supersymmetric particles which might explain dark matter.

Though the increased luminosity supports the search for new physics, an increase of energy of the individual particles in a bunch would provide better conditions. Therefore, discussions beyond the HL-LHC have started [33]. Two main issues arise: firstly, the availability of stronger dipole magnets and secondly, the cost if a new, even longer tunnel needs to be dug.

Two projects should be mentioned here: the High Energy Large Hadron Collider (HE-LHC), which is a plan for an upgrade of the LHC in the same tunnel, and the Future Circular Collider (FCC) in a new longer tunnel. The goal of the HE-LHC is to reach a center of mass energy of 33 TeV—more than twice the actual value. To reach this energy scale, superconducting dipole magnets with a field of 20 T would be necessary. The ideas of the FCC go further; a new tunnel with a circumference of up to 100 km would be dug at CERN. Together with 20 T magnets, a center of mass energy of 100 TeV would be reached.



(a) (b)

Figure 2.4.: Subfigure (a) shows the particles in a recorded event from May 2016 at a center-of-mass energy of 13 TeV. The situation at the HL-LHC is shown in Subfigure (b) that shows a simulation of a collision in 2025. Each yellow track starting at the origin corresponds to one charged particle. Courtesy of CERN.

2.3. The CMS experiment

As already mentioned, the CMS experiment [7] is one of the two large, general-purpose experiments at the LHC. The whole detector is 21.6 m long, has a diameter of 15 m and weighs about 14 000 metric tons. A large international collaboration consisting of more than 3500 scientists and engineers from 43 countries is involved in the experiment [34].

If not mentioned explicitly, the information in this section refers to the initial CMS experiment before the upgrades, as built in 2008.

2.3.1. Detectors of the CMS experiment

A single type of detector that can locate and measure all kinds of particles does not exist. Therefore, the typical particle collider experiment consists of several different detectors for different particles and physical quantities. In the case of the CMS experiment, the detectors are arranged onion-like around the collision point of the particle beams [7]. Figure 2.5 shows a cutaway of the experiment, so the location of the different sub-detectors is visible.

All detectors are split into two parts. Firstly, the barrel section where the sensors are arranged cylindrically around the interaction point. Secondly, the

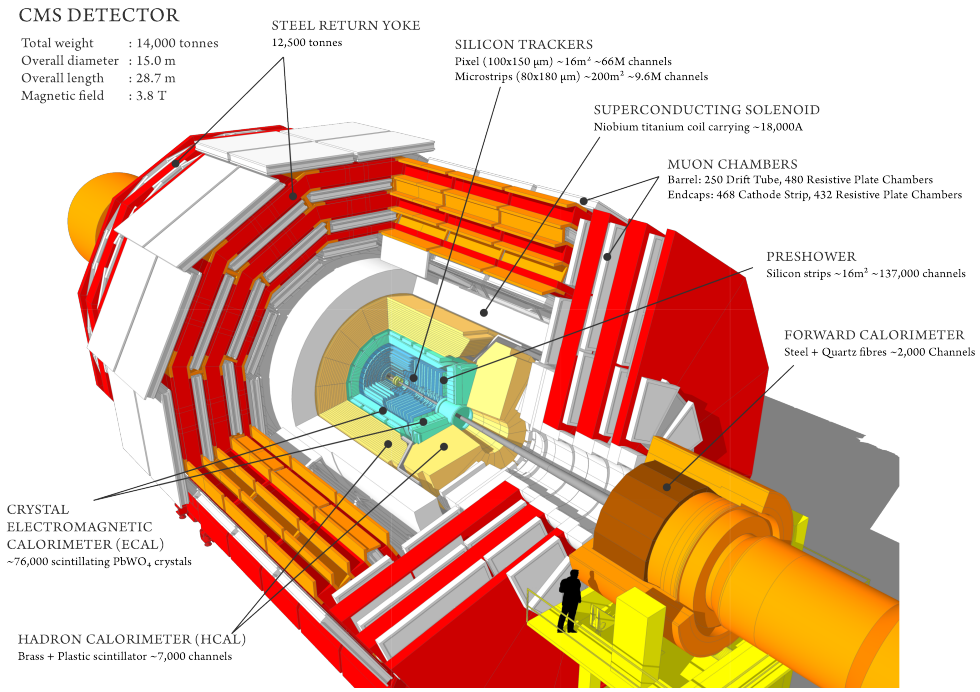


Figure 2.5.: Overview of the different sub-detectors of the CMS experiment. Courtesy of CERN.

endcaps consisting of discs of sensors that close the barrels in the front and the back. Because much ionizing radiation is created by the colliding beams, radiation-hardness is a common issue that must be considered during the development of the detectors. This is especially important for the detectors closest to the interaction point.

A crucial part of the detector is the 4 T solenoid magnet [35]. With an inner diameter of 6 m, it is the biggest superconducting magnet ever built. This large inner diameter has the advantage that both calorimeters can be placed within the magnet. The result is a higher energy resolution of these sub-detectors because the particles do not have to pass the magnet material. The purpose of the magnet is to bend the paths of charged particles, such as electrons and muons. Thereby the transverse momentum and indirectly the energy of charged particles can be determined.

The innermost detector of the CMS experiment is the silicon tracker. It consists of thin sheets of silicon that are able to detect the passage of charged particles.

More on the functionality of silicon trackers can be found in Section 3.1. Multiple layers of these silicon detectors are used to track the path of a charged particle. The CMS silicon tracker is split into an inner and an outer tracker. The inner tracker consists of pixel modules that allow a fine spatial resolution. The closest modules are mounted just about 4.4 cm from the collision point of the particles. In total 1440 modules with 66 million pixels form the inner tracker. The outer tracker is built of strip silicon sensors to save money and reduce the power dissipation within the detector. This optimization is possible because the resolution requirements are lower in modules further away from the interaction point. The outer tracker is built of more than 15 000 modules on 10 layers with an active area of about 198 m^2. As little material as possible is used to build the silicon tracker so that multiple scattering is minimized for optimum momentum measurement.

Then the particles reach the next detector—the Electromagnetic Calorimeter (ECAL) [36]. Calorimeters are used to measure the energy of particles; the ECAL in particular measures the energy of photons and electrons. The CMS ECAL consists of about 76 000 lead tungstate ($PbWO_4$) crystals. When a high-energetic photon or electron interacts with a crystal, a shower of low-energetic photons is generated [37]. A shower means that out of a high-energetic particle many particles with lower energy are generated. These low-energetic photons are eventually absorbed by the crystal, and they cause the crystal to scintillate, i.e. to reproduce a flash of light. The energy produced in this light flash is proportional to the energy of the original particle. Avalanche photodiodes (barrel section) and vacuum photo triodes (endcaps) glued to the crystals detect the light pulses. The CMS ECAL has been designed to have a high energy resolution, to be fast and to be radiation resistant.

The last detector inside of the superconducting solenoid is the Hadron Calorimeter (HCAL) [38]. It measures the energy of hadrons, e.g. protons, neutrons, pions, kaons. Also, the HCAL is very important to indirectly detect neutrinos or unknown particles that are not interacting with any of the detectors. Missing energy provides evidence of these types of particles. Therefore, precise measurements from the HCAL are essential.

The HCAL consists of alternating layers of brass or steel absorbers with plastic scintillators. Hadrons interact with the matter of the brass and steel absorbers, and secondary particles are created. The secondary particles may interact again, and so a multitude of particles is created in a so-called particle shower. These showers cause the plastic scintillators to emit light, which is read out by optical

fibers and detected by photodiodes. Seventeen of these metal absorber, scintillator combinations are stacked into towers. The whole HCAL consists of 4300 of such towers.

As muons do not interact much with matter, the muon system [39] can be located outside of the superconducting solenoid. The muon system is complex and consists of 1400 muon chambers of three different types. The main detection system consists of Drift Tube (DT) wire chambers in the barrel section and Cathode Strip Chambers (CSCs) in the endcaps. Both have a similar working principle but differ in their properties. A wire chamber is a box filled with gas in which an array of wires is placed [37]. The wires are biased by a high voltage to produce an electric field in the chamber. If a particle passes through the detector, the particle ionizes the gas. The charge of the produced electrons and ions is then collected by the wires and readout by electronics.

The wire chambers are arranged in layers interleaved with the steel return yoke of the solenoid. This allows to reconstruct the tracks of the muons and measure their transverse momentum.

The third detector type of the muon system is the Resistive Plate Chambers (RPCs) that are built of two conducting plates with gas between them. They have a coarser position resolution but are faster than the other two detector types. The RPC have been added in the same areas as the wire chambers and serve as an independent system for the trigger.

2.3.2. Level-1 trigger of CMS

When particles collide, new particles are produced and decay then into other, secondary particles. These processes happen with a certain probability. Unfortunately, collisions with interesting physics processes are rare. For instance, only about one in a billion events contains an interesting Higgs boson process. Therefore, as many collisions as possible need to be recorded to obtain statistically significant data. This is achieved by high luminosities, see also Section 2.2.3, which means to push the collision rate to the maximum possible, limited by the detector technology.

High crossing rates result in extreme data rates. For instance, the data produced by all detectors of CMS in one collision event have a size of about 1.5 MB. Multiplied by the LHC bunch crossing rate of 40 MHz, a data rate of 480 Tbit/s would be the result. It is simply not feasible to readout these amounts of data, process them by sophisticated algorithms and store them away.

For this reason, most of the high-energy physics experiments include triggers that select the events with interesting processes at an early stage [37, 40]. This selection is possible because interesting physics processes occur rarely and only events containing those need to be stored. Triggers search in real-time for certain signatures within an event, which are indicative for interesting physics. The selected events are transferred to the next processing stage, and the other events are discarded. As the data of uninteresting events are lost forever, effective trigger algorithms are crucial for the quality of the final results. To define the trigger signatures, intense studies were made in theoretical physics and by simulations.

Modern trigger systems usually consist of several trigger levels. At each level, the event data are stored in a buffer parallel to the processing. If the trigger decides to keep the event, all data that belongs to this event are passed to the next level. The idea behind multi-level triggers is that the lower level triggers run simple algorithms with low latency to keep the data buffers small. The higher-level triggers run more sophisticated algorithms, as the event rate has been reduced significantly by the lower level triggers.

The CMS trigger system [8] as shown schematically in Figure 2.6 consists of two levels: the Level-1 trigger (L1 trigger) and the High-Level Trigger (HLT). The L1 trigger processes data with coarse resolution from the calorimeters and the muon chambers. Data from the silicon tracker are not used by the initial L1 trigger but will be included after the Phase-II upgrade (see Section 2.3.5). The data needed by the trigger are extracted from the raw data stream and sent to the L1 trigger. The electronics systems of the L1 trigger are located in the counting room—a room close to the detector but shielded from radiation. The raw data itself are kept on the detector in pipelines within the front-end electronics. As the pipelines can store data for 3.2 µs or 128 bunch crossings respectively, the latency of the L1 trigger must not exceed this limit.

Few functions of the L1 trigger are executed by electronics on the detector, but most of the L1 trigger is located in the underground control room. As the control room is located 90 m away from the detectors, the signal propagation time reduces the time available for the trigger further. To achieve the low latency, the algorithms of the L1 trigger run on dedicated hardware, mainly FPGAs. The trigger decision is sent back to the front-end pipelines, where it triggers the readout of the raw data of the correspondent event. The L1 trigger reduces the event rate from the 40 MHz at the detector down to about 100 kHz.

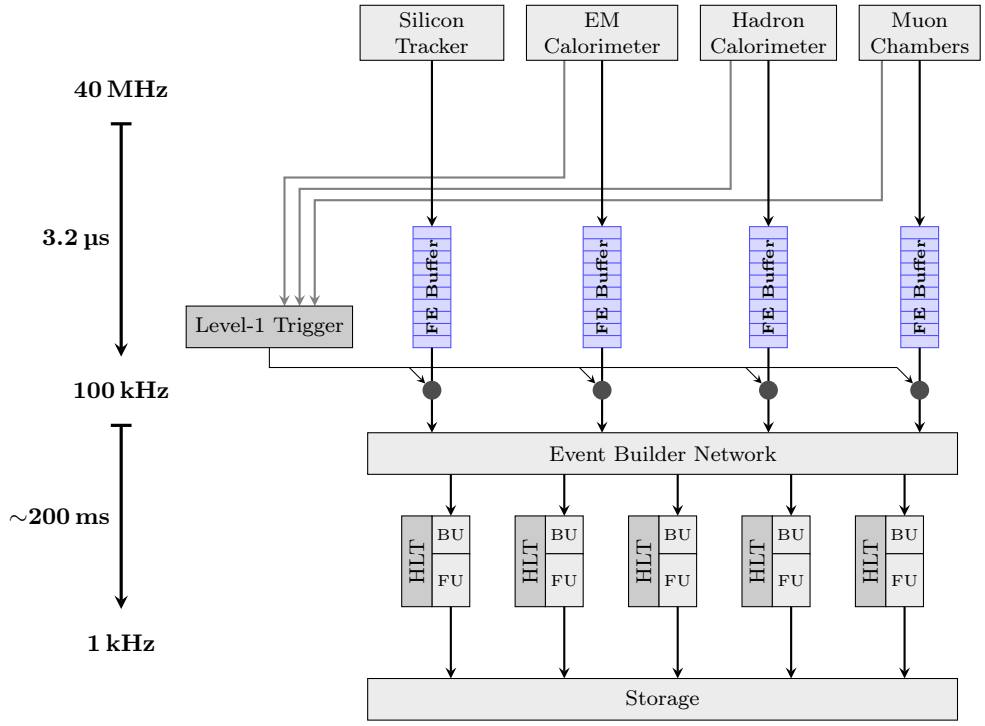


Figure 2.6.: Overview of the initial trigger and Data Acquisition (DAQ) system of CMS before any upgrades, as built in 2008.

2.3.3. High-level trigger and DAQ system of CMS

At the CMS experiment, the HLT is tightly coupled with the DAQ system [41–43]. After receiving the trigger signal, the data of the event stored in the different front-end pipelines is fed to the Event Builder Network. The event builder network is a complex switched computer network with many nodes that collects the data from the different front-end modules, organizes the data into events and assigns them to a computational node of the HLT. During these steps, the DAQ data are also transferred from the underground control room to the computing facility on the ground.

From the Event Builder Network, the data of one event is assigned to one computer in a computer farm. Each computer runs a Builder Unit (BU) and a Filter Unit (FU). The BU reformats the data to create the final event type and buffers them until the FU is ready to process it. The algorithms of the HLT

are executed by the FU. As the HLT algorithms are implemented as software, the HLT is very flexible, and new trigger algorithms can be included as needed. The average time to process an event at the HLT is 200 ms.

In 2012, the event rate was reduced by the HLT to a maximum of 1 kHz. Thus, around 12 Gbit/s were stored in the storage facilities of CERN. From the storage facility, physicists access the events for off-line analyses.

2.3.4. Phase-I upgrade at CMS

During the long shutdown 2, the CMS experiment will be upgraded for the operation under Phase-I conditions of the LHC [32]. From 2020 on, the LHC will provide twice the nominal luminosity. However, the necessary changes at the CMS experiment are moderate.

The pixel layers of the silicon tracker have to be exchanged completely. On the one hand, the current readout chips cannot master the higher number of particles. On the other hand, the radiation-hardness is not sufficient for longer operation under Phase-I conditions. Additionally, a fourth layer in the barrel section and a third endcap disk will be added. They serve as a kind of backup in the case that the innermost detectors will fail due to high radiation. Half of the new pixel detector will be produced at KIT and the other half at the Deutsches Elektronen-Synchrotron (DESY) in Hamburg [44].

Other upgrades will be undertaken in the muon system, the hadron calorimeter and the DAQ system. A new layer will be added to the muon system, and some electronics will be upgraded to the newest technology to improve performance. Similarly, the electronics of the hadron calorimeter will be upgraded. The bandwidth of the DAQ system has to be increased by a factor of 2 to 5. Likewise, the computational power of the HLT computer farm needs to be increased to cope with the new conditions.

2.3.5. Phase-II upgrade at CMS

The even higher luminosity at the HL-LHC requires major changes at the CMS detectors during the Phase-II upgrade [5].

Once more, the radiation takes its toll and the complete silicon tracker—pixel and strip sections—has to be exchanged. The new silicon detector design will be extremely radiation-tolerant to ensure its functionality for several years

under the HL-LHC conditions. To separate the tracks of individual particles in hundreds of overlapping pile-up events, the pixel and strip granularity will be reduced by a factor of four. Additionally, an entirely new concept is introduced. The detector modules comprise two closely placed silicon sensors which help to provide an estimate of the transverse momentum of a particle already at the module level. Chapter 3 describes the CMS silicon tracker after the Phase-II upgrade in detail.

Also the calorimeter endcaps, both ECAL and HCAL, have to be replaced completely due to radiation damage. With the replacement, calorimeters with a significantly better resolution are installed. Thereby, detailed 3D reconstructions of electromagnetic and hadronic showers will be available.

The CMS trigger system will undergo major upgrades. The most significant change, and also the setting of this thesis, is the integration of the silicon tracker into the trigger. A track trigger will be implemented that preprocesses the tracker data for the L1 trigger. As shown in Figure 2.7, the CMS track trigger receives its data directly from the silicon tracker and provides information about particle tracks to the L1 trigger. A detailed description of the CMS track trigger follows in Chapter 4. To provide enough calculation time to the trigger functions, the latency of the L1 trigger is increased from 3.2 µs to 12.8 µs, which correspond to 512 bunch crossings. As the front-end electronics on the detector buffer the raw data, the electronics need to be upgraded to store more events. Also, the L1 trigger acceptance rate needs to be increased to at least 750 kHz.

Due to the higher trigger rate and the larger event size, also the bandwidth of the DAQ system will be upgraded. This change draws through the whole data processing chain. For example, the HLT selects in average every hundredth event. Therefore, its acceptance rate will rise to 7.5 kHz under Phase-II conditions [5].

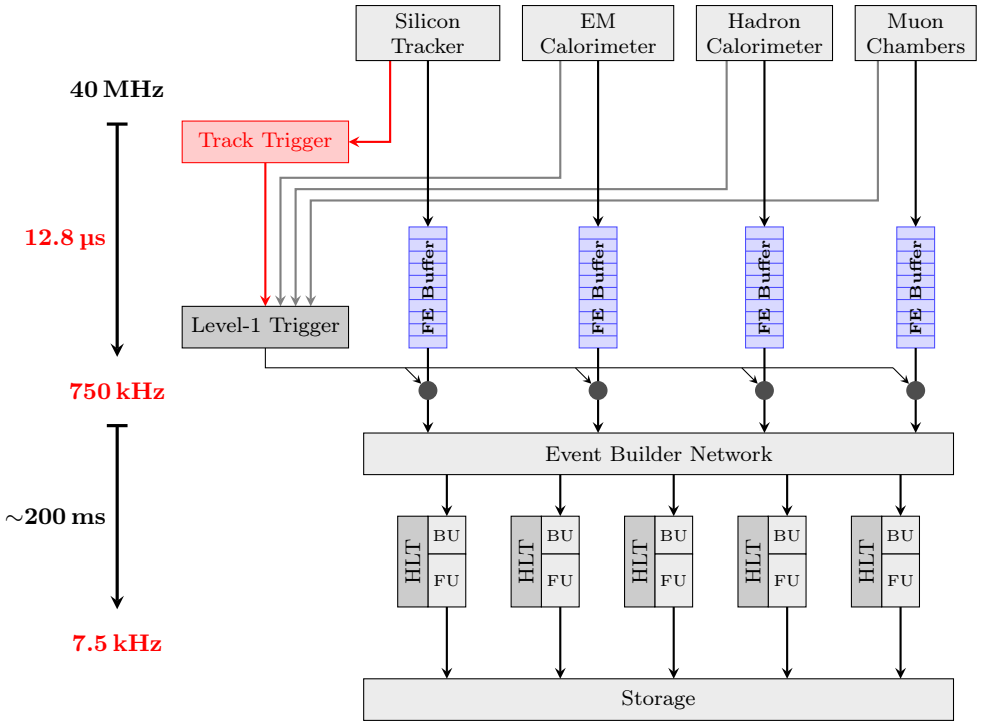


Figure 2.7.: The HL-LHC CMS trigger system with the added track trigger. Changes from the initial trigger systems are marked in red.

3. The CMS Silicon Tracker

Since the early 1980s, silicon trackers are a valuable tool to record high-resolution 3D images of particle collisions in high-energy physics experiments [37]. Silicon trackers record these images at very high speeds, for example with a repetition rate of 40 MHz at CMS. By the analysis of these images, the path and the momentum of charged particles can be determined.

A silicon tracker is built of silicon detectors that are arranged around the collision point, as shown in Figure 3.1, to detect charged particles that were created in the collision. Charged particles—such as electrons, muons or charged hadrons—generate a signal at the location where they pass through the silicon detector. Such a detected location of a particle is called a *hit*. Because the silicon tracker is relatively light-weight and does not affect the particles much, it is usually the innermost detector of an experiment.

One important feature of silicon trackers is the measurement of the transverse momentum of charged particles, which can be determined by the curvature of the track of the particle. At CMS, the Lorentz force caused by the magnetic field of the superconducting solenoid (3.8 T) bends charged particles into a curved track. Whether the particle turns clockwise or counterclockwise indicates the polarity of the charge of the particle. The radius of the curvature is a measure for the transverse momentum of the particle. Particles with high transverse momentum stay on an almost straight path. In contrast, particles with very low transverse momentum start to spiral in the detector. An illustration of particles with different transverse momenta is shown in Figure 3.2.

The functionality of silicon trackers in general and the composition of the CMS silicon tracker after the Phase-II upgrade are described in this chapter. As the innermost detector, the silicon tracker is located within the electromagnetic calorimeter. The silicon tracker is approximately 2.7 m long and has a diameter of about 2.2 m. The CMS silicon tracker for the HL-LHC will be the first tracker that adopts a concept for the determination of the transverse momentum at the module level [46]. Hits from high transverse momentum particles are

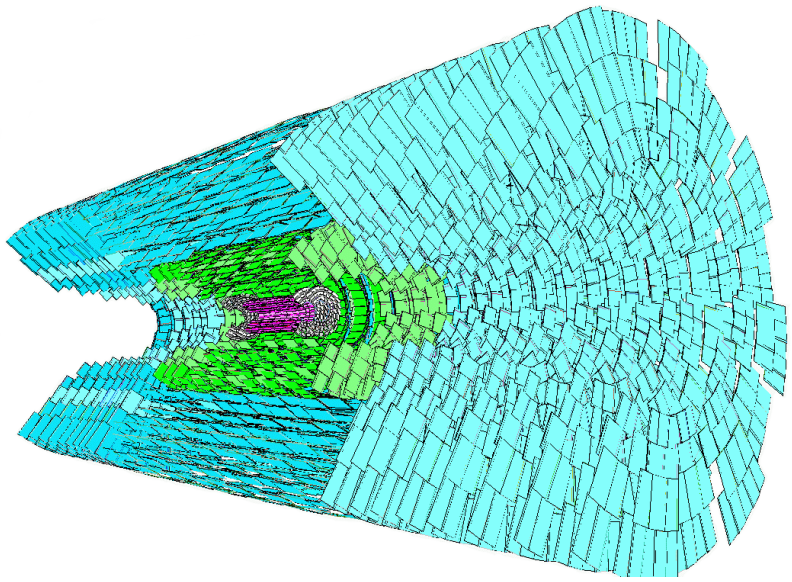


Figure 3.1.: 3D Rendering of the initial CMS tracker. The pink modules are pixel modules, the green and blue modules are strip modules. Courtesy of Giuseppe Zito [45].

recognized and only the information about them is transmitted off-detector and then processed by the CMS track trigger.

3.1. Particle detection by silicon detectors

While passing through matter, charged particles ionize the matter along their path. Detectors called ionization chambers make use of this effect to track charged particles. In the early days of high-energy physics experiments, bubble and wire chambers were used [37, 47, 48]. Both of them consist of large tanks filled with a liquid or a gas in which tracks of particles can be detected. Unfortunately, these detector types are slow. Therefore, silicon detectors were developed in the early 1970s and installed at high-energy physics experiments from the 1980s on. Although silicon detectors are expensive in production and put dense matter into the path of the particles, they have several advantages [49]:

- Silicon is widely used for electronics devices, and the production processes are well understood and easily available.

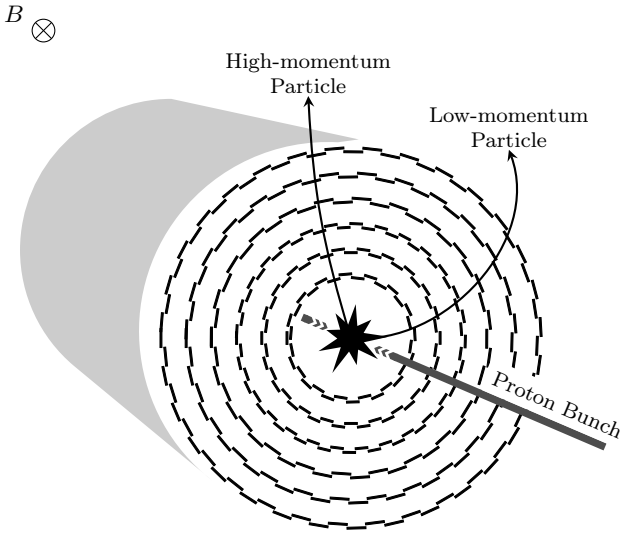


Figure 3.2.: Particles with different transverse momenta in the CMS silicon tracker.

- Small elements (pixels) can be produced on a silicon wafer. Thus, detectors with fine resolution are possible.
- The small size of the detector and the high speed of the charge carriers in silicon lead to very fast detectors.
- As the detectors are produced by the same process as electronics, electronics may be directly integrated within the sensor.

Minimum ionizing particles that pass through such a silicon detector, as shown in Figure 3.3, create about 80 electron-hole pairs per micrometer path length in silicon [49]. Typically, the thickness of a silicon detector is of the order of a few hundred micrometers. For example, in the inner part of the original CMS silicon tracker, sensors with a thickness of 320 µm are installed. Therefore, a particle passing straight through the sensor generates approximately 25 000 electron-hole pairs. Electrodes are placed on both sides of the sensor. A bias voltage is applied between these electrodes so that the electrical field collects the freed electrons and holes. The resulting current is then amplified, digitized and can be processed. By segmenting at least one of the electrodes into strips or pixels, sensors with position sensing capabilities can be built. The readout of pixelated sensors is complicated and requires a lot of power. However, it is usually sufficient to have a high resolution in only one direction. Therefore,

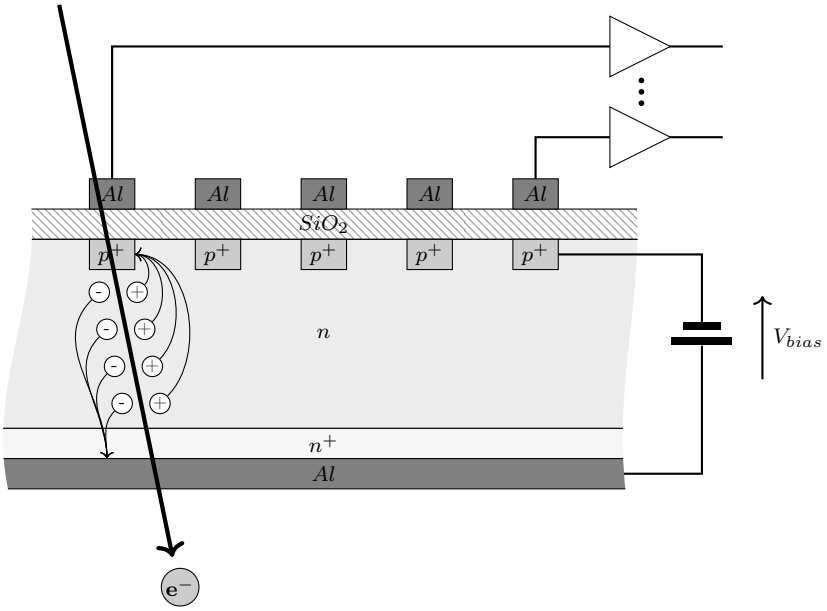


Figure 3.3.: Cross section of a silicon detector. A high-energetic electron passing through it creates electron-hole pairs. The electrons and holes are then collected by the electrodes on both sides of the sensor. Source: [49].

pixel sensors are often used only in the inner parts of the detector and strip sensors are used in the outer parts. This also reduces the cost of the detector.

As the number of charge carriers depends on the sensor thickness, a thicker sensor produces a larger signal. Unfortunately, thicker sensors also have some clear disadvantages. More material affects the trajectory of the particles more and, thereby, the accuracy of the outer detectors is degraded. In addition, a higher voltage needs to be applied to achieve the same electric field in the sensor. Therefore, the sensors are kept as thin as possible so that the resulting signal is just large enough to be measured accurately.

The problem with intrinsic silicon is that the currents needed to achieve fast readout of the sensors are impractically high. A numerical example from [49] should exemplify that. The typical signal current in a silicon detector is of the order of microamperes— 1 µA are assumed here. Silicon can be grown with a resistivity of maximum 10 kΩ cm which leads to a resistance of 300 Ω for a 300 µm thick and 1 cm^2 large sensor. Radiation damage and thermal excitation of electron-hole pairs decrease the resistivity even more. The current caused by the voltage that creates the electric field for the charge collection should be

small compared to the signal current. So, if the quiescent current has to be below 0.1 µA, the voltage applied must not exceed 30 µV. Not only that this voltage is difficult to keep stable but also the speed of the charge collection would be incredibly slow.

The solution is to increase the resistivity of the sensor. This is achieved by doping the silicon like a diode and operate the sensor in reverse bias. Thus, the silicon becomes an insulator, and the quiescent current is very low. The high resistivity allows applying high bias voltages—often in the range of several hundred volts—which lead to a fast collection of the charge carriers. The collection time in a fully depleted sensor is defined in [49] by

$$t_c \approx \frac{d^2}{\mu_c V} \tag{3.1}$$

where d is the thickness of the sensor, μ_c the mobility of the charges and V the applied voltage.

Figure 3.3 shows the cross section of a typical p-in-n silicon detector. An n-type substrate builds the base in which the strips are doped as p+-type regions. An insulation layer of silicon dioxide (SiO_2) is placed above the strips, then aluminum electrodes are added on both sides. Due to the insulation layer, the electrodes are de-coupled from the bias voltage. The AC-coupled signal is then amplified and further processed by the detector electronics. Some silicon detectors have the amplifiers and some data processing, e.g. filtering, discretization, integrated with the pixels—these are called Monolithic Active Pixel Sensors (MAPS).

The outer tracker of the original CMS consists of p-in-n silicon detectors with thicknesses of 320 µm and 500 µm [7]. The bias voltage at the sensors is up to 500 V, and the sensor readout is DC-coupled. For the CMS outer tracker at the HL-LHC, n-in-p sensors are under development [5]. Both AC- and DC-coupled readout will be used: AC-coupled readout for the strip sensors and DC-coupled readout for pixel sensors.

3.2. Evolution of silicon trackers

First experiments with particle detectors based on semiconductors took place at the Bell Telephone Laboratories in Murray Hill, New Jersey in 1950 [50]. A pn-junction of a germanium detector in reverse bias was bombarded by alpha

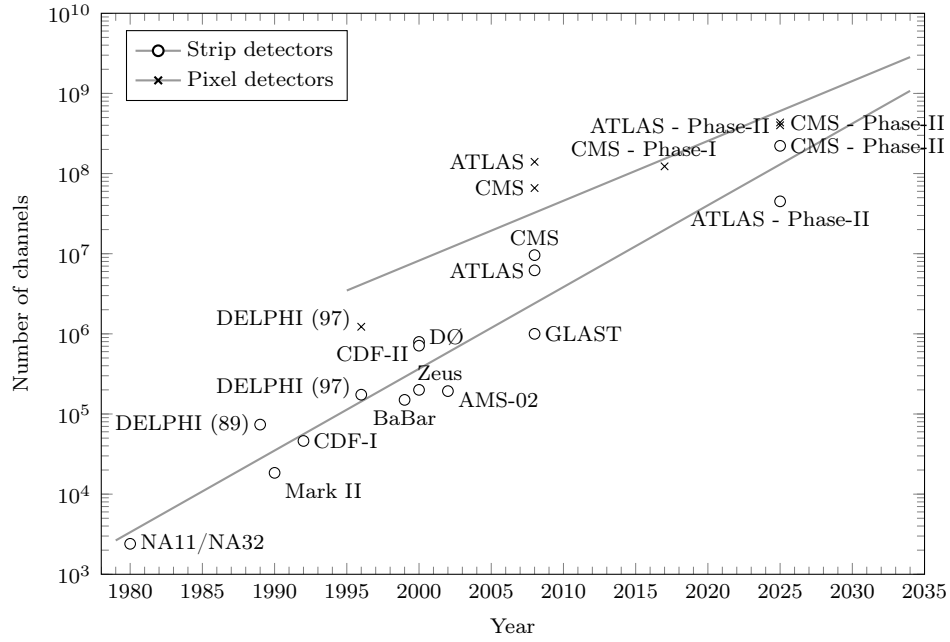
particles, and the generated charge was collected and amplified—basically, the same principle as applied today. Around 1960, monocrystalline silicon became available, and the research continued mainly with silicon detectors [51]. In the early 1970s, the development of particle strip detectors started. Among others, the Kernforschungszentrum Karlsruhe was involved, and a striped silicon detector for digital position encoding was presented in [52]. However, these first sensors were not used in high-energy physics experiments but as spectrometers for energy measurement.

The first high-energy physics experiment that included a silicon strip detector as a tracker was the NA11 experiment at CERN in 1980 [53]. The goal of this experiment was to study the production and properties of charmed particles, i.e. composite particles that contain at least one charm quark. The short lifetime of these particles of the order of 10^{-13} s requires a detector with high spatial resolution. To fulfill this and other requirements, a silicon strip detector with 1200 strips was developed of which every third was read out. NA11 was a fixed target experiment where a beam of protons hit a block of beryllium. Particles produced by collisions of the protons with beryllium atoms were detected by six strip detectors located upstream the beryllium target. In total, the tracker had 2400 readout channels.

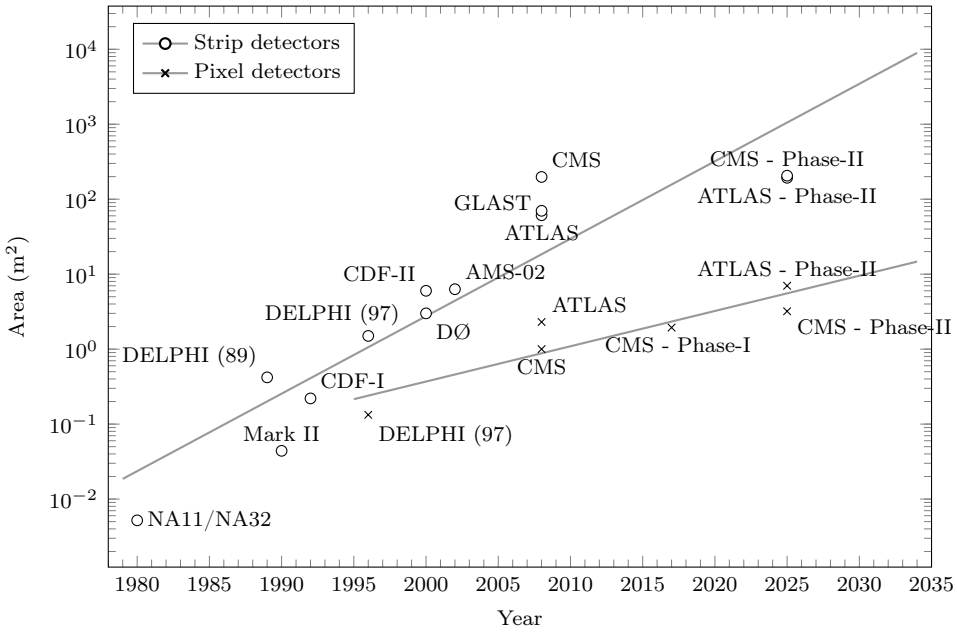
In 1989, the "Detector with Lepton, Photon and Hadron Identification" (DELPHI) experiment at the LEP included the first silicon tracker similar to the one at CMS. Silicon strip detectors were arranged on a barrel-like structure around the electron-positron interaction point [54]. Three layers of in total 96 detector modules provided about 73 000 readout strips. Later in 1997, the detector was upgraded with better modules and, endcaps with pixels and mini strips were added [55].

Currently, the silicon trackers of the two large state-of-the-art experiments have millions of channels: CMS has more than 9.6 million strips and 66 million pixels, and ATLAS has 6.2 million strips and 140 million pixels. During the upcoming upgrades of CMS and ATLAS, the number of readout channels will be increased again. The CMS silicon outer tracker at the HL-LHC alone will provide data from more than 260 million readout channels to the track trigger. In addition, there are the pixel channels of the inner tracker.

The development of silicon detectors over the last 25 years shows that the experiments include larger and larger silicon trackers. Figure 3.4 shows the evolution of the silicon trackers from the NA11 experiment up to the recent plans for the Phase-II upgrades of CMS and ATLAS. The driving force behind the increasing channel number is, on the one, hand the desire for ever higher



(a) Number of readout channels of silicon trackers.



(b) Active area of silicon trackers.

Figure 3.4.: Evolution of readout channels (a) and area (b) of silicon trackers in high-energy physics experiments. Sources: [5, 7, 24, 51, 53–68].

spatial resolution in the detection of the particle tracks and, on the other hand, the increasing particle density at high-energy, high-luminosity colliders. Under such conditions, a higher channel number facilitates to distinguish between the separate tracks.

However, an increased channel number causes also higher data rates.

$$rate_{data} \propto n_{channel} \cdot f_c \tag{3.2}$$

The collision rate (f_c) is chosen as high as possible to maximize the number of recorded events. Moreover, the amount of data ($rate_{data}$) that can be processed on-line with reasonable effort is limited by the current technology. Then the number of channels ($n_{channel}$) may not be freely chosen but depends on the current data processing technology. Therefore, it is interesting to compare the increase of readout channel numbers of strip detectors with the increase of transistors on chip predicted by Moore's law [69]. Figure 3.4 (a) shows the evolution of the number of tracker channels. The number of strip readout channels increases by a factor of 1.6 every two years, whereas the number of transistors on a chip increase by a factor of 2 every eighteen months. The readout channel number thus increases slightly slower than the transistor count on chips. All data can be found in Appendix D.

3.3. Detector coordinates and track parameters

Different coordinate systems are used for the CMS experiment. These coordinate systems are illustrated in Figure 3.5. When Cartesian coordinates are used, the z-axis runs along the particle beam, the x-axis points to the center of the LHC ring and the y-axis points upwards [70] with the origin at the interaction point. However, a specific coordinate system is often used that reflects better the nature of the detector and the particles. In this coordinate system, the distance of a point from the beam axis is indicated by r, and the angle on the xy-plane is φ. The angle on the plane between the r and the beam (rz-plane) is denoted by θ. Beside θ, also the pseudorapidity η, defined by

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \tag{3.3}$$

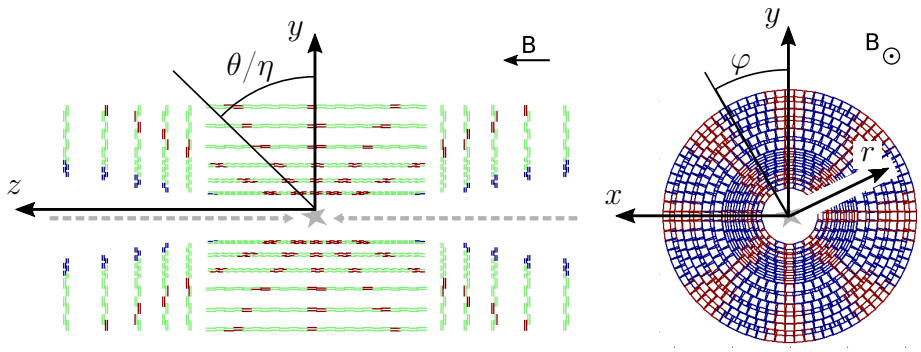


Figure 3.5.: Coordinates in the silicon tracker: On the left side, the detector is cut along the beam axis. On the right side, the detector is cut perpendicular to the beam axis.

is used to measure the angle on the rz-plane. Pseudorapidity has the advantage that within its intervals the flux of particles is approximately constant.

Due to the magnetic field within the CMS tracker, charged particles start to spiral and their track is a helix. Thus, a track is characterized by five parameters, all of them visualized in Figure 3.6. The collision point of the particles and origin of the track is specified by d_0 and z_0. Whereas z_0 indicates the location on the beam axis around the interaction point, and d_0 denotes the closest distance of the helix from the beam axis. The starting angle of the particle track on the $r\varphi$-plane is indicated by ϕ_0 and the angle between the track and the particle beam by θ. The fifth parameter is the curvature R; it denotes the radius of the helix. Via the curvature, the transverse momentum of the particle can be determined.

3.4. CMS silicon tracker layout

The building blocks of the CMS tracker are the detector modules: silicon detectors combined with the readout electronics and the power supply. The tracker layout describes the arrangement of these modules to form the silicon tracker. As the tracks originate from a small volume in the center of the detector a spherical arrangement would be ideal but building a spherical structure is complicated. The compromise is to arrange the modules in a cylindrical structure. The region around the interaction point where the modules are placed on the

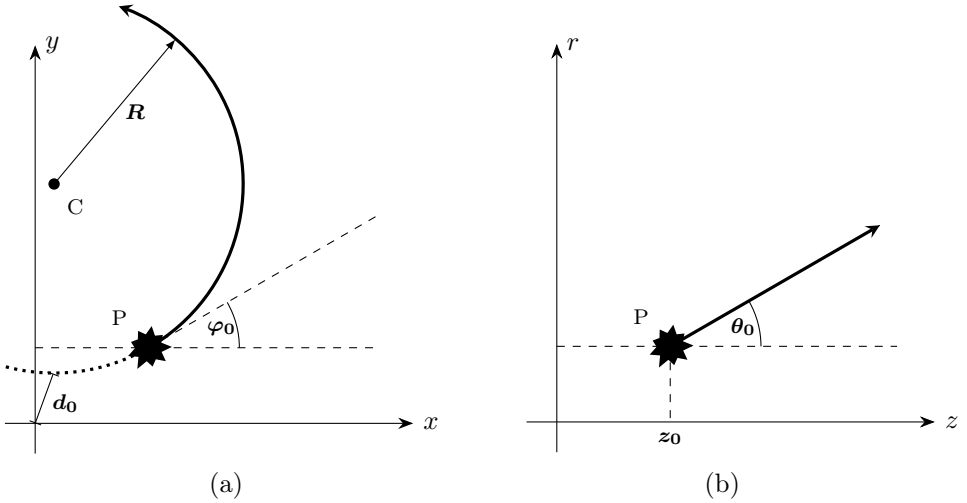


Figure 3.6.: Parameters of a particle track: The left side shows the transversal plane on which the path of the particle is bent. The right side shows the path of the particle on a plane parallel to the beam direction. Source: [71].

curved surface of the cylinder is called the barrel section. Endcaps complement the detector in the front and the back of the cylinder, as shown in Figure 3.7.

At CERN, a tool has been developed that is used to simulate proposed tracker layouts; the result is published in [72], and data are available at [56]. In the same paper, Bianchi presented the new baseline layout, which is called the *tilted barrel geometry* and is shown in Figure 3.7. Its name originates from the fact that some modules in the barrel are tilted towards the interaction point. Thus, the detector modules are more perpendicular to the path of the particles, and fewer modules are needed to detect all particles. This leads to less material in the detector that deflects the particles. The drawback is the more complicated construction of the tracker.

The tracker consists of two parts: the *inner tracker* built of pixel sensors and the *outer tracker* built of stacked modules with strip and macro-pixel sensors. The inner tracker consists of four layers and a number of modules in the endcap regions. Due to the usage of pixel sensors, the spatial resolution of the inner tracker is very high, and the number of readout channels is high too. Therefore, it is not feasible to process the data of the inner tracker by the L1 trigger at the full LHC rate of 40 MHz. The inner tracker data are processed at the high-level trigger only. The outer tracker consists of detector modules, which are composed of two stacked silicon detectors that allow distinguishing the

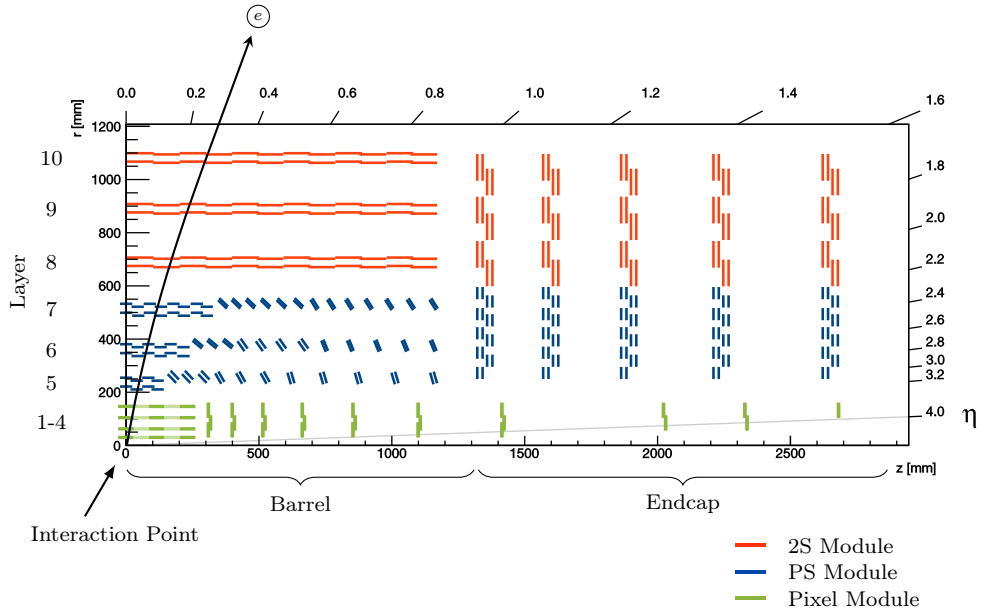


Figure 3.7.: Quarter of the CMS silicon tracker at the HL-LHC. The tracker layout used is `Baseline2015_tilted_Pixel_V1_1`. Source: [56]

transverse momentum of a particle at module level. Two different types of detector modules are used: the Strip-Strip (2S) modules consist of two strip sensors and the Pixel-Strip (PS) modules consist of a strip and a macro-pixel sensor. The detector modules are explained in more detail in Section 3.5.

As compiled in Table 3.1, the silicon tracker consists of 3316 pixel detector modules in the inner tracker and 14 172 stacked modules in the outer tracker. The total active area is around 200 m^2, which corresponds roughly to the area of a tennis court [73].

Strip detectors are used in the outer layers because the track location in the φ-direction is more interesting than in the ϑ-direction. The reason is that the magnetic field of the detector is parallel to the z-axis and deflects the particles in the φ-direction. The physical property related in φ-direction is the momentum that should be measured with high precision. In contrast, the information obtained in ϑ-direction serves to separate particles and interaction vertices. Therefore, the resolution in ϑ-direction may be reduced in the outer layers where the particle density is lower than close to the interaction point.

Table 3.1.: Detector modules according to the `Baseline2015_tilted_Pixel_V1_1` layout. Source: [56]

		Modules	Active Area (m^2)	Channels ($million$)
Outer Tracker	2S modules	8424	154.8	34.2
	PS modules	5748	51.0	187.6
	Total	14172	205.8	221.8
Inner Tracker	Pixel modules	3316	3.2	434.6

3.5. Stacked detector modules

The number of all particles passing through the silicon tracker is of the order of ten thousand. To process all of them by the CMS track trigger and the L1 trigger within the required time is not feasible and, therefore, their number must be reduced. The vast majority of particles have a low transverse momentum. However, particles with high transverse momentum are a hint for interesting physics processes. In a first step, it is enough to process only data from these particles. Therefore, Jones et. al developed a concept to determine the transverse momentum (p_T) of particles and presented it in [46]. The data processing on the detector modules select hits of particles with a transverse momentum above a certain threshold and send them to the CMS track trigger.

The basic idea is to use detector modules that consist of two separate silicon detectors stacked on each other as illustrated in Figure 3.8. The distance between the two detectors is of the order of a few millimeters depending on the module location. When a charged particle passes through the detector module, it causes a hit in both of the silicon detectors. As the particle is diverted to a different extent by the magnetic field depending on its transverse momentum, the location of the hits in the two sensors differs. The distance between these two hits is inversely proportional to the transverse momentum of the particle but also depends on the separation of the silicon detectors, the distance of the module from the interaction point and the magnetic field strength. The distance between the two hits in strips is called *bend*.

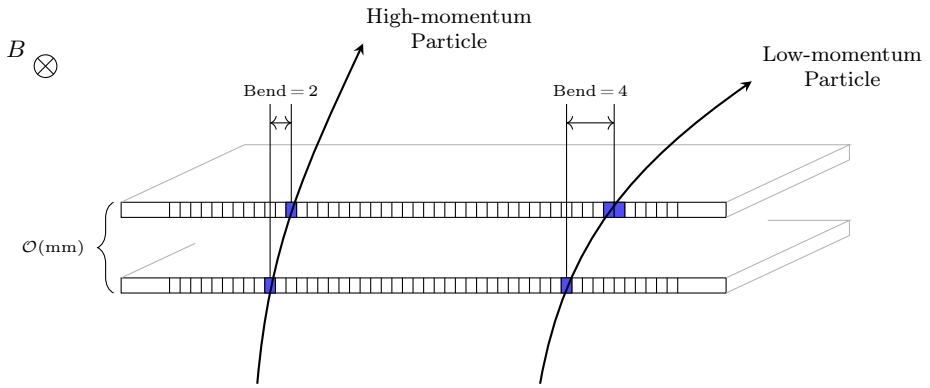


Figure 3.8.: Stacked silicon detector to determine the transverse momentum (p_T) of charged particles.

3.5.1. Composition of the detector modules

The detector modules itself, as shown in Figure 3.9, are self-contained. This means that the module consists not only of the two silicon detectors but also include: readout chips, data concentration chips, an optical transmission system and a power conversion unit. Only two connectors remain: a bi-directional optical link for data transmission and control signals, and a power line. The general composition of the detector module is the same for the 2S and PS modules. They only differ in the implementation of some components. These differences are described in the following Sections 3.5.2 and 3.5.3. The information in this section is compiled from sources of the two module types [5, 74–77].

The central part of the detector modules are the two stacked silicon detectors. In both module types, the silicon detectors are divided into 8×2 areas. Each of these areas is read out by an individual readout chip. Only a few strips and macro-pixels at area boundaries are processed by two adjacent readout chips. The two symmetric parts of the detector module are part of the address that is used to identify a strip and are called z-segment.

The readout chips accommodate the main intelligence of the detector modules. The strips and macro-pixels of the silicon detectors are directly connected to the readout chip. In a first step, the signal is amplified and compared with a threshold to distinguish particle hits from noise. The amplitude information of a hit is lost in this step and, therefore, the technique is called binary readout. The discretized hits are stored in a readout pipeline to be read out as raw data in case of a positive trigger signal. This readout pipeline is 512 cells deep

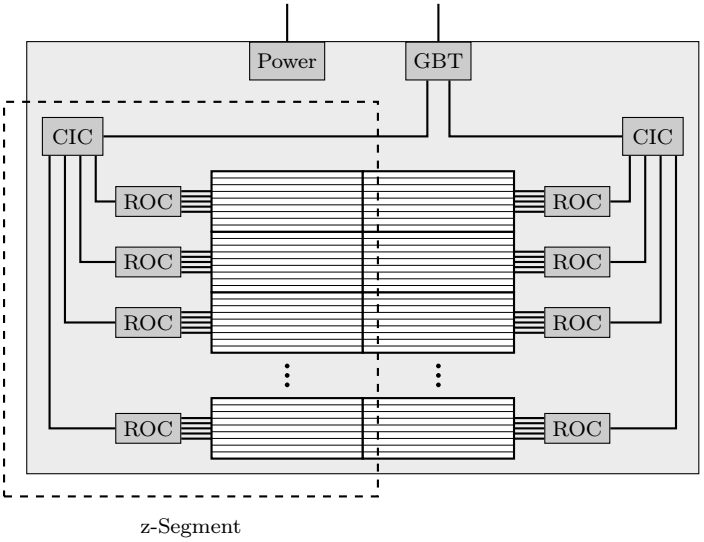


Figure 3.9.: Generalized structure of the detector modules, applies to both the 2S and PS modules.

and, thus, large enough to store all events during the 12.8 µs latency of the L1 trigger.

In the next processing step in the readout chip, adjacent hits are combined into clusters, and only the center of this cluster is passed on. As the center of a cluster could lie between two strips, virtual intermediate strips are added. Thus, The resolution of a stub is twice the resolution of the silicon detectors.

For the discrimination of the transverse momentum of the particles yet another step is necessary. For each cluster on the inner silicon detector, a cluster in the outer silicon detector is searched within a coincidence window. The width of this coincidence window is programmable and defines the minimal transverse momentum of a recognized track. If a pair of hits is found within the coincidence window, their data are transferred to the Concentrator Integrated Circuit (CIC).

Two hits that are allocated to a specific track in this way are called a stub—a common expression used in the CMS track trigger community. A stub consists of two elements: the location and the bend. The location corresponds to the relative cluster position in the inner silicon detector. The bend is the distance between the hit on the inner and the hit on the outer silicon detector. It is

measured in a number of strips along the detector and is a rough measure of the transverse momentum.

The CIC collects the data of eight readout chips, which corresponds to one z-segment of the detector module. The CIC receives both trigger data and raw data, arranges them in packets and puts them into a single data stream. As the data rate of the output stream is limited, only a limited number of stubs can be transmitted off-detector. However, stubs are not detected by all the readout chips in every clock cycle. Therefore, the CIC collects the stubs over eight clock cycles to balance the data rate. Only the stubs with the highest transverse momenta within that time frame are kept. If there are more stubs as the ones that can be transmitted, they are discarded. Simulations have shown that the tracker efficiency reduction by the limitation of transmitted stubs is acceptable. The CIC adds the readout chip number and a timestamp, representing the position within the eight clock cycle window, to the selected stubs. These prepared stubs are then transmitted to the optical transmission system during the subsequent eight clock cycles.

In the final step, the radiation-tolerant optical transmission system developed in the GigaBit Transceiver (GBT) project at CERN sends the data off the detector. The GBT ASIC merges the data streams from the CIC chips of the two module z-segments into one stream and transmits the merged stream over an optical fiber off-detector.

3.5.2. 2S detector module

Detector modules with two stacked strip silicon detectors are installed in the outer three layers of the outer tracker and the outer parts of the endcaps. Both silicon detectors of these so-called 2S modules are of the same type with 2×1016 strips with a size of $50\,\text{mm} \times 90\,\mu\text{m}$ each. The active area of the 2S module is $\sim 10 \times 10\,\text{cm}^2$ [78]. The distance of both sensors is 1.8 mm or 4 mm depending on the distance from the interaction point.

The readout chip of the 2S module is the CMS Binary Chip (CBC) [77, 79]. As illustrated in Figure 3.10, the CBCs are mounted on a flexible hybrid close to the silicon detectors, which are wire-bonded to the hybrid. One CBC reads and processes 127 strips from each of the two silicon detectors. For every bunch crossing, the CBC transmits up to three stubs to the CIC located on the same hybrid. In the rare case that more than three stubs are detected by a single CBC, the stubs with the highest transverse momenta are selected.

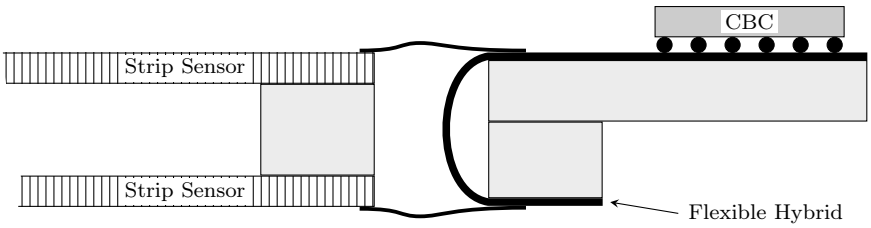


Figure 3.10.: Cross section of the 2S detector module. Source: [80].

As described above, the CIC chip collects stubs from all connected CBCs over eight consecutive clock cycles. Of these accumulated stubs, the CIC selects the twelve with the highest transverse momenta, packs them together with buffered raw data and sends the packet to the GBT ASIC.

3.5.3. PS detector module

The inner three layers of the outer silicon tracker and the inner parts of the endcaps are built of PS modules [74, 78], which are roughly half the size of the 2S modules. They consist of two different silicon detectors one with macro-pixels and one with strips. Both have the same active area of around $44\,\mathrm{cm}^2$ [78]. The outer silicon detector is a strip sensor with 960×2 strips of the size $100\,\mu\mathrm{m} \times 25\,\mathrm{mm}$. For the inner sensor, each strip is segmented into 16 parts—the macro-pixels. The size of a macro-pixel is $100\,\mu\mathrm{m} \times 1.5\,\mathrm{mm}$ and each silicon detector has 960×32 macro-pixels. In total, a PS module has 30 720 macro-pixels and 1920 strips.

Two different readout chips are used to read out the two different silicon detectors. The chip that reads the data from the strip sensor is the Strip Sensor ASIC (SSA). Similar to the CBC on the 2S modules, eight SSAs are mounted on the hybrid along the sides of the silicon detector, each processing 120 strips. The strip silicon detector is wire-bonded to the hybrid. In contrast to the CBC, the SSA only performs the analog processing and discretization of the sensor signal, but not the further processing steps.

The readout chip of the macro-pixel sensor is the Macro-Pixel ASIC (MPA) [75]. Sixteen of the MPAs are bump-bonded directly to the sensor die. This arrangement is shown in Figure 3.11. Besides the analog processing and discretization of the macro-pixel signals, the MPA also performs the cluster building and stub generation. The readout pipeline for the raw data is also located within the

MPA. To balance the data rate, the MPA buffers all stubs for two bunch crossing and sends then the four stubs with the largest bend in the two subsequent clock cycles.

The CIC of the PS module is located on the hybrid. Thus, the MPA transfers the created stubs via bond wires to the hybrid. Also, the CIC on the PS module accumulates the stubs over eight bunch crossing. As the size of a stub with macro-pixel information is larger (20 bit) than the one of the 2S module (16 bit), the PS modules send only the ten stubs with the highest transverse momentum off-detector.

3.6. The GBT optical transmission system

Under the conditions at the HL-LHC, the communication between detectors and the counting room faces two big challenges: the high data rates and exposure to high radiation fluency. As this is an issue that applies to all experiments, a joint project between CMS and ATLAS has been initiated to develop a suitable data transmission system. The goal of the GBT project [81, 82] is to develop a bi-directional optical transmission system that withstands the high radiation doses and strong magnetic field at the detector. The GBT communication incorporates all the different transmission channels needed from and to the detector module. These are timing and trigger signals to the detector, the trigger and DAQ signals from the detector, and slow control signals. Employing just one link for all these functionalities simplifies the link topology significantly and reduces the material in the detector. However, it becomes more challenging to provide accurate timing to the detector modules that is necessary for the triggering.

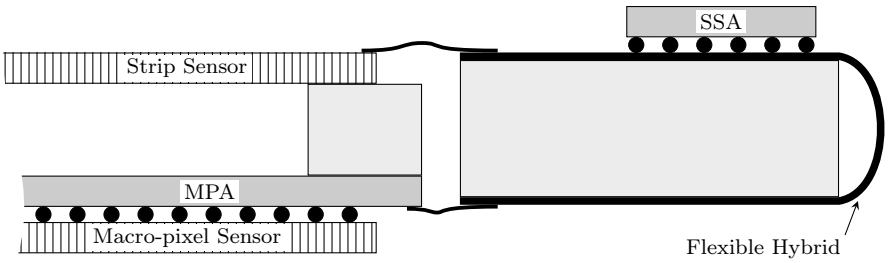


Figure 3.11.: Cross section of the PS detector module. Source: [74].

The GBT ecosystem covers the complete communication chain from the output of the front-end chips to the recovered data stream at the back-end. Multiple ASICs are under development: a transimpedance amplifier for the receiving photodiode, a laser driver for the transmitter, a serializer/de-serializer chip (GBTX) and a chip for slow-control (GBT-SCA). All of them are designed for the usage on the detector and to withstand high radiation doses. Off-detector, the usage of commercial components is foreseen. Thus, an FPGA Intellectual Property (IP) core is under development that implements the GBT protocol on the off-detector side. For the physical layer, GBT incorporates the Versatile Link project [83]. The goal of the Versatile Link project is to define all necessary components for a radiation-hard physical link for the detector data. Within the project, a radiation-hard transceiver for use in the detector that fits into a Small Form-factor Pluggable (SFP) plug is developed. In addition, possible candidates for the fiber and commercial transceivers for the off-detector usage are evaluated.

The serial data transmission on the optical link runs at 4.8 Gbit/s. To reduce the effects of radiation, the GBT protocol includes a Reed-Solomon error detection and correction code. Thus, for 80 bit of data 32 bit of coding overhead are transmitted. Another 8 bit carry the header and slow control information. The available bandwidth for detector data is therefore 3.2 Gbit/s. However, a faster version of the GBT link is also under development, which will provide bandwidths of 4.5 Gbit/s and 9.1 Gbit/s. Towards the sensor side, the GBTX provides either a 40 bit double-data-rate electrical bus or multiple serial electrical links to connect the front-end electronics. Summing up, GBT provides a transmission system that is highly configurable and suits many types of detector readout situations.

4. The CMS Track Trigger

In high-energy physics, (particle) tracking is understood as the measurement of the path and the transverse momentum of a particle. Two parts are necessary for the tracking: Firstly, a detector that detects space points (hits) along the path of the particles, e.g. the silicon tracker of CMS described in the previous chapter. Secondly, a device that processes the data produced by the detector, e.g. the CMS track trigger, which is described in this chapter.

Figure 4.1 illustrates the simplified functionality of a tracker processor. As shown in the top part, the input of a tracker processor, such as the CMS track trigger, consists of a cloud of particle hits in the detector. In a first step (track finding), the subset of hits that belong to one track is identified and combined to a so-called track candidate. This problem is similar to draw-by-number known from children books. In the second step (track fitting), a track is fit to the hits of each track candidate. The resulting track parameters are the ones presented in Section 3.3. The analysis of the tracker data can take place at different stages of the data processing: at the different trigger levels or only in the final data analysis.

4.1. History of track triggers

Two examples of experiments that incorporate a hardware tracker processor as part of the trigger are presented next. Firstly, the track trigger of the Collider Detector at Fermilab (CDF) experiment for which the first version of the AM Chip was developed. Secondly, the Fast TracKer (FTK) of ATLAS that performs the tracking after the L1 trigger but comprises many concepts that will also be used for the CMS track trigger. These two have been chosen, as concepts developed within these two track trigger projects have been adopted in the Associative Memory (AM) approach of the CMS track trigger, presented in detail in Chapter 5.

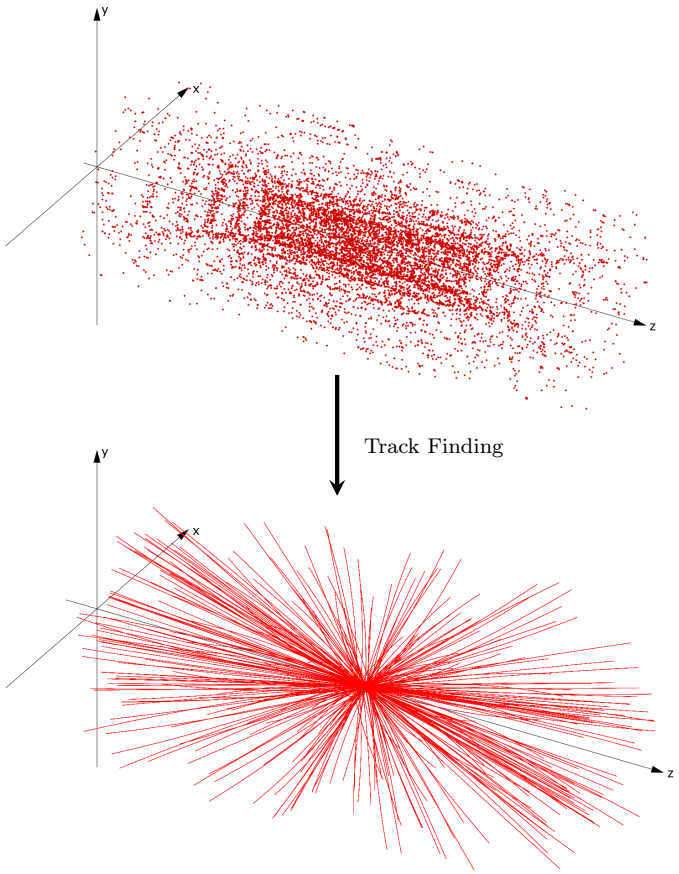


Figure 4.1.: The concept of track finding: The top plot shows the cloud of hits that is the input to the track processor. The bottom plot shows the tracks taht are found within these hits.

4.1.1. Track triggering at CDF

In 1987, the fast hardware track-finder [84] was added to the CDF experiment at Fermilab. The goal of the fast hardware track-finder is to identify tracks with high transverse momentum in the data from the Central Tracking Chamber (CTC). As in the CMS track trigger, a threshold for a minimum transverse momentum of a track was set. In contrast to the CMS track trigger, the data originates from a wire drift chamber (CTC) and not a silicon tracker. A silicon tracker did not exist at the CDF experiment at that time. The chamber consists of 4392 wires on nine superlayers.

The CDF experiment had a three-level trigger system. To the first trigger level, the fast hardware track-finder delivers a decision signal based on the transverse momenta of the tracks with a latency generally below 3 µs. More detailed data about the tracks are delivered to the second level trigger within 8 µs. Data are processed by a 19-stage digital pipeline that is built of combinatorial logic components, Programmable Logic Arrays (PLAs) and Random Access Memory (RAM) table lookups. In 2000, after the run II upgrade of CDF, a new drift chamber was installed, and the fast hardware track-finder was replaced by the eXtremely Fast Tracker (XFT) [85].

During the run II upgrade, the CDF Secondary Vertex Trigger (SVT) [86] was added to the level 2 trigger to achieve a better tracking performance and to trigger on the displayed vertices from heavy quark decays. The SVT processes the data from four of the five layers of the Silicon VerteX detector for run II (SVXII) [61]—the silicon tracker of CDF. The SVXII is the central part of silicon tracker system of CDF and has about 405 000 strip channels [58]. Additionally, the SVT requires a coincidence with a track from the XFT. To keep up with the requirements of the Level-2 trigger (L2 trigger), the latency of the SVT is around 30 µs at an input rate of 350 Hz.

The system architecture of the SVT can very much be seen as the predecessor of the CMS track trigger. It is already based on the two steps: coarse resolution track finding by AM Chips and track fitting in FPGAs. The AM Chip is an ASIC specifically designed for finding particle tracks in data from silicon trackers. The first generation of these AM Chips (AM01) with a capacity of 128 patterns were applied to the SVT. The detector is split into twelve sectors, each covered by a bank of 256 AM Chips and 32 000 patterns. To cover the entire detector, 3072 AM Chips with 384 000 patterns are necessary. The key difference of the SVT compared with the CMS track trigger are that the SVT appears on the second trigger level and that the SVXII had only about 405 000 channels in total.

To keep up with the increasing luminosity at the Tevatron, the hadron collider of Fermilab, the SVT needed to be upgraded [87]. The goal was to introduce as little new hardware as possible. Therefore, the main change was the replacement of the AM Chips by AM Chips of the newest generation (AM03) with a capacity of 5120 patterns. Thereby, 6.1 million patterns could be compared by the AM Chip banks in total.

4.1.2. ATLAS Fast TracKer (FTK)

Also, the ATLAS experiment needs to be upgraded because of the increasing luminosity and will introduce a track processor—the ATLAS FTK [88–90]. This upgrade at ATLAS will already be installed during the phase-I upgrade in 2018 and 2019. The FTK may be seen as a kind of predecessor of the CMS AM trigger concept as presented in this thesis. Many ideas are adopted from the functionalities of the FTK.

In contrast to the CMS track trigger, the ATLAS FTK provides tracks of particles with transverse momenta larger than 1 GeV/c (rather than 2 GeV/c) to the L2 trigger (rather than the L1 trigger). The FTK also processes data from the pixel sensors and not only from the strip sensors like the CMS track trigger. In the current ATLAS trigger system, track processing takes place in the L2 trigger. Figure 4.2 gives an overview of the ATLAS trigger system and the integration of the FTK. The L2 trigger is based on Central Processing Units (CPUs), and the calculation of the tracks is computationally intensive. By moving the track processing to hardware, the track parameters are calculated with a lower latency. Additionally, the freed computing power can be allocated to other tasks. The FTK works at the full L1 trigger rate of 100 kHz, and the data processing takes 100 µs.

In a first step, the FTK searches for clusters of activated strips and pixels in the tracker data coming from the Read-Out Drivers (RODs). Then the data are split into 64 sectors that are processed independently. Afterward, the data from eight of eleven tracker layers are processed with a coarse resolution by the track finder based on AM Chips. The data of the remaining layers are directly sent to the second fitter stage.

The FTK uses the AM Chips for track finding [92], the same way as it was done in the SVT. The updated version of the AM Chip, the AM06, now stores 128 000 patterns. Section 5.4.1 describes these chips closer. In total 1 billion (10^9) tracks are stored and may be found in the complete FTK, which correspond to approximately 8000 AM Chips. The AM processor with the AM06 chips is able to perform 8×10^{17} comparisons per second.

After track finding, the hits belonging to found tracks are fitted with full resolution but only with the hits from the eight layers involved in the track finding process. Due to the nature of the AM-based track finding, duplicate tracks and fake tracks occur in the data. After the first track fitting stage, these duplicates and fakes are eliminated as much as possible. Then the tracks are merged with

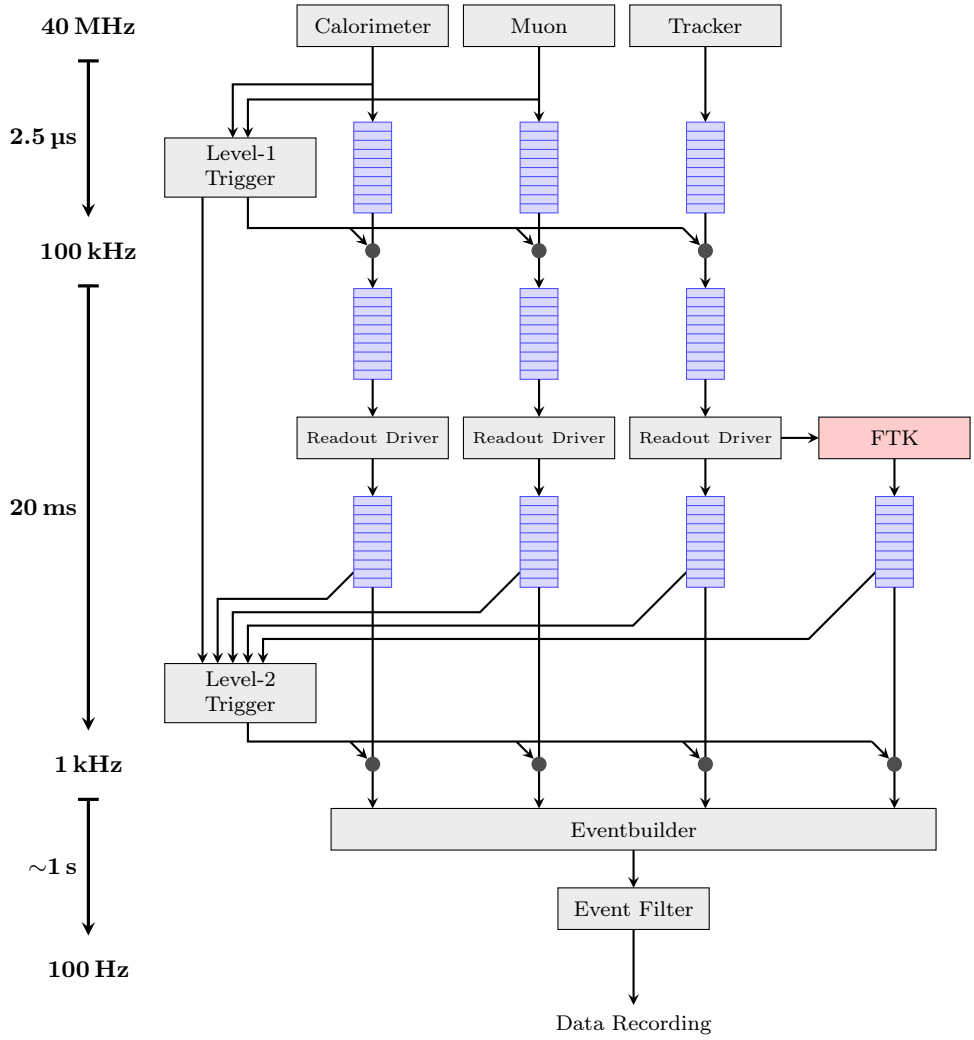


Figure 4.2.: Trigger System of the ATLAS experiment after the addition of the FTK during the Phase-I upgrade. Source: [91].

the data of the remaining layers and fitted a second time. The fitting algorithm used is a linearized fit [40].

4.2. Requirements of the CMS track trigger

The large number of 140 or even 200 colliding particles in each event will be a major challenge for the L1 trigger of CMS at the HL-LHC. The information provided by the calorimeters and the muon system will not be sufficient to keep the trigger rate below 1 MHz. In particular, the momentum resolution of the muon system will not be high enough under the new conditions. Therefore, it is essential to include the data from the silicon tracker in the L1 trigger decision. The CMS experiment will be the first collider experiment that incorporates a track trigger for the silicon tracker at the first trigger level.

Although the CMS track trigger is called a trigger, it is not a trigger in the traditional sense, as it does not reduce the recorded event rate by itself. In fact, the CMS track trigger is a part of the global L1 trigger, see also Figure 2.7. The CMS track trigger preprocesses the data from the silicon tracker, selects the high-momentum tracks and delivers their parameters to the global L1 trigger. The global L1 trigger takes the decision, whether an event is interesting or not, based on the combined data from the track trigger, the calorimeters and the muon system. As the latency of the CMS track trigger and the L1 trigger must be very low, both are located in the counting room next to the CMS experiment. Thereby, the propagation delay on the optical fibers is kept low.

As the transverse momentum of about 99 % of the particles in an LHC event is below 2 GeV/c [93], the output data rate can be reduced significantly by the stacked detector modules. Nevertheless, the remaining amount of data transmitted by the silicon tracker is huge, and the processing of them is challenging. Accordingly, the requirements for the track trigger processor are enormous:

- **Latency below 5 µs** — This latency corresponds to the time from the collision event until the track parameters are available to the L1 trigger. Thus, it also includes the processing on the detector modules and the time for the data transfer which alone takes about 1 µs.
- **Event rate of 40 MHz** — Every 25 ns a new event is delivered to the track trigger processor. By applying time multiplexing and pipelining, this can be handled extensively.

- **Input data rate of nearly 100 Tbit/s** — While data may not be transmitted for every event from every module, optical links with that total bandwidth are connected to the track trigger processor. During each event, in average 15 000 hits belonging to roughly 300 particles arrive on the links.
- **High efficiency and low fake rate** — On the physics side, the track trigger should ideally identify all particles (efficiency), but should not find tracks that do not belong to any particle (fakes).

As a consequence of these challenging requirements, especially the low latency, the track trigger processor has to be built with dedicated hardware. The designs currently considered are based mainly on FPGAs, but also the application of dedicated ASICs is investigated. The final track trigger processor will consist of 1000 to 2000 FPGAs.

4.3. Proposed CMS track trigger concepts

Almost ten years before the CMS track trigger is brought on-line in 2025, multiple research groups investigate three quite different concepts for the CMS track trigger [93]. KIT is mainly involved in the AM approach, which builds on a highly specialized associative memory ASIC for track finding. Another approach adopts the time multiplexing trigger hardware concept of the CMS ECAL trigger and implements the CMS track trigger on this platform. A third approach is based on the extrapolation of pairs of hits in the silicon tracker—the tracklets—to other detector layers.

The year 2016 is particularly important, as in May a pre-review, and in December the actual review of the different approaches take place. At these reviews, the different research groups present the concepts and report on the status of their R&D. Afterward, the CMS collaboration will decide in which direction the development will continue, and finally, one concept will be implemented.

As all concepts are under development, the values presented here are preliminary values and correspond to the status at the May pre-review.

4.3.1. Overview of the AM approach

This section summarizes the most important facts of the AM approach. Figure 4.3 shows a simplified data flow of the AM approach. As the system sim-

ulation of the CMS track trigger (Chapter 7) is based on the AM approach, a detailed description of the AM approach follows in Chapter 5.

To reduce the amount of data that has to be processed by one track trigger processor, the AM approach splits the detector into 48 independent geometrical sectors—8 in φ and 6 in η. Such a sector is also called trigger tower. The splitting of the detector data is possible because the paths of high-momentum particles passing through the silicon tracker are not bent much. However, the tracks of the high-momentum particles are not completely straight, and some detector modules at the sector boundaries belong to two sectors. The data of these modules are duplicated at the Data, Trigger and Control Boards (DTCs). The DTCs are the first boards on which the data from the detector arrive and are a common element among all three different approaches. For the AM approach, the data of each sector are transferred from the DTCs to the trigger tower processor.

All the data of one sector is processed within one trigger tower processor. Multiple processing boards exist within a trigger tower processor to provide the single processing boards enough time to process the data. The data are distributed by round-robin scheduling to the separate processing boards and, thus, implementing time multiplexing of the processing boards. Consequently, all the data of one sector that belong to one bunch crossing is processed by a single processing board. The time multiplexing and data distribution are implemented in the so-called Data Organizer (DO), which is the first processing element within the trigger tower processor.

The actual processing of the tracker data takes place on the processing board. The first step is the track finding by the AM Chips [94]. The AM Chip is an ASIC that is specifically designed for track finding in tracker data. Patterns

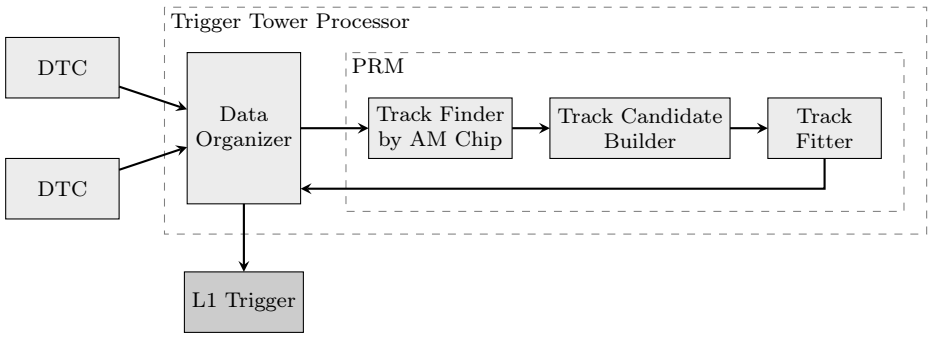


Figure 4.3.: Simplified data flow of the AM approach.

that represent the track of a particle within the detector can be stored within the AM Chip. One pattern is stored within the AM Chip for each high-momentum track which should be found by the CMS track trigger. All the data belonging to a bunch crossing are fed to all AM Chips of the processing board and are compared with the stored patterns. The AM Chips return then all the tracks that were found within the data set of the bunch crossing.

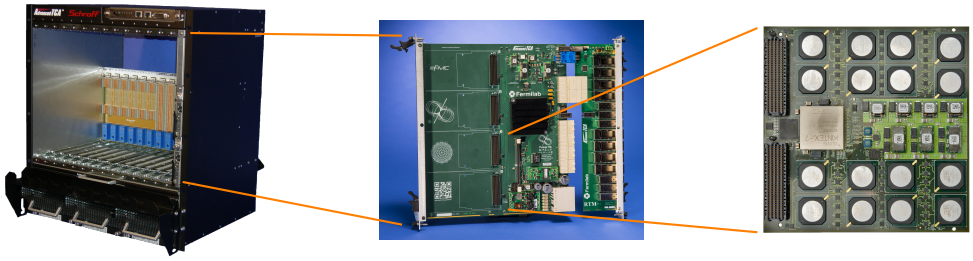
Due to the limited resolution of the track finding, as explained later, it is possible that the data set of a track may have multiple hits on each layer. However, the implemented track fitting algorithm requires clean data sets with at most one hit per layer. Therefore, a track candidate builder is added to clean such data sets. To achieve this, the track candidate builder builds pairs of hits on the inner layers [71]. These pairs are then projected in 3D to the other layers, and on each layer just the closest hit is selected for the track candidate.

In the final processing step, a fit is applied to each track candidate. The AM approach employs a linearized track fit with constants calculated by the Principal Component Analysis (PCA) method [95]. The parameters are then transmitted back to the DO and forwarded to the L1 trigger.

The current research of the AM approach CMS track trigger adapts the Pulsar IIb hardware platform from ATLAS FTK [88]. Figure 4.4 provides an overview of the hardware components of the AM approach. The Pulsar IIb is an Advanced Telecommunications Computing Architecture (ATCA) blade that can host up to four FPGA Mezzanine Cards (FMCs). The processing board itself is a double-wide FMC and, thus, two of the actual processing boards can be mounted on one Pulsar IIb. The processing board, which is named Pattern Recognition Mezzanine (PRM), is specifically developed for the CMS track trigger. It is expected that 20-fold time multiplexing is sufficient to get the data processed by the processing boards. As one ATCA crate can host ten Pulsar IIb blades, that carry processing boards, one crate is necessary for each trigger tower processor. Thus, the CMS track trigger will be comprised of 48 crates. In the future, the number of crates may be reduced to 24 or even fewer.

4.3.2. Time Multiplexing Track Trigger (TMTT) approach

Another discussed approach for the CMS track trigger is based on a time multiplexed trigger concept, which has been developed for the ECAL trigger for the Phase-I upgrade of CMS [96, 97]. Hall et al. presented it in [98, 99]. The data processing of the Time Multiplexing Track Trigger (TMTT) approach is



ATCA crate Pulsar IIb PRM

Figure 4.4.: Overview of the hardware used for the AM approach CMS track trigger. Photos courtesy of: Pentair/Schroff, Fermilab.

performed by FPGAs only. The main idea of the time-multiplexing concept is that for each event the data of the whole detector, or at least a large part of it, are processed within a single processor node. However, different events are processed by different processor nodes. Therefore, the processing takes place parallel in time and not parallel in towers as with the AM approach. This is especially useful when an algorithm works on global data. In doing so, the time-multiplexing trigger concept avoids data sharing between processors. As the tracks processed by the CMS track trigger are relatively straight and, therefore, the algorithms do not need data of large parts of the detector, this point is of no advantage for the CMS track trigger.

Figure 4.5 visualizes the concept of a time multiplexed trigger. The data from the detector are sent to a first layer of front end-boards. Each front-end board receives the data of the connected detector modules for every single event. The different events, visualized by different colors, arrive after each other at the front-end boards. In the case of the CMS track trigger, the DTCs implement the first layer, and there is no difference to the AM approach so far. The track processors form the second layer and perform the actual tasks of track finding and track fitting. The data of one event (same color) is processed by only one track processor. The data from different events (different colors) are processed by different track processors. The task of the first layer is to transfer the data of one event to the corresponding track processor. As every first-layer board needs to be connected with every track processor in the second layer, the two layers are connected by an interconnect with a high bandwidth.

As each track processor needs to process all the data of one event, every track processor is exactly the same. The track processors are not only built on the same hardware but are also configured with the same firmware and the same

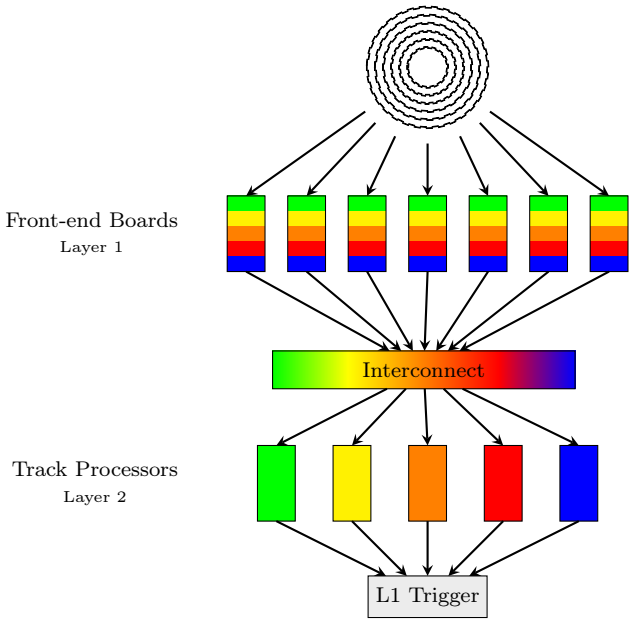


Figure 4.5.: The time-multiplexing concept applied by the TMTT approach. The different colors visualize data of different collision events. Source:[99].

parameters. This is the biggest advantage of the time-multiplexing approach for the CMS track trigger. To employ only equal processing nodes facilitates the expansion of the system, i.e. in the case of missing computing power simply more track processors are added. The additional track processors can also be used to create redundancy to ensure the track trigger functionality in case of broken track processors.

Unlike the time-multiplexed trigger for the ECAL at the Phase-I upgrade, the TMTT is not able to transfer the data of the complete tracker to one track processor. The number of optical links would be simply too large. Therefore, the TMTT splits the detector into eight sectors and duplicates the data at the sector borders. Compared to the AM approach with 48 sectors, the TMTT approach processes a much larger part of the detector within one track finding processor. The assignment of the stubs to a sector and the transformation from detector-specific to physical coordinates is done in the DTCs.

Any further processing takes place at the track processor whose data flow is shown in Figure 4.6. The first step in the track processor is the geometric processor. Due to the smaller number of sectors in the TMTT, the number of

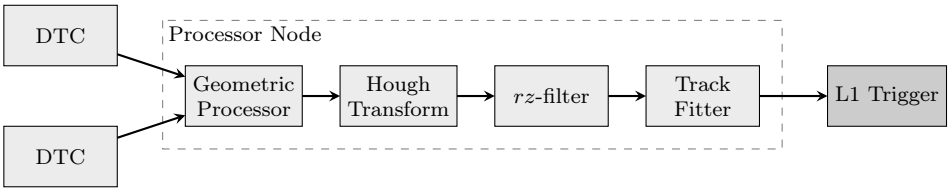


Figure 4.6.: TMTT data flow.

stubs that need to be processed by a track processor is higher than at the AM approach. To facilitate the processing of this amount of data, the stubs are assigned to geometric segments, which can be seen as a kind of smaller sectors. Each sector is divided into four segments along φ direction and nine along η. In the following steps, each geometric segment is processed by an independent processing block. The geometric processor also performs some preprocessing on the stub data, such as coordinate transformations.

The stubs of each geometric segment are then processed separately by the track finding stage that is implemented on the base of a Hough transform in an FPGA [100, 101]. The Hough transform is an image processing method that facilitates finding lines within an image [40]. The TMTT applies the Hough transform to the data of a geometric segment in the $r\varphi$-plane to find particle tracks. The Hough transform takes the coordinates of a stub and transforms them into a line in the transformed space. A point in the transformed space represents one track in the detector, and its coordinates are the track parameters: φ_0 and R. The transformed line represents all tracks that possibly go through the given stub coordinates. As shown in Figure 4.7, the Hough transform is applied to every stub of the geometric segment. The points of the resulting lines are then accumulated in a 32×32 histogram. Because all stubs that lie on the same track cross at one point in the transformed space, a peak appears in the histogram at locations that correspond to a detected track. Track candidates are then created from histogram positions that have been filled by stubs from at least five different layers At the end, the stubs that build a track candidate are transferred to the rz-filter.

The track candidates are built only in the $r\varphi$-plane. Therefore, it is possible that stubs of a track candidate do not belong to the same track because they do not lie on a line in the rz-plane. The rz-filter removes these undesirable stubs. The working principle is similar to the track candidate builder (Section 5.2.4) of the AM approach. Firstly, seeds are built from all possible combinations of two stubs from the PS modules. Secondly, seeds whose origin is not close enough

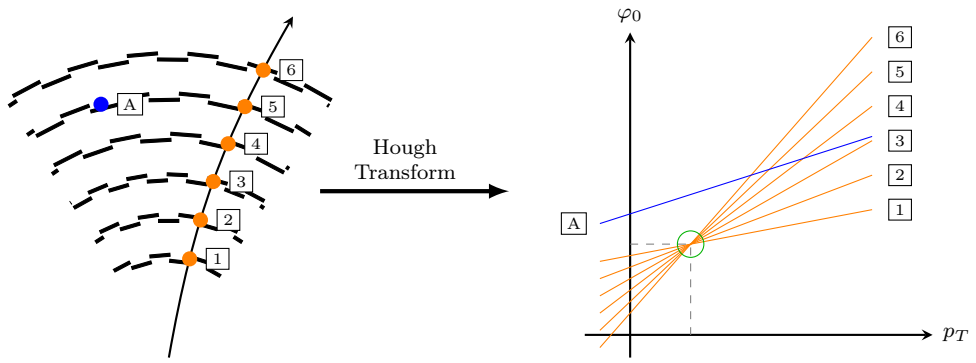


Figure 4.7.: The principle of track finding by Hough transform: The six hits (1-6) of the particle produce six lines in the transformed space. These six lines cross each other in a single point whose coordinates correspond to the parameters (φ_0, p_T) of the track. The line generated by the single hit (A) does not cross this point, as it does not belong to this track. Source: [100].

to the interaction point or cross the geometrical section border are rejected. Lastly, the remaining seeds are projected to the other layers, and stubs close to the projected points are searched.

The track candidates with enough stubs are then processed by the track fitter that determines the real track parameters. Two possible track fitting algorithms are under discussion. The first option is a linearized χ^2 fit as it is used by the tracklet method, described in the next section. The alternative is a Kalman filter based track fitter [40]. The last stage is a duplicate removal whose task is to remove all tracks that have more than four stubs in common with another stub. The track with stubs on the fewest layers would be removed. This step may be combined with the rz-filter or the track fitter.

The groups around the TMTT have developed a very powerful processing board—the Imperial Master Processor Virtex-7 (MP7) [102, 103]. The MP7, as shown in Figure 4.8, has been designed as a generic processing board according to the Micro Telecommunications Computing Architecture (MTCA) standard [104]. It consists of a large Xilinx Virtex-7 FPGA and an optical interface with an enormous bandwidth of 740 Gbit/s in each direction. To provide this bandwidth, the MP7 contains six Avago MiniPOD [105] receivers and six Avago MiniPOD transmitters. Each of the Avago MiniPODs provides twelve optical links running at 10.3 Gbit/s. Around the MP7 board, a firmware framework has been developed that provides functions to communicate with

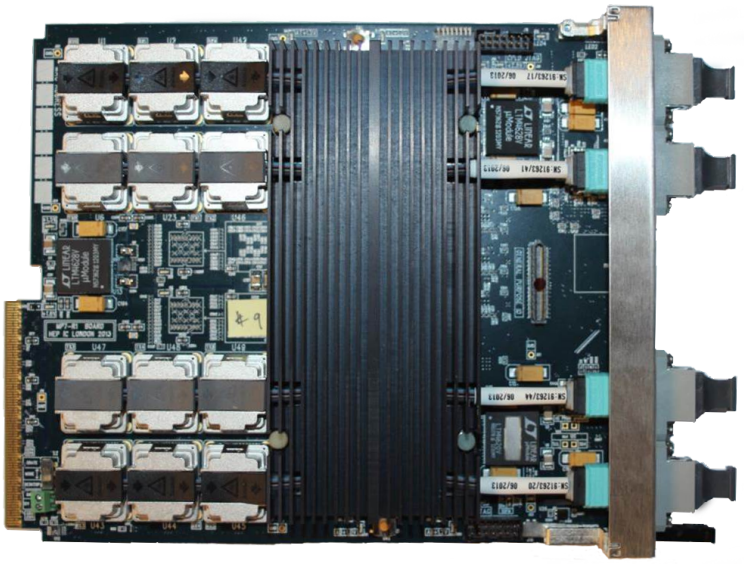


Figure 4.8.: The MP7 processor board with the MiniPOD optical connectors on the left hand side and the FPGA in the center covered by the heat sink. Courtesy of Imperial College [102].

the interfaces of the board and standardized interfaces for the communication between the functional blocks within the FPGA. This facilitates the testing and commissioning of new functions.

At the current state of the development, it is expected that one processor node can be realized by at most five MP7 boards. To cope with the high event repetition rate, each processor node is duplicated 36 times for time multiplexing. For each of the eight sectors, one of these systems exist. A total of 1440 MP7 boards is necessary, which are mounted in at least 120 MTCA crates with twelve slots.

4.3.3. Tracklet approach

The tracklet approach implements the track finding on the base of tracklets [106]. A tracklet is a fraction of a track that is generated by combining two stubs on adjacent layers of the CMS outer tracker. For the tracklet approach, the silicon tracker is divided into 28 sectors in the $r\varphi$-plane. The size is chosen, so that a track of a 2 GeV/c particle is always contained in at most two sectors.

Consequently, a sector contains parts of the barrel and the endcaps of the silicon tracker. Figure 4.9 shows the data flow of the CMS track trigger based on the tracklet approach. This algorithm is implementable in commercial FPGAs and can completely be pipelined.

The data processing of the tracklet approach already starts in the DTCs where the local coordinates are transformed into global coordinates, i.e. from a layer, module number, strip number representation to physical coordinates (r, ϕ, η). In the AM approach, this coordinate transformation takes place much later before the track fitter. The first step of the track trigger itself is the generation of seeds, i.e. the tracklets. These seeds are generated from stubs on adjacent layers. To generate the seeds, pairs of stubs from layers 5 and 6, layers 7 and 8, and layers 9 and 10 within the same sector are built. The biggest challenge of the tracklet approach is the large number of seeds that can be built of all the stubs within one sector. Only from the two innermost layers, 3600 seeds can be generated for an average event [106]. The angle that a sector spans is defined on the base of a 2 GeV/c particle passing through sensors on the inner- and outermost layers. If the difference in φ of the same particle passing through two adjacent layers is taken, the angle that is necessary to cover the track of a 2 GeV/c particle is smaller than it is chosen for the sector. Therefore, not all combinations of stubs on two adjacent layers can lead to valid (> 2 GeV/c) tracks. By taking this into account, the number of seeds can be reduced significantly. This works similarly for the track candidate builder of the AM approach, which are illustrated in Figure 5.8 in Section 5.2.4.

In the next step, the generated seeds are projected to the other layers of the tracker, both inside-out and outside-in depending on the layer pair to which

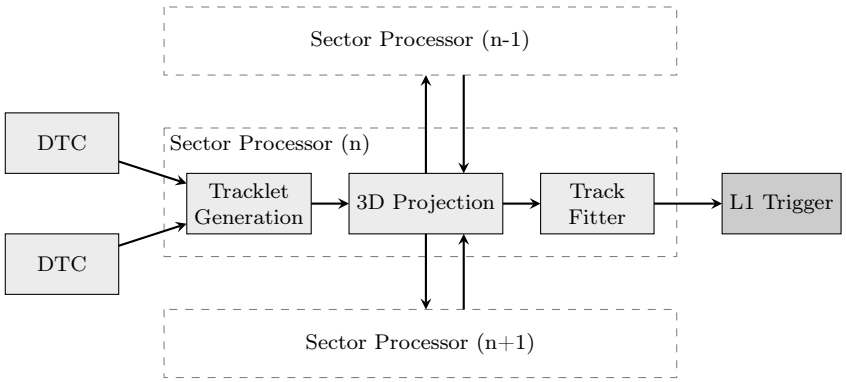


Figure 4.9.: Data flow of the tracklet algorithm.

the seed belongs. For the seed projection, the track direction indicated by the tracklet is extended until it hits another layer. The position on that layer is the projection. Within a predefined window around this point, stubs are searched on this layer. If a stub is found, it is added to the track candidate. If multiple stubs exist within the window, the one with the smallest distance is selected. A projection may be within a neighboring sector; in this case, the projection coordinates are sent to the sector processor of the neighboring sector. The neighboring sector then executes the stub selection and returns the stubs to the processor on which the seed has been generated. The projection of the seeds can be made in parallel for every single seed. In contrast to the other approaches, the tracklet approach uses the full resolution for track finding.

After the stubs of a track are bundled, they are transferred to the track fitting stage. As in the other approaches, the track fitting determines the exact parameters of a track. The tracklet approach uses a linearized χ^2-fit. Although the name is similar as for the fitting method used in the AM approach, a different algorithm is applied. The fitting can fit all tracks for which at least four stubs were found. Four (p_T, η, φ_0 and z_0) track parameters and a quality factor (χ^2) are calculated and transmitted to the L1 trigger. Most of the operations needed for the track fitting are well implementable in an FPGA, but for some more resource-hungry operations, such as arcsin(), the results are precalculated and stored in Lookup Tables (LUTs).

The final step in the processing chain is the duplicate removal. Duplicate removal is necessary because multiple seeds from different layer pairs can lead to the same track. This results in a high track finding efficiency, as every possible track is also found in case of missing hits, but duplicate tracks are generated. To find the duplicates, the duplicate removal stage compares pairs of tracks, first within a sector and then also tracks across the sector boundaries. A duplicate is defined as a track with less than three independent stubs. Only one of these duplicates is kept.

The current development is based on the CTP7 board, an MTCA board with two FPGAs developed by the University of Wisconsin for the 2016 L1 trigger upgrade [107]. For the final system, an ATCA system is targeted. One ATCA board is foreseen for the processing of one tracker sector. To process all the data, it is estimated that the system has to be replicated six times for time multiplexing. Therefore, a total of 168 (6×28) ATCA blades will be needed for the whole CMS track trigger, which corresponds to 18 crates. The communication between neighboring towers runs over optical fibers, and the backplane is used

for the slow control and DAQ. In total, the communication runs over roughly 5000 fibers.

4.3.4. Comparison of concepts

Since the status of the work on the three different concepts is quite different, it is hard to compare them directly, and a detailed evaluation is not possible. Nevertheless, this section compiles important information from the previous sections in Table 4.1 and compare some aspects of the three concepts next.

The first point to compare is the division of the system that includes two elements: the number of sectors in which the detector is divided and the time-multiplexing factor. The product of these two values indicates the number of equal sector processors on which the CMS track trigger runs. Both the TMTT and the tracklet approach divide the detector only in sectors in the $r\varphi$-plane. The TMTT approach uses the largest possible sector size limited by the detector readout scheme. In contrast, the tracklet approach uses the smallest possible sector size that is given by the curvature of 2 GeV/c particles. Both approaches divide the sector further within the sector processor to reduce the size of the data set that needs to be processed by the individual processing blocks. The processing of the smaller data sets takes less time, and the latency is minimized by parallelizing the processing. The AM approach is the only approach that splits the detector into sectors also along η. Therefore, three different types

Table 4.1.: Properties of the different CMS track trigger concepts. Status of May 2016. Sources: [88, 93, 98–103, 106]

Property	AM Approach	TMTT Approach	Tracklet Approach
Hardware Sectors	$8 \times 6 = 48$	8	18
Time-Multiplexing Factor	20	36	6
Sector Processors	960	288	168
Data Sharing btwn Sectors	no	no	yes
Track Finding Method	AM + 3D Projection	Hough transform + rz-Filter	Tracklet + 3D Projection
Track Fitting Method	PCA-based Linearized Fit	Linearized χ^2 Fit / Kalman Filter	Linearized χ^2 Fit
Processing Devices	FPGA and AM ASIC	FPGA	FPGA
Hardware Architecture	ATCA	MTCA	ATCA
Crates	48	120	18
Processing Board Format	Double-wide FMC	MTCA Board	ATCA Board
Processing Boards	960	1440	168

of sectors (barrel, hybrid, endcap) exists in the AM approach, i.e. three different sector processors have to be developed, which may differ slightly in their hardware architecture.

The sector size has a direct effect on the data that has to be transferred to a sector processor for a single collision event: smaller sectors generate less data, but more data need to be transferred to multiple sector processors. To keep the input bandwidth small despite large sectors, the time-multiplexing factor may be increased. This can be seen clearly at the TMTT approach that has small sectors but a high time-multiplexing factor. Another way to reduce the input bandwidth is to share the sector overlap data directly between the sectors. The AM and TMTT approaches duplicate the data of the overlap regions at the DTCs. Therefore, all stubs that belong to an overlap region are duplicated. In contrast, the tracklet approach does not exchange data until the projection stage where only data from tracks crossing the boundaries have to be transferred. Another advantage of data sharing between the sector processors is that the data may be transferred over the backplane of a crate. The disadvantage of the data sharing between sector processors is an increase in latency that is caused by the additional communication step.

Although the algorithms for the data processing stages are completely different among the three concepts, two basic parts—track finding and track fitting—exist within all of them. Therefore, some blocks of the data processing chains could be adapted and integrated into any of the three concepts. Beside the track fitting block that is basically exchangeable between all three approaches, also elements of the seed generation and 3D projection can be exchanged between the AM and tracklet approaches.

The hardware implementation of the three CMS track trigger concepts differs widely. Due to the challenging requirements of the CMS track trigger, all three concepts use FPGAs for the data processing and boards with extreme data transmission capabilities. Therefore, the ATCA standard seems to be a good choice, as there is more space for optical connectors and full-mesh backplanes are available for the communication within the crate. Nevertheless, the TMTT approach builds upon an MTCA system whose boards are much smaller than the ATCA boards.

The smaller size of the MTCA boards may also be a reason for the high number of processing boards for the TMTT approach. Another reason is the conservative assumption that five boards are necessary for one sector processor. One board is foreseen for these processing stages that are not implemented yet, the rz-filter and the track fitter. With the other two approaches in mind, one board

will most likely be enough to execute both of them. Furthermore, one instead of two boards for the Hough transform may be sufficient. With the increasing number of logic elements in the FPGAs, the number of boards may also be reduced. The number of boards for the tracklet approach is so low because the number given is the extrapolation to the final system, i.e. the technical progress is included in the calculation.

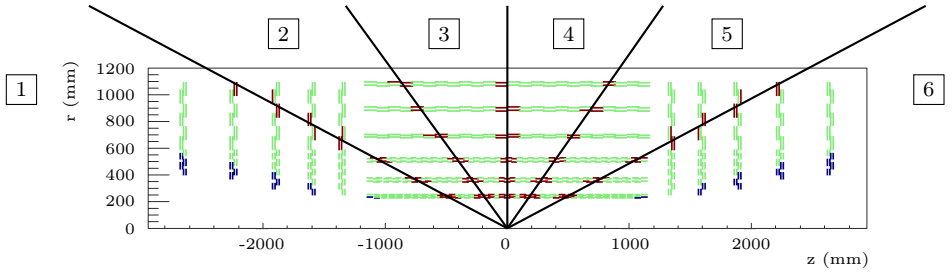
5. A CMS Track Trigger Concept Based on Associative Memories

After the introduction of the CMS track triggerin general and the three different approaches in the last chapter, the AM approach is explained here in detail. The AM approach of the CMS track trigger builds on the AM Chip designed for the CDF SVT and ATLAS FTK. Besides this central component, also other concepts and a hardware board were adopted from the FTK design.

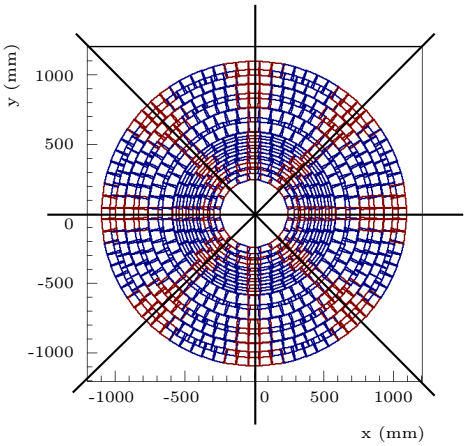
5.1. Data distribution

The amount of data produced by the CMS silicon tracker is enormous. In average around 10 000 stubs and for some events up to 20 000 stubs are sent to the track trigger each bunch crossing. It is just not feasible to process this amount of data on a single hardware board with the required latency. Therefore, the detector is divided into independent geometrical sectors, and the data from each sector is processed independently. This is possible since the CMS track trigger only delivers data from particle tracks with a transverse momentum above 2 GeV/c. These tracks are not bent much by the magnetic field and, thus, each track stays within a narrow part of the detector.

In the approach presented here, the detector is divided into 48 sectors—also called *trigger towers*. Figure 5.1 shows the division of the outer tracker into six sections in η and eight sections in φ. There are three types of trigger towers depending on their location in η: barrel, hybrid and endcap. The barrel towers (3 and 4 in Figure 5.1) contain only detector modules of the barrel section of the detector. Correspondingly, the endcap towers (1 and 6) contain mainly modules belonging to the endcap and the hybrid towers (2 and 5) include modules from both—the barrel and endcap sections. The eight pieces into which the sectors are divided in the $r\varphi$-plane are all equal. A few modules at the sector boundaries are shared between adjacent sectors. This overlap ensures that every high momentum track is completely contained in at least one trigger tower.



(a)



(b)

Figure 5.1.: Division of the tracker into six sectors in the rz-plane (a) and eight sectors in the $r\varphi$-plane (b). The red modules at the sector boundaries are shared between two trigger towers. Numbers 1 and 6 are endcap sectors, 2 and 5 hybrid sectors, 3 and 4 barrel sectors.

Due to the high data rates, not only the processing but also the distribution of the data to the final processing boards is challenging. Figure 5.2 shows the data distribution schematically from the detector modules to the processing boards.

The first board on which the data arrive after leaving the detector is the "Data, Trigger and Control Board" (DTC). Currently, neither the detailed functionality of this board nor its hardware implementation is defined for the AM approach. However, three tasks can be identified that define the minimum functionality of a DTC: (1) The DTC splits the data stream into trigger and raw data and sends the latter to the DAQ system of CMS. (2) The DTC redirects the trigger

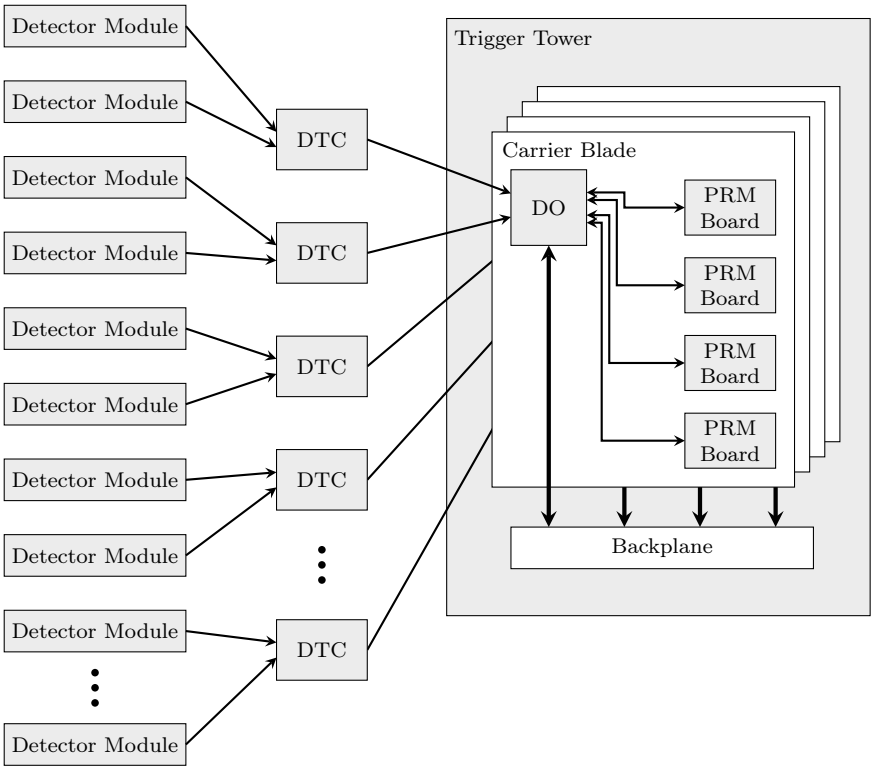


Figure 5.2.: Data flow from the detector modules to the processing boards (PRM) of the CMS track trigger.

data from the modules to the crate that processes the corresponding trigger tower. The data from modules at sector boundaries are duplicated and sent to all sectors to which they belong. (3) As each module is connected by an individual link to the DTCs, only the module internal coordinates are stored with the stubs. Therefore, the module coordinates must be added to the stubs at the DTC. Furthermore, the DTC packs all stubs belonging to an eight clock cycle window, as given by the CIC on the detector modules (Section 3.5.1), into one data frame.

Unfortunately, the optical links from the silicon detector cannot be arranged according to the trigger towers. The optical fibers are already bundled into ribbons and cables at the detector. The fibers that belong to a specific cable are given to some extent by the mechanical construction of the silicon tracker. As a consequence, a cable may contain fibers that belong to modules that from different trigger towers. However, one cable of fibers can only be connected

to one DTC. Therefore, a DTC may send data to multiple crates, and more output links are necessary as if a DTC would process data from only one trigger tower.

All the data of one trigger tower are processed in at most one crate. Currently, it is assumed that even two trigger towers can be processed in one crate. If the technical progress makes it possible, four trigger towers may be processed in one crate. Carrier boards are installed within the crates that host the actual processing boards. Again, depending on the technological progress in FPGA technology two or four processing boards may be mounted on one carrier board. One processing board performs all data processing of one specific event within one sector. The events are assigned to the different processing boards within a crate by round-robin time multiplexing.

The distribution of the data to the processing boards within a single crate is rather complex. Due to the high input bandwidth at each crate of up to 1 Tbit/s, and as the data are transmitted from multiple DTCs, the optical links are connected to all the carrier boards of a crate. The data organizer implemented on the FPGA of the carrier board performs the data distribution within the crate. In a first step, the DO collects one frame of stubs arriving at all the optical inputs of the carrier board. As the stubs in one received frame belong to different events, the stubs are sorted by storing the stubs of each event to a different buffer. After processing the complete input frame, the stubs in each buffer are transferred to the carrier board that hosts the processing board to which the event stored in the buffer is assigned. This transfer takes place over the backplane of the crate.

The DO on the receiving carrier board collects the stubs from all the other carrier boards of the crate. Towards the processing boards, the stubs are transferred on separate links for each detector layer. Thus, the DO determines the layer of the stubs and stores them in a buffer dedicated to the layer. From these buffers, the stubs are transferred to the processing boards for the actual track finding and fitting.

The calculated track parameters from the processing boards are transferred back to the DO of the carrier board. The DO transfers the results to dedicated gateway boards installed within the same crate. These gateway boards implement the communication to the L1 trigger.

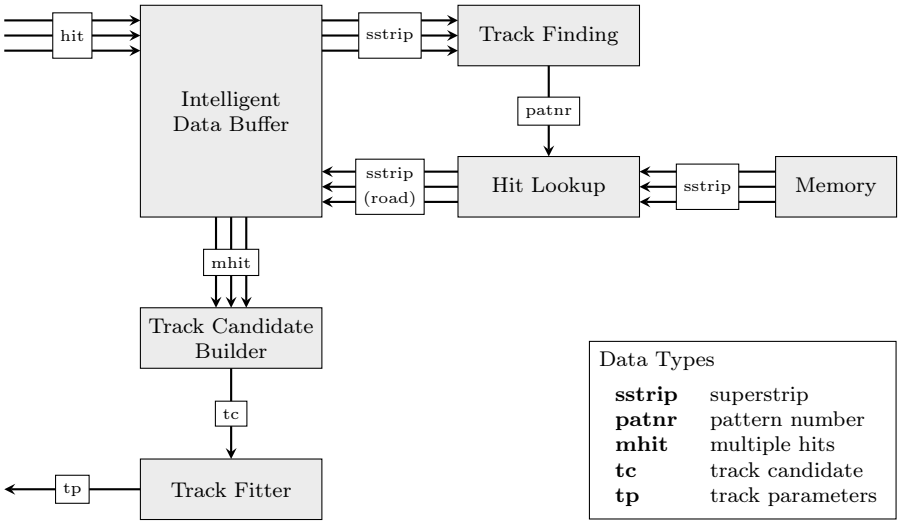


Figure 5.3.: Data flow on the PRM board of the CMS track trigger.

5.2. Algorithm of the AM-based CMS track trigger

The first part of this section provides an overview of the data flow of the actual processing that takes place on the processing board. The following Subsections 5.2.1–5.2.5 describe the individual elements in detail. The data flow on the PRM board including the data formats is shown in Figure 5.3. The description of the algorithms is based on [71]. To keep the explanations simple, only the processing of a trigger tower in the barrel section is described here. Generally, the endcap and hybrid trigger towers are processed in the same way, only some details and numbers have to be adjusted.

The input data set arriving at the processing board consists of all hits of one collision event that belong to one trigger tower. The hits arrive ordered by detector layer on separate data streams, i.e. six streams for the six layers in the barrel section. For most of the processing steps, these data streams remain separated.

The track finding cannot handle the full-resolution hits (19 bit for 2S modules and 23 bit for PS modules). Therefore, the resolution is reduced, and so-called *superstrips* with lower resolution (16 bit) are generated by the *Intelligent Data Buffer (IDB)*, more in Section 5.2.1. As the final processing step, the track fitting needs the full-resolution hits; they have to be buffered. The main task of

the IDB is to buffer the full-resolution hits indexed by the superstrip for later retrieval.

The central and name-giving part of the "Associative Memory approach" is the *track finding* by the AM Chips (see Section 5.2.2). Pre-calculated patterns that represent interesting particle tracks are stored within the AM Chips. All the superstrips that belong to one event are fed to the AM Chips. Each superstrip is compared with a part of each pattern that represents the same layer as the superstrip belongs to. If the superstrip and the part of the pattern are equal, this match is stored. After processing all superstrip, the AM Chip searches for patterns for which a majority of the layers were matched.

As the IDB needs the superstrips to look up the hits, the superstrips of the patterns are recovered from a memory. This memory stores all the superstrips that belong to a certain pattern. The *hit lookup* block (see Section 5.2.3) is the logic that realizes this memory lookup. The result is a set that consists of one superstrip on each detector layer, which is also called a road. A road can be seen as a coarse resolution corridor through the silicon detector within which multiple tracks can exist. However, only a small number of hits are usually part of a road, and possible tracks can be found without large combinatorics.

After the recovery of the hits by the IDB, multiple hits may exist on a single layer. The *track candidate builder*, described in detail in Section 5.2.4, selects the best combination of hits so that at maximum one hit exists on each layer. These six hits are called a track candidate. The track candidate is transferred to the final block—the *track fitter*, described in detail in Section 5.2.5. The track fitter applies a linearized fit to the hits, and the calculated track parameters are then sent to the L1 trigger.

The track trigger algorithm does not work perfectly. Two examples of imperfections shall be mentioned here. The same track can be found multiple times because the AM Chips do not work with full-resolution tracks. Although this is not implemented yet, duplicate removal is possible at different stages of the processing. The other issues are related to fake tracks. These are tracks that do not correspond to a particle track but were found due to imperfections of the algorithm and the presence of additional hits (noise). Fake tracks cannot be removed as they cannot be distinguished from real tracks. However, the algorithms can be optimized to reduce the number of fake tracks appearing in the output data of the CMS track trigger.

5.2.1. Intelligent data buffer

The IDB is the central controller of the data flow between the data input, the track finder and the track fitter. It takes the data streams, transforms the data formats and forwards them to the next block. Figure 5.4 shows the conceptual working principle of the IDB for one layer. This processing chain exists for every detector layer, and each layer works mostly independent from the others.

As already described in the previous section, the resolution of the hits from the detector must be reduced for the track finding. This is achieved by combining adjacent strips on a module to so-called superstrips, as shown in Figure 5.5. Currently, the size of the superstrip addresses is limited by the design of the AM Chip to 16 bit. Two solutions have been proposed to implement this resolution reduction: Firstly, as the last bits of the incoming hits represent the strip address within the module, these least significant bits can simply be truncated. Secondly, several regions within the data format of a hit represent the same detector coordinate, e.g. the z-coordinate is defined by the z-position of the module, the segment of the detector module and possibly the pixel number. As the individual components are represented by a fixed number of bits whose ranges are usually not fully used, few bits can be saved by re-combining all components into one coordinate to achieve an optimal representation. The superstrip generation is then applied to these new coordinates. For the following steps, it does not matter how the superstrips are generated as long the patterns for the AM Chip are generated accordingly.

Beside the generation of the superstrips, the IDB is also responsible for the recovery of the full-resolution hits from the superstrips after the track finding. To do so, the IDB includes a buffer in which all the incoming hits are stored. This buffer is organized in a way, that the hits can be found according to the superstrip to which they belong. When superstrips are returned from the track finding, the hits must be addressable by the superstrip to which they belong. The buffer may be implemented as an AM that is tailored for this purpose. As multiple hits may occur within a single superstrip, the buffer must be able to store a list of hits for each superstrip. A superstrip may belong to more than one track and, therefore, it must be possible to read out the same hits multiple times.

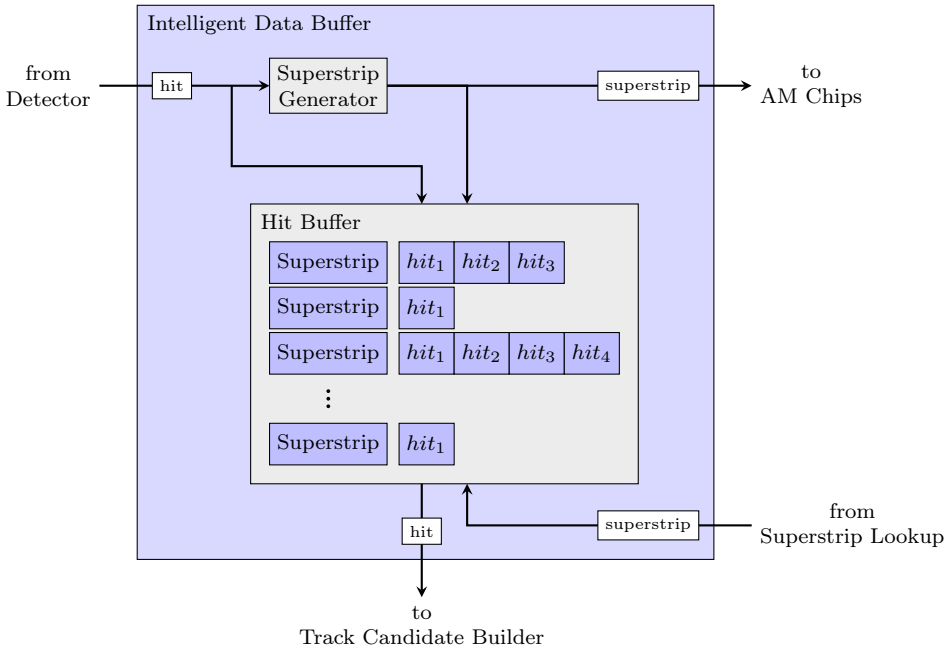


Figure 5.4.: Working principle of the intelligent data buffer for one of the detector layers.

5.2.2. Track finding by associative memories

The purpose of the track finding is to identify subsets of hits (roads)—within all the hits of one sector belonging to a single event—which may form a track candidate. The hits arrive on separate data streams for each detector layer, but within a layer, they arrive unordered. The number of hits that arrive on each layer for one event is up to 70. Out of all these hits, of the order of 118×10^9 combinations can be built. It is just impossible to check all these combinations within a short amount of time.

To address this issue, an ASIC called AM Chip has been developed [11, 108] that is specialized in finding predefined tracks in streams of hits. At the core of the AM Chip, there is an AM that stores patterns for all conceivable tracks that belong to high-momentum particles ($>2\,\mathrm{GeV/c}$) and compares the incoming hits with these tracks.

When a high-momentum particle crosses the detector, it crosses each layer once and generates a hit in a detector module on each layer. Therefore, a particle

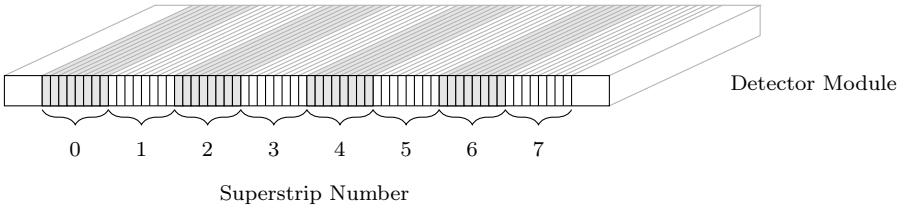


Figure 5.5.: Superstrips are the combination of adjacent strips into strips with coarser resolution.

track in the barrel section is defined by six hits, as shown in Figure 5.6. The coordinates of these six hits build together a pattern, which is the basic storage element within the AM Chip. The set of patterns that cover one trigger tower is called a pattern bank. To keep the number of patterns low, not the full-resolution hits but the superstrips are used in the patterns. Each pattern bank consists of one to two million patterns. Up to 100 million tracks are stored in the AM Chips to cover the complete detector.

The patterns are generated prior to the operation of the track trigger within CMSSW—a simulation environment for the detectors and the data processing of CMS [12]. For this, Monte Carlo simulations of proton collisions are applied to a virtual detector to generate particle tracks [71]. The tracks are then analyzed, and patterns are generated from the high-momentum tracks. These pattern banks are finally loaded to the AM Chips on the processing boards of the CMS track trigger.

During operation, the incoming superstrips are compared with the stored patterns. When a pattern is found, its number is returned. Figure 5.7 illustrates the track finding process. At the top, a pattern bank with four patterns is shown. The hits from the detector are now compared with this pattern bank, and the found hits are marked for each pattern (middle). Finally, the hits that belong to the found tracks are output as packets (bottom). The track finding ideally fulfills three requirements:

1. All tracks of particles with a transverse momentum above a certain threshold, currently 2 GeV/c, are found.
2. All hits that belong to a specific track are gathered in one packet.
3. Hits that do not belong to any track are removed from the data.

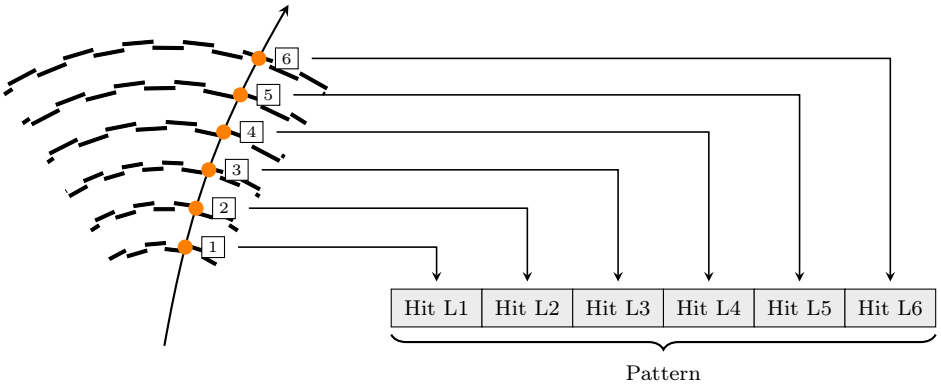


Figure 5.6.: The coordinates of all the hits that belong to a track that shall be found build together a pattern. Such patterns are stored in the AM Chip.

Algorithm 5.1 describes in pseudocode the track finding by the AM Chip in more detail. Generally, it consists of two parts (divided by the dashed line) that are executed after each other. The first part reads all the hits of each layer from the corresponding input. All the hits of a layer are then compared with the section of each pattern that belongs to that layer. If the hit is equal to the pattern section, a flag is set that marks the occurrence of the pattern section in the hit stream. In the second part, the patterns with all flags checked are identified. For the identified patterns, the pattern number is sent to the next processing step. It would be beneficial if the hits and not the pattern number could be put out, but this feature is not supported by the current AM Chip.

Both the first and the second part can widely be parallelized in hardware. In the first part, all the streams from the layers are read in parallel, and the comparison of the hits with all the patterns happens concurrently. Also, the counting of the marked hits is implemented in parallel. The pattern numbers are then written to the output one after each other. Therefore, the latency depends mainly on the number of input hits, the number of found patterns and the I/O speed with which they are read, respectively written.

So far, these are the basic functionalities of the track finding. Some additional features are part of the AM Chip to optimize the track finding. The most important additional feature is a programmable threshold that defines how many hits must be activated within a pattern that it is found. For example, it is possible to find tracks of which only five hits on six layers were detected. This may happen for several reasons: a broken detector module, the signal caused by the particle was too small in one module, the hit was discarded by

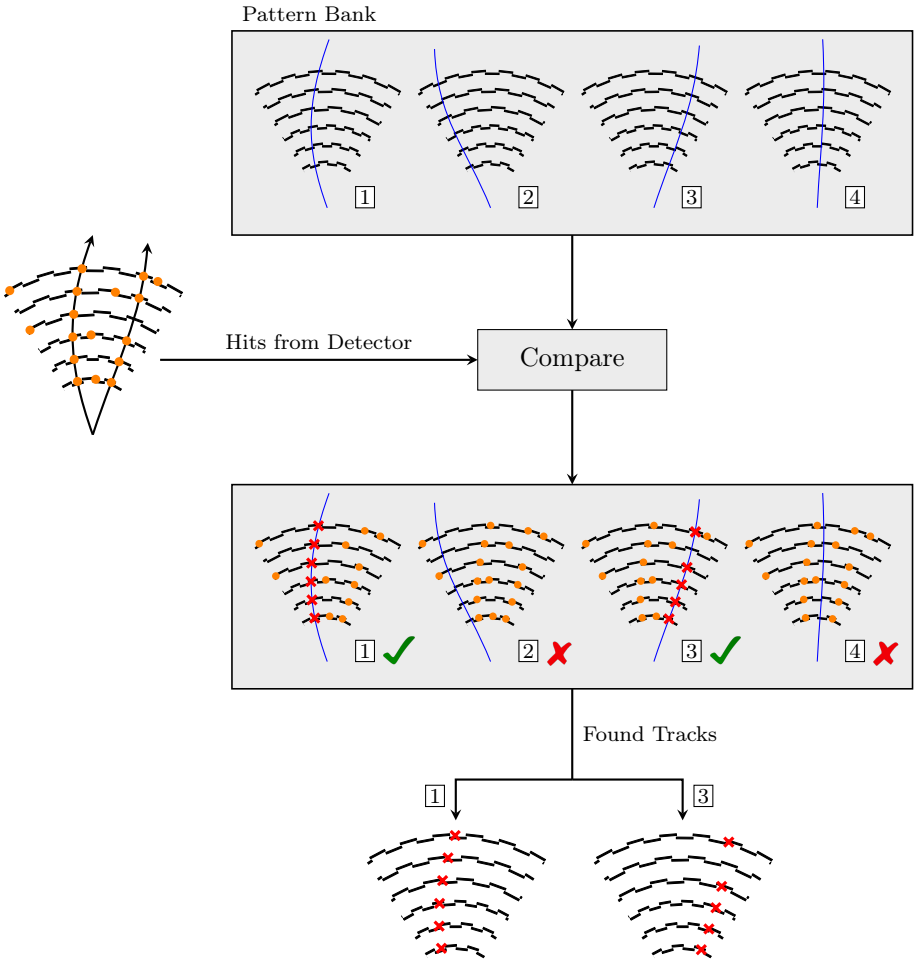


Figure 5.7.: The principle of track finding in the AM Chip. The incoming hits are compared with the patterns stored in the pattern bank. All the tracks with five or more hits activated in the corresponding pattern (1 and 3) are returned by the AM Chip.

Algorithm 5.1 Track finding in the AM Chip

```
for layer ← 1 to layer_count do
    while hits available for event do
        hit ← input[layer]
        for pattern ← 1 to pattern_count do
            if patternbank[pattern, layer] = hit then
                hit_flag[pattern, layer] ← true
            end if
        end for
    end while
end for
— — — — — — — — — —
for pattern_nr ← 1 to pattern_count do
    hit_counter ← count_true_elements(hit_flag[pattern_nr])
    if hit_counter = layer_count then
        output ← pattern_nr
    end if
end for
```

the concentrator on the module, etc. This feature highly improves the track finding efficiency.

Another feature helps to reduce the size of the pattern banks. Two patterns often differ only by few bits, because the tracks that they cover just cross the superstrip boundary. By using ternary logic for the last few bits of the superstrip representation in the patterns, it is possible to merge patterns that differ only in those Don't Care (DC) bits into one pattern [11, 109]. Ternary logic is used to represent the value "DC". A DC means that this bit can be either one or zero and, thus, two different superstrips can be addressed by a single pattern. To keep the number of changing bits between to adjacent strips of the detector low, the hit addresses in the patterns are stored in Gray coding.

5.2.3. Superstrip lookup for pattern

As described in the previous section, the AM Chip returns just the number of a found pattern. However, the individual hits belonging to a pattern are necessary to fit a track. Before the IDB can recover the hits in the next step, the superstrips belonging to a found pattern must be recovered.

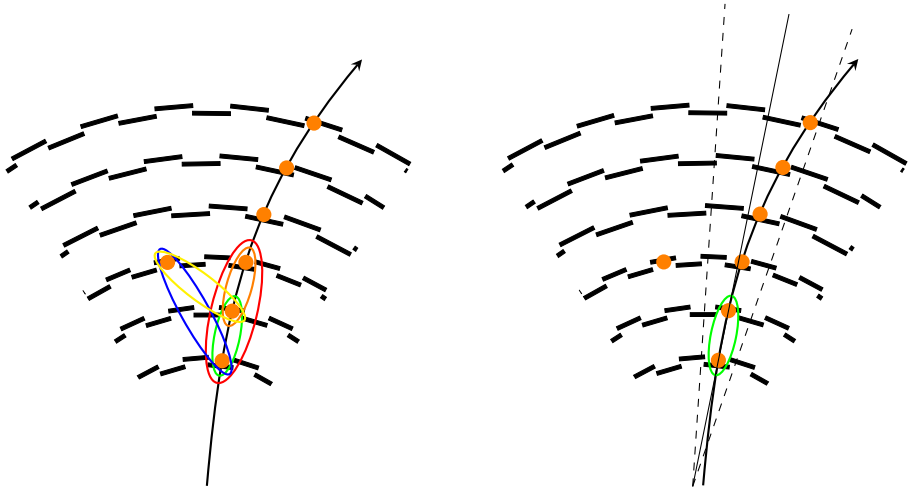
This lookup is basically the reverse function of the track finding and is implemented with a memory. This memory contains the same pattern bank as used for the track finding with the reverse association—the superstrip is derived from the pattern number. To find the superstrips of a pattern, simply the memory address where the data of that pattern is stored is applied to the memory. For a pattern that has missing superstrips, the lookup returns all superstrips that may be part of the pattern. The superstrips that were added by the lookup must be removed at a later stage. Likewise, for patterns containing DC bits, all possible superstrips that may be represented by the pattern must be resolved. Therefore, it is possible that a packet representing a found track contains more than one superstrip on a layer.

5.2.4. Track candidate builder

After the recovery of the full-resolution hits from the IDB, it is possible that a road contains multiple hits per layer. However, the chosen track fitting method (Section 5.2.5) requires clean sets of hits to work properly. This means that only one hit per layer may exist in the set and at most one hit is missing. The goal of the track candidate builder is to select such a combination of the provided hits, which is called a track candidate. The track candidate builder selects only the hit combination that will give the best result in the track fitter [71]. The algorithm of the track candidate builder is rather complex, and it does not have a fixed latency. It can be split into these three basic steps:

1. generate seeds from hits on the three innermost layers,
2. project the seeds to the outer layer and select the hits with the closest distance to the projected track, and
3. select the best track candidate.

In the first step, all possible combinations of two hits from the PS modules are generated. Such a pair of hits builds a vector that is called a seed. As all three layers with PS modules are used, a seed can be created even if a hit of a track is missing on one layer. The hits from the PS modules are used, as they provide better resolution in the z-direction and the projection becomes more accurate. Figure 5.8 (a) shows the hits belonging to a found track and the five seeds that can be generated from the hits on the three innermost layers.



(a) Seed generation (b) Projection of one seed (simplified)

Figure 5.8.: The first two processing steps of the track candidate builder.

In the next step, every generated seed needs to be projected in 3D to all the layers that are not part of the seed. To simplify the implementation in FPGAs, instead of a full 3D projection, the seeds are projected two times in 2D: once in the $r\varphi$-plane and once on the rz-plane. Furthermore, the $r\varphi$-plane projection is linearized to avoid operations that are executed slowly in FPGAs. Figure 5.8 (b) shows the projection of one seed in the $r\varphi$-plane. Acceptance windows are defined in the $r\varphi$- and the rz-plane, of which the $r\varphi$-window must be adapted according to the linearization of the $r\varphi$-projection. For each hit that lies on one of the layers to which the seed is projected, the distances to the projected position in the $r\varphi$- and the rz-plane are calculated and compared with the acceptance windows. If both distances lie within the windows, the hit is accepted for the track candidate; otherwise, it is rejected. If more than one hit is located within the limits, the one that is closest to the projection in the $r\varphi$-plane is selected.

In the last step of the track candidate builder, the track candidates generated from the different seeds are compared, and only the one with the smallest distances from the projection is selected and sent to the track fitter. Track candidates with more than one missing hit are immediately rejected. If there are still multiple track candidates, the ones without missing hits are prioritized. If several track candidates with one missing hit exist, the one with a complete set of hits in the PS modules is selected [71].

5.2.5. Track fitter using a linearized fit

In the final step of the CMS track trigger processing, a track is fit to the track candidates provided by the track candidate builder. The goal of the track fit is to calculate the track parameters, described in Section 3.3, as accurately as possible. The track fitter of the CMS track trigger ignores the distance from the beam axis d_0, thus four track parameter remain: the location on the beam axis z_0, the angles φ_0 and η, and the curvature R. The applied fitting method is *linearized track fitting*, which is well suited for an implementation in FPGAs in terms of resources and latency. Each track parameter p_i is calculated by the formula

$$p_i = \sum_{l=1}^{N} A_{i,l} x_l + q_i \tag{5.1}$$

where N is the number of hit coordinates in the set and x_l is the coordinate of the hit on layer l. The hit coordinate is multiplied by a constant $A_{i,l}$ and then another constant q_i is added. The constants A are precalculated for a specific fitting problem.

This off-line calculation is exactly what makes the linearized fit so powerful for hardware implementations. Complicated calculations such as trigonometric functions do not need to be calculated on the hardware. Nevertheless, the linearized fit can be applied to many kinds of problems, i.e. the same fitting algorithm is used to calculate both the curvature of the track and the location on the beam axis. A track candidate of the CMS track trigger consists of twelve coordinates: z and φ for every hit. Therefore, a set of twelve constants A and one offset q exist for the calculation of a track parameter. For every fit parameter, an independent set of constants has to be calculated. As the name of the method implies, the fitting problem is linearized in the constant calculation process. Thus, the result of the fit is indeed accurate around the point at which the linearization has been performed but is less accurate in other areas of a trigger tower. To achieve good fitting results over the entire trigger tower, the trigger tower is split into smaller sections, and a different set of constants is calculated for each section. The set of constants is chosen by a pre-estimation of the parameter by which the section is defined. For example, the trigger tower is split into five sections in φ, then five sets of constants are calculated one for each φ section. The set can simply be selected by the φ coordinate of the hit from the outermost layer, as this gives a rough estimate of

the φ_0 of the track. For the CMS track trigger, the constants of the linearized track fit are determined by a method called PCA [95]. With the PCA, a set observations can be examined and correlated variables can be converted into a set of linearly uncorrelated variables. Thereby, the dimensionality of a data set may be reduced. In case of the CMS track trigger, the observations are the properties of tracks simulated in CMSSW.

One problem of the linearized track fitting is the dependence on a complete set of hits. From Equation 5.1, it is obvious that a missing coordinate would falsify the result of the summation. However, the track candidate builder allows one hit to be missing. For this reason, the track fitter contains multiple sets of constants: one set for the case if all hits are available, and a set for each case in which a hit is missing on one layer. Thus, seven sets of constants are necessary for trigger towers in the barrel section that have six layers.

Together with the track parameters, an error χ^2 is calculated that provides information about the quality of the fit. A significant advantage of the linearized track fit is that errors are calculated similarly as the fit but with an additional set of constants. The error is used to determine if a track is a valid track or the hits of the track candidate belong to different tracks.

5.3. Hardware platform

Besides the algorithm, specific hardware for the CMS track trigger is being developed by the groups belonging to the AM approach. The here described hardware components are developed for the review in December 2016. The goal of this system is not primarily to achieve the full performance needed by the CMS track trigger but to demonstrate that a CMS track trigger based on AM Chips and FPGAs is feasible for 2025.

5.3.1. Crate system

A system based on the ATCA standard is proposed for the hardware of the AM approach CMS track trigger. The ATCA standard originates from the telecommunication industry [110, 111]. The standard has been defined with high data throughput in mind. Therefore, ATCA also suits well the high requirements for communication bandwidth of the CMS track trigger. Two factors enable this high data throughput: on the one hand the large size of the system components and on the other hand the support of full-mesh backplanes.

The crates, as shown in Figure 5.9, of the ATCA standard are relatively large. They can host large Printed Circuit Boards (PCBs)—called blades in the ATCA standard—which are 280 mm deep and 322 mm high. A crate that fits into a 19-inch rack can host up to 14 blades. The blades are often used as carrier boards, and smaller boards are plugged onto the blades. The long front of the blades makes it possible to place many optical or electrical connectors. Additionally, a so-called Rear Transition Module (RTM) may be plugged to each blade in the back of the crate. The RTM is foreseen for the communication towards the back side of the crate. This allows the cables to stay connected to the crate when a blade is exchanged within the crate. The RTM may also be used to increase the bandwidth as connectors can be placed on both sides of the crate.

A backplane connects the blades with each other within the crate. Different network topologies exist for the backplanes [111]. The backplanes used for the AM approach CMS track trigger has a full-mesh topology [112], which is illustrated in Figure 5.10. On a full-mesh backplane, each blade is connected to each other blade by designated communication lines. The advantages of the full-mesh backplane are maximum throughput and low latency. The maximum throughput is achieved by the fact that blades do not share the lines, and each blade can send data constantly with maximum speed to any other blade constantly No switch is necessary as every blade can directly send data to all the other blades. Thus, it is also possible to communicate by a protocol with very little overhead, which increases the throughput further. Currently, a backplane capable of communications with up to 10 Gbit/s is installed in the test crates of the AM approach. In the future, backplanes with 40 Gbit/s per link may be installed, and backplanes with 100 Gbit/s have already been announced [113].

5.3.2. Carrier blade: Pulsar IIb

The Pulsar IIb is the proposed ATCA blade for the CMS track trigger. It serves as the carrier for the PRMs and is responsible for the data distribution both between the carrier blades and to the PRMs. It is a further development of the Pulsar IIa [115, 116] and was originally developed for the ATLAS FTK [117]. The target of the Pulsar II developments was a blade with very high communication capabilities. Figure 5.11 shows a photo of the Pulsar IIb blade.

The central element of the Pulsar IIb is a large Xilinx Virtex-7 FPGA. Different pin-compatible FPGAs with 410 to 690 thousand logic cells can be mounted on the Pulsar IIb. As the schematic overview of the Pulsar IIb in Figure 5.12

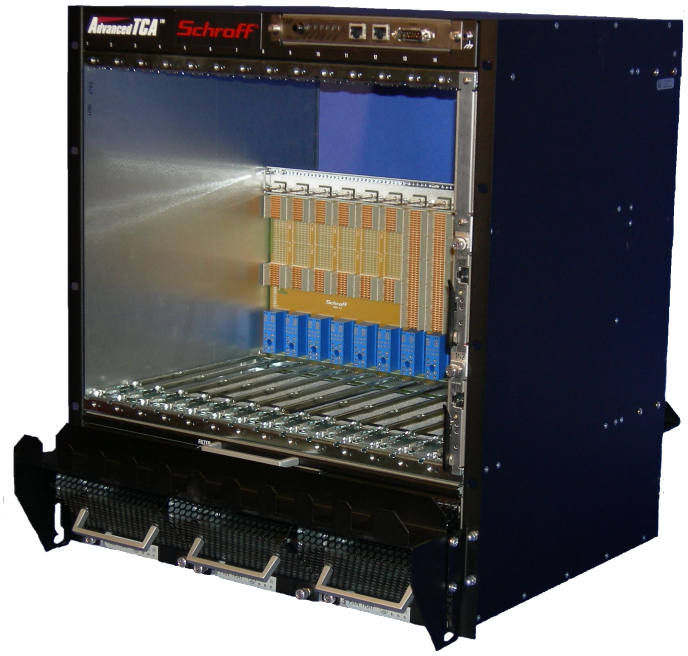


Figure 5.9.: 12-slot ATCA crate. Courtesy: Pentair/Schroff.

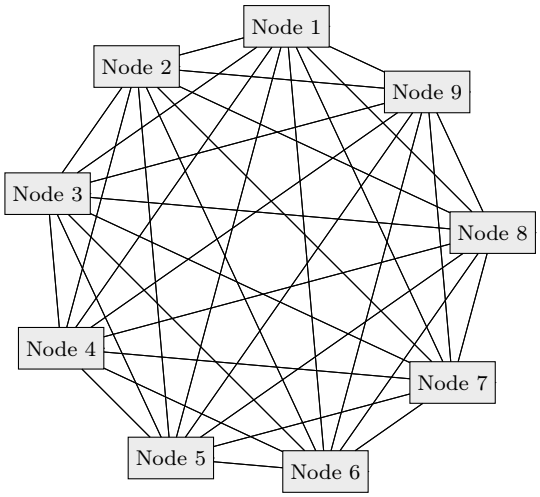


Figure 5.10.: A full-mesh network topology with nine nodes. Every node is connected to all the other nodes by designated links. Source: [114]

Figure 5.11.: The Pulsar IIb ATCA carrier blade with an old version of the RTM attached on the right side. Courtesy of Fermilab.

shows, all the data communication is connected to the FPGA. Additionally, a 256 MB DDR3 RAM is directly connected to the FPGA.

The Pulsar IIb can host up to four mezzanines that follow the FMC standard [118] and defines the connector type, the pin assignment, etc. It is also possible to plug a double-wide FMC to two adjacent FMC connectors. Each FMC connector provides three bidirectional High-Speed Serial Links (HSSLs) with up to 10 Gbit/s each that are connected to GTH transceivers of the FPGA. Additionally, 34 unidirectional Low-Voltage Differential Signaling (LVDS) signals at 1 Gbit/s are available at each FMC connector. Thus, the total bandwidth is up to 64 Gbit/s for a single FMC and up to 128 Gbit/s for a double-wide FMC.

Within the crate, the Pulsar IIb blades communicate over a full-mesh backplane with each other. The Pulsar IIb provides 28 GTH transceivers for the communication over the backplane. With 14 blades in one crate, two HSSLs connect each pair of blades. For the communication off the crate, an RTM has

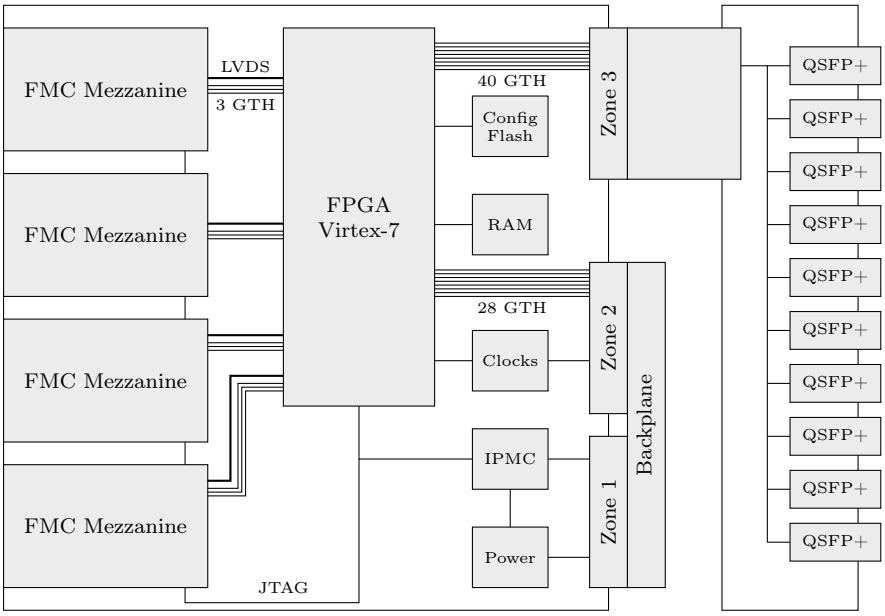


Figure 5.12.: Schematic of Pulsar IIb with the RTM (version 1.0). Source: [117].

been developed together with the Pulsar IIb, see Figure 5.13. This RTM is connected to 40 of the GTH receivers of the FPGA, providing a total bandwidth of 400 Gbit/s. To provide this bandwidth to the outside, 10 Quad Small Form-factor Pluggable (QSFP+) cages are mounted on the RTM. A QSFP+ cage is a mechanical device to which a QSFP+ transceiver can be plugged. The QSFP+ transceiver itself is joint with a fiber-optical cable or a copper cable and provides a bandwidth of up to 4×10 Gbit/s.

Besides these core functionalities, the Pulsar IIb also contains other components that provide different services. The Intelligent Platform Management Controller (IPMC) module implements the board control, monitors the temperatures, voltages and currents, and provides an Ethernet connection for the configuration of the FPGA. The FPGA configuration can also be stored in a flash memory on the Pulsar IIb. Other components provide power and clocks for the Pulsar IIb.

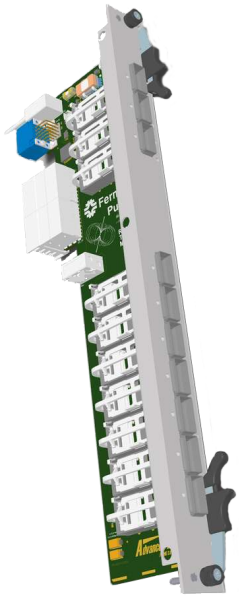


Figure 5.13.: Version 2.0 of the Pulsar IIb RTM. Courtesy of Fermilab.

5.4. Pattern recognition mezzanine

At the heart of the CMS track trigger, the PRM executes the CMS track trigger algorithms for the track finding and fitting. The PRM is a fully packed, complex board that hosts an FPGA and multiple AM Chips. It is developed at Istituto Nazionale di Fisica Nucleare (INFN) Perugia, and three of the PRM05 boards have been produced and successfully tested at KIT.

5.4.1. The associative memory chip

For the AM approach, the AM Chip is the central component for the track finding. The AM Chip is basically a CAM that is highly specialized for track finding in particle tracker applications.

The functionality of a CAM is to take a search word and compare it to a table of stored data words (patterns) [119]. This comparison is executed completely in parallel, which can be seen in Figure 5.14 where the search lines are connected to all the patterns. If the search word is found within the table, then the matchline of this pattern is activated. The encoder translates the matchline

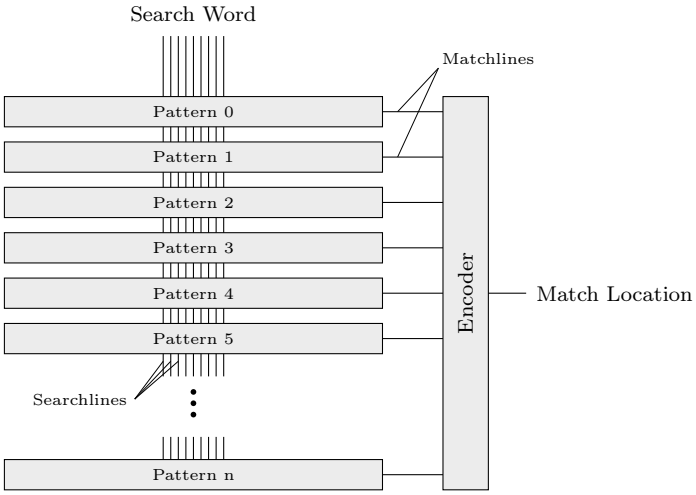


Figure 5.14.: Conecptual block diagram of a commercial Content-Addressable Memory (CAM). Source: [119]

to the storage location of the activated pattern. This storage location is then returned by the CAM. The advantage of CAMs over other search methods is that they are very fast and the result is available after one clock cycle in the best case. Commercial CAMs are mainly used in network routers for the routing and classification of network packets [120, 121]. Other applications that were presented are for instance: Hough transformation [122] and image coding [123].

The development of the AM Chip as a track finding device for multi-layer detectors at trigger level started in the late 1980s [108]. The first version of the AM Chip was presented in [125] and has been used in the SVT of CDF. The working principle of the AM Chip as shown in Figure 5.15 did not change much until today and is described in [108, 124, 125]. In the AM Chip, the stored patterns are split into multiple segments—one segment for each layer of the detector. Correspondingly, a separate hit input bus exists for each layer. A separate CAM unit exists for each pattern segment. The data at the hit input is then compared with the stored pattern segments by these CAMs. The comparison is performed concurrently for all the patterns. If a hit matches a pattern segment, the hit flip-flop that belongs to that segment is set. The output of the hit flip-flops is routed to the majority logic. The majority logic checks the number of activated hit flip-flops of each pattern. If the number of activated hits is larger than or equal to a predefined threshold, the pattern

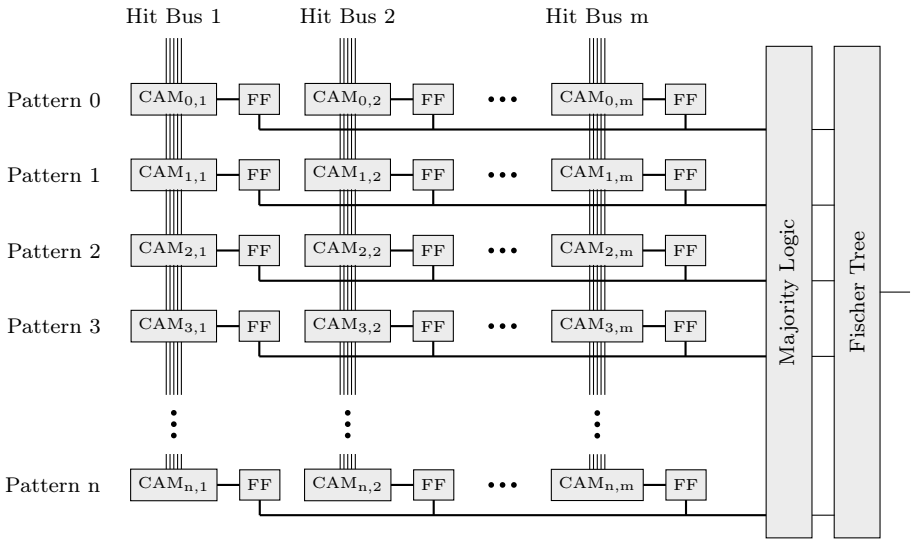


Figure 5.15.: Conceptual block diagram of the AM Chip. Source: [124].

is considered as found and passed to the Fischer tree [126]. The Fischer tree collects all the activated patterns and generates their pattern number. After all hits of an event have been applied to the AM Chip, the hit flip-flops are reset.

Although the functionality of the AM Chip is similar to a CAM, there are three main differences that disqualify commercial CAMs to be used for track finding: (1) The patterns and the search word are split into layers, and each layer section of the search word is compared independently with the incoming search words. (2) Not only one search word is applied to the input of AM Chip, but all the search words (hits) belonging to one event are applied sequentially. (3) All detected patterns are returned and not only a single one as it is usually the case for commercial CAMs.

The current version of the AM Chip is the AM06, which is the production version of the prototyping chip AM05 [11]. The AM06 chip has a capacity of 128 000 patterns that are split into eight segments for eight layers. Each segment has a width of 16 bit of which 2 to 9 can be configured as ternary cells. The logic of the chip runs at 100 MHz. For each layer a dedicated HSSLs at 2 Gbit/s is available. The found pattern numbers are written out of the chip at 60 MHz, which corresponds to a link speed of 2.4 GHz/s. Besides the pattern number, the AM06 chip also returns a hit map that indicates which layers of the pattern have been matched. The AM06 chip incorporates a Joint Test Action

Group (JTAG) interface to load the patterns, to configure the chip and to read out the status. The configuration consists of different parameters such as the threshold for the majority logic, the link speed, the usage of ternary cells can be configured over the JTAG interface. Before a run of the CMS track trigger, the patterns are also loaded to the AM Chip over the JTAG interface.

The development of the AM Chip took place in three phases. The different chip versions are listed in Table 5.1. The first generation AM01 was developed for the SVT of the CDF experiment. Its capacity was only 128 patterns. The limited capacity was compensated by building AM banks of 256 chips and 32 768 patterns in total. The AM02 chip was a design study to implement the AM Chip on FPGAs [127], which were a new device at that time. The second generation of the AM Chips is represented by the AM03 with 5000 patterns. The AM03 chip was used for the upgrade of the SVT in 2006 [87]. Since 2010, the development of the AM Chip has been driven by the ATLAS FTK. Three new versions of this third generation AM Chip based on 65 nm technology were developed: AM04, AM05 and AM06.

The AM06 will be the version of the AM Chip built-in to the CMS track trigger demonstrator. The development continues, and the discussion on a 28 nm chip has started. The reduction of the feature size by a factor of 2.3 will presumably increase the capacity by a factor of 5. An AM Chip with more than 500 000 patterns can be expected for these next generation of AM Chips. The AM05 chips were developed for the ATLAS FTK, which is part of the L2 trigger, where the latency requirements are not as tight as for the CMS track trigger. Therefore, the latency was not the main optimization goal, and a new AM Chip could be optimized in this direction. It would be beneficial if both the clock frequency and the I/O bandwidth would be increased. Ideas of an AM Chip that applies 3D integration technologies are also under development [130] with the goal to increase both the pattern density and speed further. The

Table 5.1.: Different versions of the AM Chip. Source: [124, 125, 127–129]

Property	AM01	AM02	AM03	AM04	AM05	AM06
Year	1992	1999	2000	2012	2014	2016
Patterns	128	64	5120	8192	4000	128 000
Layers	5	5	6	8	8	8
Ternary Bits	N/A	N/A	N/A	3–6	2–9	2–9
Clock	30 MHz	30 MHz	50 MHz	100 MHz	100 MHz	100 MHz
Technology	0.7 µm	FPGA	180 nm	65 nm	65 nm	65 nm

replacement of the AM Chip by FPGAs is also investigated again with the goal to realize a more flexible solution without ASICs [131].

5.4.2. PRM05 board

The PRM05 board has been designed to host the AM05 chip and is the first version of the PRM [132]. The purpose of the board is to test algorithms, to test the interfaces and to evaluate the performance of the parts. As the AM05 chip incorporated by the PRM05 is a prototyping chip with a low number of patterns, the PRM05 is not suitable to check the overall system performance. For this reason, the PRM06 board will be developed, which is described in Section 5.4.3. Figure 5.16 shows a photo and Figure 5.17 shows a simplified schematic of the board.

The PRM05 is a double-wide FMC with two FMC connectors and, thus, two PRM05 can be mounted on a Pulsar IIb carrier blade. The central processing device is a Xilinx Kintex-7 FPGA with up to 477 000 logic cells (Xilinx XC7K480T) [133]. The FPGA is directly connected with the FPGA on the carrier blade by eight bidirectional HSSLs with a bandwidth up to 10 Gbit/s each—four through each FMC connector. Additionally, 68 LVDS links are available between the two FPGAs.

The PRM05 board hosts 16 AM05 chips with a total of 64 000 patterns. Both the patterns and the numbers of the found roads are transferred between the FPGA and AM Chips via HSSLs with bandwidths of 2 Gbit/s. As the hit data of each layer are fed on a separate link, 128 HSSLs would be necessary to send the patterns to all the AM Chips. However, the FPGA provides only 24 transceivers in total, and some are already dedicated to the communication with the carrier blade. Therefore, fanout buffers duplicate the output HSSLs of the FPGA. This is possible because all detector hits have to be sent to all the AM Chips, which store different patterns. To increase the flexibility of the architecture, the AM Chips are split into two groups that can be accessed individually. The results from every AM Chip are transferred to the FPGA on a dedicated HSSL. The JTAG for the configuration of the AM Chips is realized separately for both groups of AM Chips. Both JTAG chains are directly connected to the FPGA that also provides a JTAG interface towards the FMCs. Thus, it is possible to control the AM Chips from the carrier blades by connecting the JTAG interfaces within the FPGA.

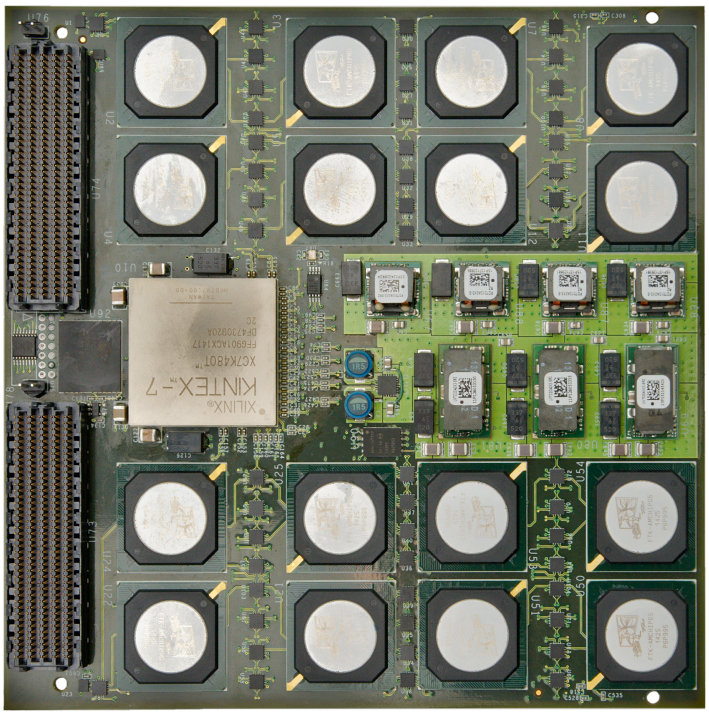


Figure 5.16.: Picture of the PRM05 board.

Beside the FPGA and AM Chips, a Double Data Rate version 2 (DDR2) memory with a size of 18 Mbit is part of the board. On the PRM05, due to a layout error, the signals connecting the FPGA to the memory are not connected properly. Thus, the memory cannot be used on the PRM05, and FPGA internal memory has to be used to emulate the memory. Due to the complexity of the board, the power supply and the clock generation are crucial for its operation and are realized directly on the PRM05 board.

5.4.3. PRM06 board

The PRM06 has been designed as the successor of the PRM05 and uses the AM06 chip. The design follows the PRM05 but fixes some errors and adds some components that were recognized as useful during the testing of the PRM05. The updated schematic of the PRM06 is shown in Figure 5.18.

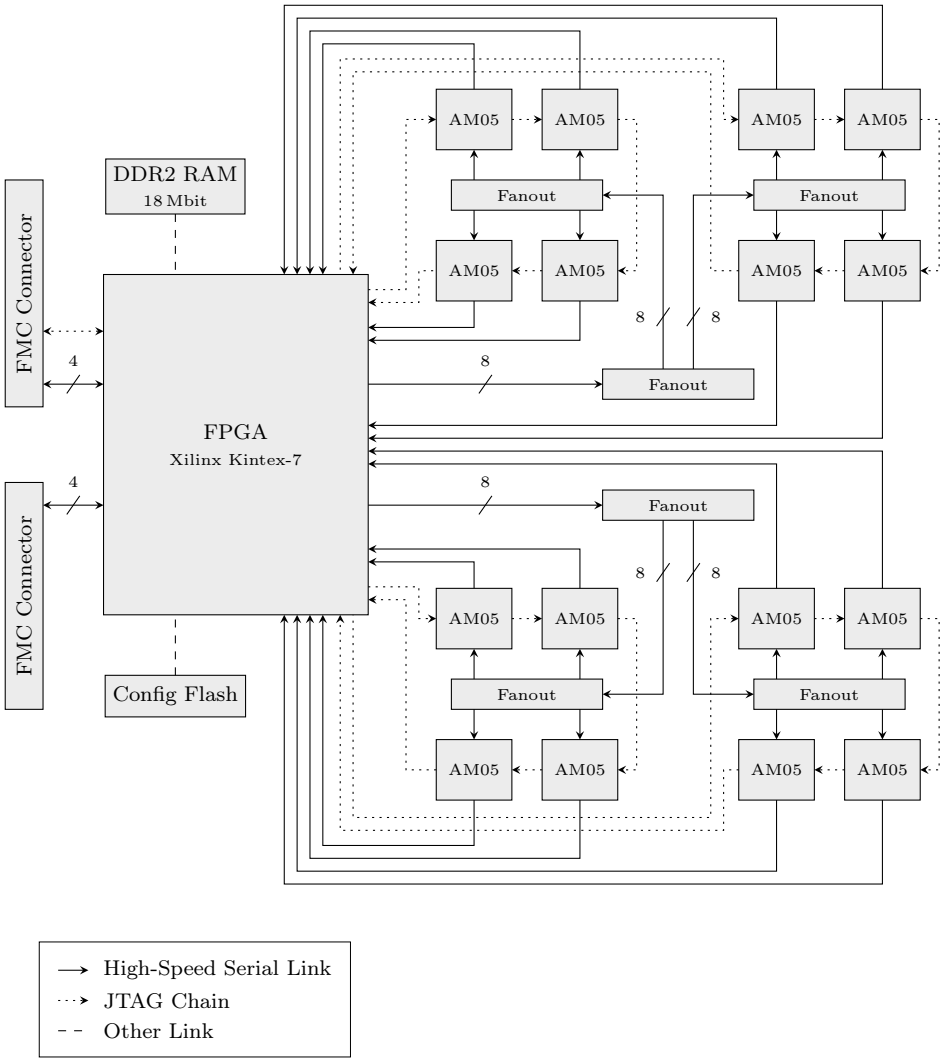


Figure 5.17.: Simplified schematic of the PRM05 board.

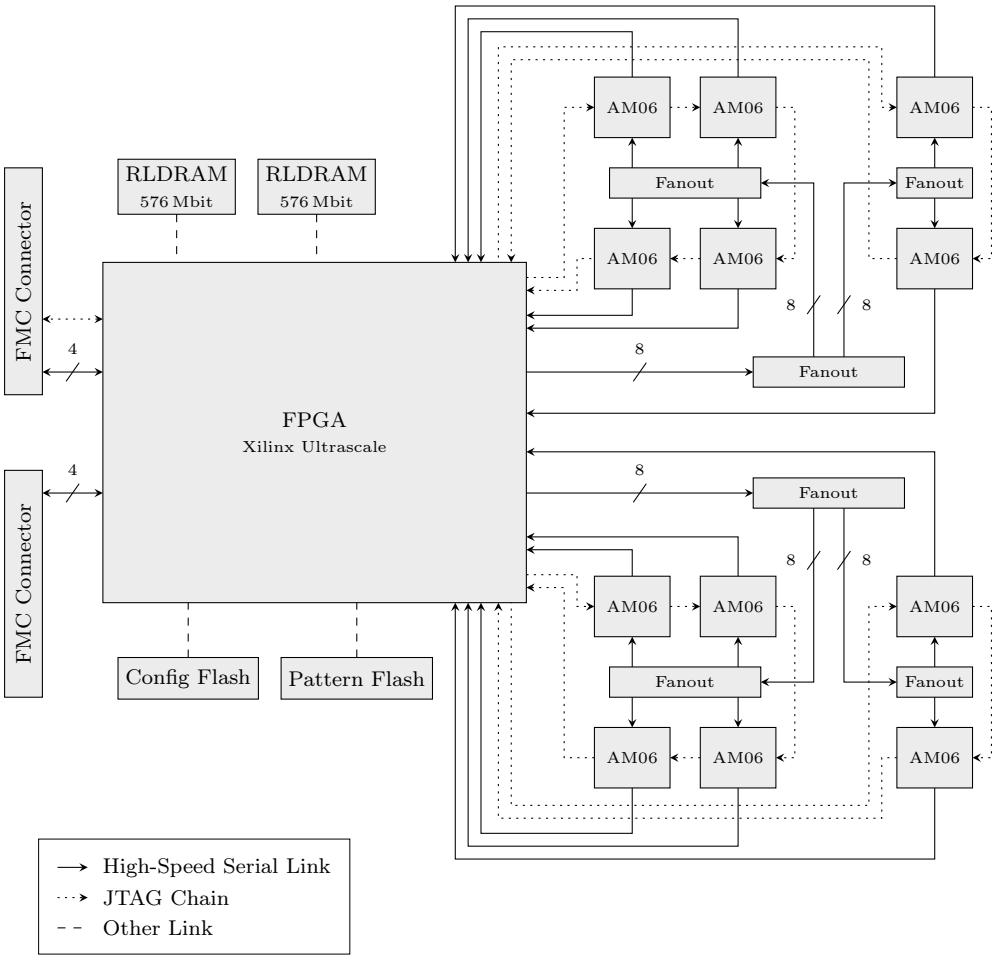


Figure 5.18.: Simplified schematic of the PRM06 board.

Because additional components are added to the PRM06 and to reduce the total power consumption, the number of AM Chips is reduced. Only 12 AM Chips are mounted to the PRM06. The capacity of these AM Chips is about 1.5 million patterns that are considered enough for a reasonable pattern bank.

The FPGA has been exchanged by an FPGA of a newer technology (Xilinx Ultrascale XCKU060F) with up to 726 000 logic cells [134]. This is about 50 percent more than the FPGA of the PRM05. With 2760 Digital Signal Processing (DSP) slices, also the DSP power has been increased by 40 percent. The external memory has been exchanged by a Reduced Latency DRAM (RLDRAM) from Micron (MT44K16M36RB-107E:A) with a capacity of 576 Mbit [135]. RLDRAMs are high-bandwidth memories with relatively low latencies [136]. They provide lower latencies than common Dynamic Random-Access Memories (DRAMs) and are larger and cheaper than Static Random-Access Memories (SRAMs). The size of the memory is large enough to store the associated superstrips of all patterns. Two flash memories have been added to the PRM06. One flash stores the configuration data of the FPGA, the other flash memory stores the patterns that are loaded into the AM Chips.

The increased number of patterns, the increased memory capacity, and the more powerful FPGA facilitate the testing and commissioning of the full system performance of a CMS track trigger based on AM Chips.

5.5. PRM05 board testing

Three PRM05 prototypes were produced and tested at KIT. The knowledge gained during the commissioning and the production test will be useful when the algorithms for the board is developed.

5.5.1. Test setup for PRM

A test setup was built up that allows the developers to access the PRM05 board remotely. The purpose of this test setup is that multiple developers can work with the same board from their desktop computers. It is especially useful for the fans necessary for cooling the boards make a lot of noise. Figure 5.19 shows the setting of the test setup.

The PRM05 board is mounted to a HiTech Global HTG-V6-PCIE-XXXX FPGA evaluation board [137], shown in Figure 5.20. The FPGA (Xilinx Virtex-6) on

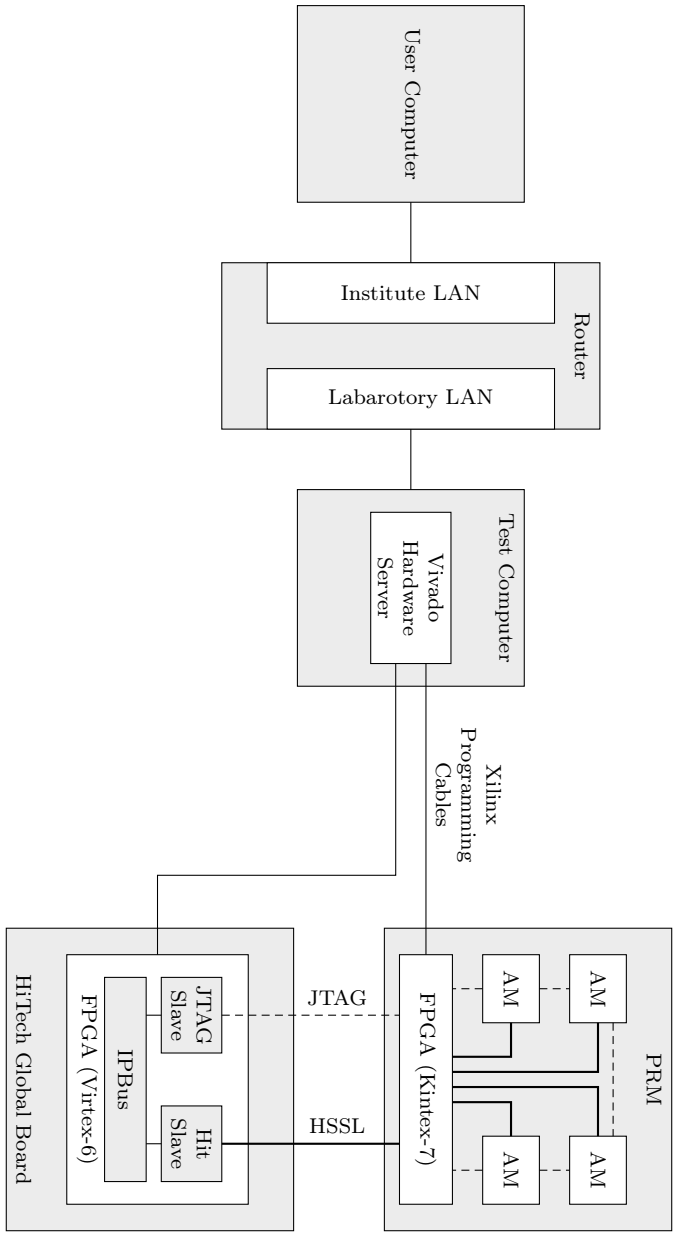


Figure 5.19.: The setup to test the PRM05 by accessing it remotely.

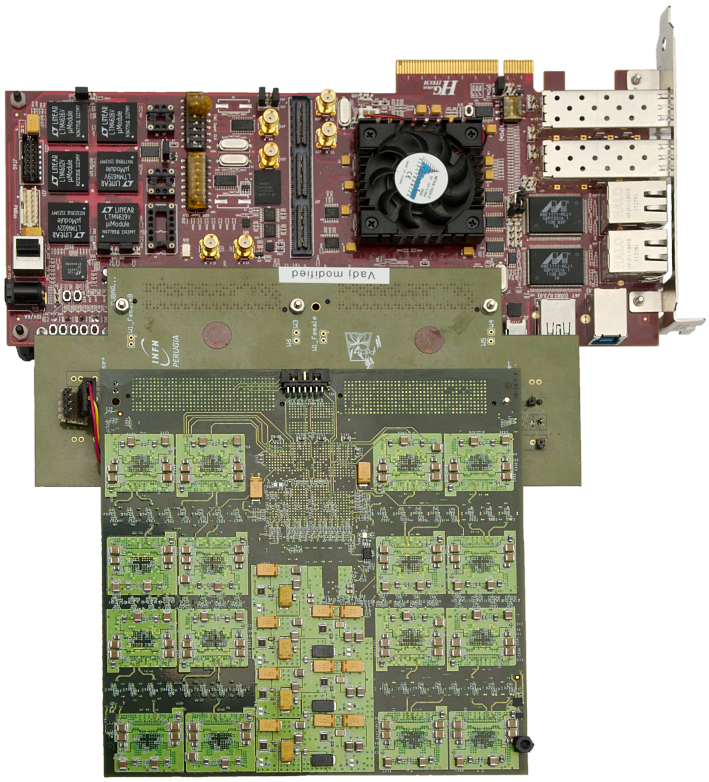


Figure 5.20.: Photo of the PRM05 mounted to the HiTech Global HTG-V6-PCIE-XXXX FPGA evaluation board.

the HiTech Global board builds the data source and sink for the PRM05 board, i.e. it emulates the Pulsar IIb carrier blade. To control the firmware in the FPGA on the HiTech Global board, the IPBus suite is used.

IPBus is a control system that facilitates writing to a bus in FPGAs over a Local Area Network (LAN) [138]. IPbus defines its on protocol on top of the User Datagram Protocol (UDP) protocol. Although UDP does not provide reliable data transmission, UDP is used because it is easier to implement in FPGAs than the Transmission Control Protocol (TCP). The IPBus suite consists of three parts: (1) A firmware block that decodes the packages and puts the data on an FPGA-internal bus. (2) A software—the control hub—that serves as a single point of access in a distributed system with several FPGA boards. (3) A Python and a C++ library that are included in the user programs controlling

the hardware.

In the FPGA on the HiTech Global board, several slaves are connected to IPBus. Four IPBus slaves build the data source for the PRM05 board. Hits are written to these slaves, which are then transmitted over four HSSLs to the PRM05 board. Another IPBus slave realizes a JTAG interface controller that is connected to the AM Chips via the FPGA on the PRM05 board. The control functions of the AM Chip are accessed by this JTAG interface, e.g. storing pattern banks to the chips, setting up the HSSLs or reading out error registers.

The FPGAs on both boards, the PRM05 and the HiTech Global, are configured from a desktop computer located next to them, This computer is running Scientific Linux 6 [139]; a Linux distribution that is maintained by Fermilab and is widely used at CERN. It is based on Red Hat Linux and adds packages for scientific applications to its repository. Scientific Linux was chosen because it is the only operating system that is supported by the IPBus suite. Additionally, Red Hat based Linux distributions are natively supported by the Xilinx FPGA design tools. To allow the firmware designers to work on the board remotely, the Hardware Server of Xilinx Vivado is running on the computer of the test setup [140]. By the use of the Xilinx Hardware Server, the FPGAs both on the HiTech Global board and on the PRM05 can be configured. The Xilinx Hardware Server also provides access to the Xilinx Integrated Logic Analyzer blocks that monitor the value of signals within an FPGA [140].

In order to make the PRM05 board via the HiTech Global board and the computer of the test setup accessible from any computer in the network, they are placed in a private network decoupled from the LAN of the institute by a router. Both the HiTech Global board and the computer are directly connected to the router. The router has port forwarding set up for IPBus packets and Xilinx Hardware Server packets. To make the port forwarding running, fixed Internet Protocol (IP) addresses must be configured for the HiTech Global board and the computer. Actually, the IPBus IP block always uses a fix internet protocol (IP) address as it has to be configured in the firmware.

5.5.2. Test procedure

To perform the production test, several firmware designs and Python scripts were necessary. Code has been provided for both firmware and test scripts by the developers of the PRM05 board. However, the provided code was written

in order to test algorithms on the PRM05 and was not suitable to analyze errors on the board. Therefore, the firmware and the scripts have been adapted so that predefined test patterns are transmitted on the connections which are easier to recognize than pseudo-random values. These test patterns were chosen that the transmission of the correct value and the endianness of the data can be tested.

The firmware on the HiTech Global board consists mostly of IPbus related parts. Changes were mainly necessary for the configurations of the high-speed transceivers of the FPGA [141]. The firmware of the Kintex-7 FPGA on the RPM05 has been reduced to the minimum. The hits transmitted from the HiTech Global board are distributed to the AM Chip groups that have been populated on the board. Instances of the Xilinx Integrated Logic Analyzer were added to the firmware design to check the received signals. Like the signals of the high-speed transceivers also the incoming JTAG interface has been connected to the JTAG chains of the AM groups.

The provided Python scripts cover different functionalities. The scripts configuring the high-speed transceivers of the FPGA on the HiTech Global board and the high-speed transceivers of the AM Chips were applied unchanged. The script that stores the patterns to the AM Chips was adapted to store the patterns in the chip which support the test process. Other scripts exist that write the hits to the IPBus slaves on the HiTech Global board and starts their transmission to the PRM05.

With these adapted the firmware blocks and Python scripts, different tests have been performed:

- **Power supplies and clock generation:** After the production, the first thing to check is that the power supplies and the clock generation work properly.
- **JTAG chains:** The JTAG interface is less error-prone than the HSSLs and, thus, the JTAG interface is tested first. As most devices with an included JTAG interface, the AM Chip provides a command to read out its device id that is used for this first test. If the AM Chips respond to the JTAG commands, they are properly powered up and running.
- **HSSL between the HiTech Global board and the PRM05:** The input links are tested by hits sent by a Python script. In the other direction, no script is available to read out the incoming data on the links. Therefore, test patterns are generated by the FPGA on the PRM05 and read by a Xilinx Integrated Logic Analyzer block.

- **HSSL from the AM Chips to the FPGA:** In a first step, the HSSLs from the AM Chips are tested with pseudo-random values, which can be generated by the transceivers of the AM Chips.
- **HSSL from FPGA to the AM Chips:** The links to the AM Chip cannot be tested independently. Patterns are stored within the AM Chips and hits that produce a match with the patterns are sent to the AM Chips. If they are successfully found, the return links are also good. By this test, the complete chain of HSSLs is tested.

Three prototypes were successfully tested by these procedures. Diverse issues on these PRB05s were discovered, among other things: a broken FPGA and wrongly mounted devices. However, the most time was spent on the configuration and debugging of the high-speed transceivers.

6. Framework for the System Simulation of the CMS Track Trigger

The system simulation presented in this chapter has been developed in collaboration with INFN Pisa. The main concepts of the simulation and the parts programmed in SystemC have been developed at KIT. Modules written in Very High Speed Integrated Circuit Hardware Description Language (VHDL) have been contributed by INFN.

6.1. Motivation of the system simulation

The system simulation is a system-level hardware simulation of the CMS track trigger according to the AM approach, see Section 5.2. It covers all parts of the data processing chain, from the readout chips of the detector up to the signal transmission towards the first-level trigger. As this includes many modules, the simulation time is an issue that has to be considered. However, the data of each trigger tower is processed independently, and it is enough to simulate only one trigger tower. Due to the random behavior of the physics processes, the distribution of the data delivered by the detector modules is non-deterministic. Therefore, it is important to apply input data to the system simulation that correspond to the output of the silicon tracker.

The main goal of the system simulation is to support the evaluation of possible system architectures for the CMS track trigger. The evaluation of system architectures comprises two aspects: Firstly, the properties of a system architecture can be determined. Examples of these properties are latencies of both individual parts and the complete system, bandwidths on data links, buffer sizes, etc. Secondly, the models of the components of the system simulation have to be easily re-configurable. Likewise, the number, arrangement of the components and the connections between them have to be flexible. This allows the system

designers to test different system architectures quickly. In other words, the system simulation lays the base for design space exploration [142] of the CMS track trigger.

Additionally, the system simulation should serve as a test bench for algorithms and FPGA code. Whereas the algorithm testing is an addition to the architecture evaluation feature, the FPGA code testing is a more practical application.

6.1.1. Selection of the simulation method

Within the CMS collaboration, a framework for the simulation of the CMS detector and its data processing has been developed [12]. The framework is called CMSSW and is available on GitHub [143]. Its goals are the development of reconstruction and analysis software for CMS.

CMSSW is based on the Event Data Model (EDM) [144]. The EDM builds the simulation core; it manages the data that belong to an event and executes the modules that process the data. Within CMSSW, an event corresponds to a bunch crossing in the detector and stores both the raw data and the reconstructed data related to the event. The processing is split into different modules that all realize a specific function, e.g. loading simulated output data from a detector, fitting the track of a particle, etc. A user of the framework sets up a simulation by configuration files. These configuration files define which data are loaded to the event and which modules are executed in which order. During the execution of the simulation, the events are processed independently of each other. Therefore, the execution of CMSSW is accelerated by running it on a computer cluster.

Although CMSSW is also used to develop algorithms that are later realized in hardware, it is not suitable to simulate hardware-specific effects. The main limitation of CMSSW is the missing timing information. As all the modules are executed during one event step, all results become available at the end of the event. Therefore, latencies cannot be modeled. Additionally, this execution model is not able to simulate silicon tracker after the Phase-II upgrade in all its details. For example, the buffering of hits in the data concentrator of the detector modules for several bunch crossings (Section 3.5.1) cannot be realized in CMSSW. By design, an event in EDM only has access to its own data and not the data from other events. Therefore, the stub selection within the data concentrator cannot be simulated in CMSSW. Other limitations of CMSSW in

terms of hardware simulation are no built-in support for data types with limited bit widths and no simulation of the communication between the modules.

The traditional Hardware Description Languages (HDLs), e.g. VHDL and Verilog, fulfill the requirement of accurate hardware simulations. They facilitate cycle- and bit-accurate simulations of hardware, i.e. all details are included. Although this is required for the final design of the components of a system, this may be not necessary for the evaluation of an entire system. Two reasons discourage the implementation of a detailed model of the CMS track trigger in an HDL. Firstly, the development of a detailed model of the CMS track trigger components takes a lot of time. When multiple options should be evaluated quickly, this is a drawback. Secondly, the simulation of a detailed model increases the simulation time significantly. It is indeed possible to model the CMS track trigger with fewer details, i.e. on a higher abstraction level. However, the traditional HDLs are not specifically designed for this purpose. Furthermore, specific tools are necessary to run HDL simulations, e.g. Mentor Graphics ModelSim [145]. This can make it cumbersome to integrate the system simulation with other software available.

To overcome the limitations mentioned above and still have the possibility to simulate hardware-related effects, SystemC [146] has been chosen as the modeling language of the system simulation. SystemC is a freely available library that extends C++ with features that make it possible to simulate hardware. As it is written in C++, it compiles and runs on every standard computer. Originally, it was designed for co-simulation of hardware and software [147] but is also well suited to simulate large hardware systems. With SystemC, systems can be simulated on high abstraction levels, but simulations at the Register-Transfer Level (RTL) are also possible. Therefore, SystemC is well suited for the system simulation of the CMS track trigger in which different parts are described with different amounts of details. On the one hand, the CMS track trigger is a large system with many components, and the system simulation covers an essential part of it. To keep the simulation time at a reasonable level, many parts of the CMS track trigger within the system simulation are modeled on a high abstraction level. On the other hand, parts of the simulation that shall be investigated in more details or interact with parts implemented in an HDL may be simulated at the RTL. This is especially useful when the system simulation is used as a test bench for HDL code.

Due to the reduction of details in the models, the time to implement a part of the system simulation takes less time. This allows the system designer to evaluate several designs of a single block faster. Furthermore, as SystemC is

based on C++, many libraries are available that may be included in the system simulation. Libraries may already implement a complex function of a block, e.g. associative containers may be used to model the core of the AM Chip. Libraries are also highly optimized and, thus, often reduce the execution time of the simulation.

6.1.2. A brief introduction to SystemC

As already mentioned, SystemC is a C++ library that adds features for hardware simulation to standard C++ [148]. It combines the advantages of C++ object-oriented programming and HDLs. SystemC facilitates the simulation of digital systems on higher abstraction levels than HDLs and, thereby, enables higher simulation speeds. As a SystemC model is basically C++ code, a simulation may be compiled into an executable that runs then like a usual program on a computer. An open source implementation of the library is available from Accellera [149]. This implementation constitutes the reference implementation of the current IEEE SystemC standard [146].

The features for hardware simulation that SystemC provides are: (1) *Concurrent execution of processes*—Hardware processes run in parallel inherently. The simulation kernel of SystemC schedules the simulation processes in such a way that the result is equal as if they would run in parallel. (2) *Notion of time*—To express the time that passes in the simulated system, the SystemC kernel adds information about time to the execution. (3) *Hardware-specific data types*—The number of bits of signals is limited in hardware, this may have some effects on the accuracy of the results. SystemC provides data types with variable bit widths. In addition, resolved data types, data types that may be at states like undefined or high-impedance, are necessary to model buses in SystemC. (4) *Modeling of hierarchy*—Most digital systems are hierarchical. In SystemC, functional modules can be nested into other modules and, thus, build up hierarchies is possible.

SystemC models basically consist of three primitives: modules, ports and channels. Figure 6.1 shows a small example with those elements. Modules represent the functionality of the model. They either contain processes that execute some code, or they gather sub-modules and, thus, build up a hierarchy. Processes can be configured to be sensitive to certain signals, i.e. their execution is triggered by the change of the signal.

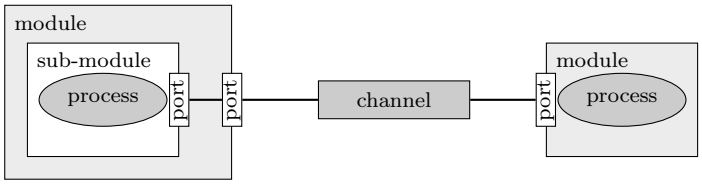


Figure 6.1.: Modeling with SystemC.

The channels transfer information between the modules. Different types of channels exist, they may model a connection as simple as a wire or as complex as a full bus system. To connect a channel with a module, the module must provide a port. Ports make data from inside a module accessible to the outside. The process of connecting a channel with a port is called binding. It is also possible to bind a port of a module to a port of one of its sub-modules.

For the system designer, the simulation of a SystemC model looks like the execution of a program, but in the background, the simulation follows some predefined steps. Figure 6.2 shows these steps that are executed during a SystemC simulation. The two most important phases are the elaboration phase and the evaluation phase. The system designer is in charge of the elaboration phase. During this first phase, he builds up and configures the model, and he sets up the simulation. With the command `sc_start()`, the control is passed to the SystemC simulation kernel. After the internal initialization, the kernel goes to the evaluation phase. In the evaluation phase, the kernel executes the processes in the modules that are scheduled at the current time. When no more processes are available to be executed, the time is advanced, and the evaluation phase starts again. After some defined time, the simulation ends, and the SystemC program finishes.

6.2. Principles of the CMS track trigger system simulation

To achieve the goals of the system simulation, some principles have been defined on which the system simulation of the CMS track trigger is based. These principles are illustrated in this section and have been presented in [13].

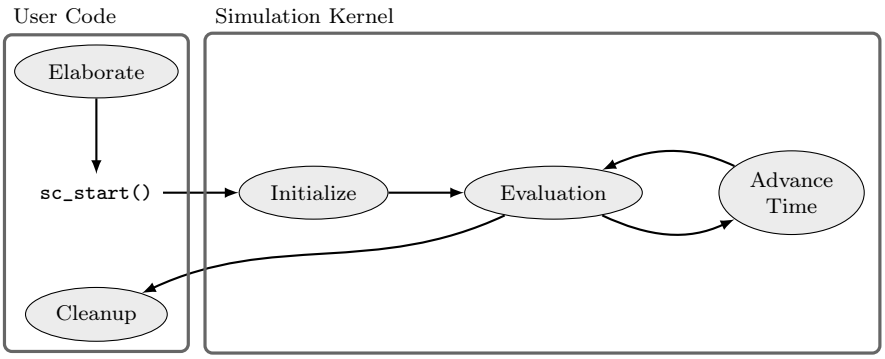


Figure 6.2.: Simulation process of SystemC. Source: [148]

6.2.1. Generic, functional blocks and channels

Within the system simulation, the simulation is split into functional blocks, which are described in Section 7. A functional block is not necessarily a piece of hardware. For instance, it could also be a block within an FPGA. The partitioning of the CMS track trigger for the system simulation is more driven by the functional blocks discussed by the CMS track trigger community.

With the goal to keep the simulation time low, just enough details, which allow the evaluation of the system parameters, are included. The blocks are also kept as generic as possible, which facilitates to test different configurations quickly. The easy configuration is also supported by the concept of configuration objects, presented in Section 6.3.5.

Besides the blocks, also the properties of the communication links can be modeled within the system simulation. The feature of SystemC to implement complex communication facilities are hierarchical channels [148]. A hierarchical channel is basically a SystemC module with processes that model the behavior of the communication link. Within the simulation framework, the properties of a communication link can be emulated by a hierarchical channel and, thus, add the correct latency to a communication link. These channels can also be configured by parameters.

6.2.2. High simulation speed

Although only one sector of the CMS track trigger has to be simulated, the simulated system is still large. Thus, it is important to model the system in a

smart way to avoid long execution times. Because the capabilities of SystemC to simulate digital systems on a high abstraction level are no guarantee for high simulation speeds, two modeling techniques were applied to the system simulation to increase the simulation speed.

Firstly, wherever possible, `sc_buffer` channels are used in the model. Usual channels notify the receiving process only if their value has been changed. However, the `sc_buffer` channel triggers the process always when a value has been written to the channel even if the value does not change [148]. The block is notified by the channel when the input has to be read, and the process that checks the input of the block does not need to be regularly activated. Therefore, it is possible to remove the clock from blocks that trigger the checking of the input. This reduces simulation time especially for the detector module models where a new value is not expected every clock cycle. The drawback is that the blocks have to be designed carefully not to mess up the arrival times of different signals at a module. Furthermore, latencies of blocks that are defined by clocks must be modeled in some other way, see Section 6.2.3.

Secondly, there exist two main classes of processes in SystemC: `SC_THREAD` and `SC_METHOD`. Usually, `SC_THREAD` is used as the modeling is easier with this process type. However, the simulation time can be significantly reduced by modeling with `SC_METHOD` processes, as Hosseinabady described in [150]. The long execution time of an `SC_THREAD` process originates from the fact that for every `SC_THREAD` process a dedicated operating system thread is started [148]. As each thread has its own stack memory, switching between two threads causes some overhead in execution time. Additionally, the number of `SC_THREAD` processes is limited by the operating, and this could be a problem for large systems, such as the CMS track trigger. In return, an `SC_THREAD` may be halted at any position of its function and does not need run completely through, as an `SC_METHOD` process has to. The reason is that the `SC_METHOD` process is just a function called by the SystemC kernel at the designated time [148]. As a C++ function stores its variables in the stack memory and shares it with the other functions from the same thread, all functions have to finish in the order they were started. Otherwise, the data in the stack memory would become invalid. Due to the simple nature of the `SC_METHOD` process, several `SC_METHOD` processes may be necessary to replace one `SC_THREAD` process. Another drawback of `SC_METHOD` is that no state can be saved within the process because the function of the process is called every time anew when it becomes activated. Class variables have to be added to the module to store data. However, the reduced simulation time usually justifies the usage of `SC_METHOD` processes for large simulations.

6.2.3. Modeling of latencies

As described in Section 6.2.1, the simulation speed of the system simulation is increased by leaving out details in the model. At the same time, the timing of the blocks and communication links should be as accurate as possible, because one of the goals of the system simulation is to evaluate the system latency.

To resolve this contradiction, an additional delay block may be added to a functional block. Thus, the function is decoupled from the timing of the block. The function may be programmed on a very high abstraction level without considering any timing. The signal generated by the functional part is then fed to a delay block that adds the required latency to the signal. In a similar manner, the latency of communication links can be emulated by hierarchical channels that include a delay block.

These concepts also facilitate the configuration of the latency of blocks and signals and, thus, to test the effects that different latencies may have. In this case, the latency of a block or link needs to be known from other sources, e.g. existing hardware, HDL simulations, or an estimate has to be used. Consequently, the accuracy of the latency evaluation of the CMS track trigger depends on the quality of the configured component latencies.

6.2.4. Input data

To achieve results, that are as accurate as possible, the input data of the system simulation should correspond to the input of the real CMS track trigger. Data sets of Monte Carlo simulations [151] of particles collisions in the CMS experiment exist. These data sets are fed into the CMSSW framework [12] that simulates the effect of the CMS detectors on the produced particles. Depending on the type of data needed, different modules can be executed in a CMSSW run and then stored to disk.

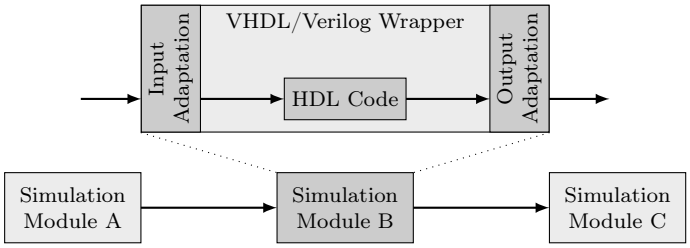
The system simulation makes use of different data sets generated by CMSSW. Firstly, for the configuration of the system, data about the silicon tracker layout, i.e. coordinates of modules, are extracted from CMSSW. The pattern banks that configure the AM Chips are also generated within CMSSW. Secondly, the hits detected by the detector modules are used in the system simulation. The format of the extracted hits corresponds to the output format of the front-end chips, i.e. a hit consists of a discretized strip number and a bend representing the momentum of the particle.

6.2.5. The system simulation as a test bench

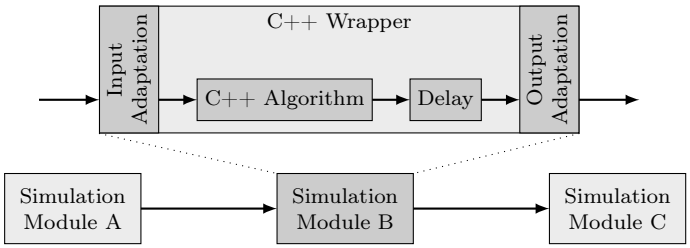
The system simulation can be used as a test bench in two cases. Firstly, HDL code that will be used to configure FPGAs can be tested within a simulation of its actual environment with realistic input data. Secondly, new algorithms existing only as software implementations may be included, and their effect on the system can be evaluated. Figure 6.3 visualizes these two cases schematically.

To use the system simulation as a test bench for FPGA firmware, the system simulation has to run in a tool that supports the co-simulation of SystemC with HDLs. In other words, the tool must be able to simulate a model that contains blocks written in SystemC and blocks written in either VHDL or Verilog. Mentor Graphics ModelSim facilitate this type of co-simulation [152]. To test a module developed for the usage in an FPGA, the corresponding SystemC module is exchanged by the FPGA firmware. Within the system simulation, the HDL module receives the same data as it would in the CMS track trigger and also returns its results to the CMS track trigger simulation. A wrapper is still necessary in most cases, because the signals coming from blocks of the system simulation may have a simplified format or are even missing in the simplified model. The wrapper adapts the data formats and emulates signals that are not present in the system simulation.

As a SystemC simulation is basically a C++ program, any code written in C++ may be included into the system simulation. This makes it possible to test new algorithms within the system simulation without much effort. It is even possible to include algorithms of scientists who do not know SystemC within the system simulation. The effects which these algorithms have on the CMS track trigger as a whole can thus be evaluated quickly. However, the algorithm testing feature is not limited to code written in C++. Many libraries exist to include code of different programming languages into C++ programs. For example Python, widely used in physics, can be executed in C++ by the Boost.Python library [153], which is part of the popular Boost library collection. As shown in Figure 6.3, a wrapper is necessary to embed an algorithm into the system simulation. This wrapper places the C++ code within a process, so it is regularly executed. It also applies the input signals of the block to the arguments of the C++ function containing the functionality to be tested. A delay may be added in the wrapper to simulate the latency of the hardware implementation of the algorithm.



(a) Testing of code for FPGA.



(b) Algorithm testing.

Figure 6.3.: Using the system simulation as a test bench. Source: [13]

6.3. Library to simulate regular structures in SystemC

Although the simulation of the CMS track trigger needs to cover only one sector of the detector, many large regular structures exist within the modules of the system. Regular structures are understood as sets of the same SystemC elements that have the same or nearly the same properties and belong together. For example, the readout chips on the detector modules of the CMS silicon tracker are arranged in a 2×8 array. Writing similar program code repeatedly is not only annoying but also bad practice, as it decreases the maintainability of the code.

SystemC does only provide limited features to implement such regular structures without writing a lot of program code. To overcome this problem, a library called sc_map has been developed that allows creating models of regular structures in SystemC easily. Additionally, it also supports the configuration of the objects and, thus, facilitates the automatic model generation of large systems.

The content of this section has been published in [14]. The following sections

contain Unified Modeling Language (UML) diagrams to visualize the structure of the library classes. These diagrams are not complete and show only the details relevant for the explanations.

6.3.1. State of the art for modeling regular structures in SystemC

One possibility to create regular structures of SystemC elements, such as modules, ports and signals, is to apply the native features of C++. On the one hand, the ordinary arrays that are part of the core language features allow gathering identical elements. For example, Black proposes the usage of arrays in [148]. On the other hand, the C++ Standard Template Library (C++ STL)[1] provides different containers to organize objects [154].

The issue of both techniques, when applied to SystemC data types, is that they require the data type of the stored objects to be default constructible. Default constructible means that there must be a constructor of the class that is called without any arguments. While SystemC elements provide such a constructor, typically at least a name is passed to the constructor of an element. By this name, an element can be identified during the execution of the simulation. Therefore, the default constructor is usually not used.

A solution to overcome this limitation is to place pointers to elements in the arrays or containers and generate the elements dynamically by `new()` in the heap memory. Thus, the programmer is responsible for the memory management of the objects, which should be avoided whenever possible. This approach also misses SystemC-features, such as port binding and writing to channels. For instance, if the ports of a module need to be bound to a set of channels, a loop iterating over all ports and channels is necessary.

To improve the situation, a container called sc_vector with better compatibility with SystemC objects has been developed. Since the 2011 revision, the sc_vector class is officially part of the SystemC standard (IEEE 1666-2011) [146]. Basically, sc_vector works equivalently to the vector container of the C++ STL but adds some SystemC-specific features. For example, a name can be passed to sc_vector that is then used as the base of the name of all its elements. It is even possible to pass more arguments to the constructor by

[1] The C++ Standard Template Library (C++ STL) is part of the C++ standard. The C++ STL provides—among other things—containers to store objects and iterators to traverse those containers.

the concept of a *creator*, explained in more details in Section 6.3.5. Furthermore, the class provides functions to bind vectors of signals to vectors of ports directly.

To model linear structures, sc_vector is well suited. However, when it comes to multidimensional arrangements of SystemC modules some kind of mapping would be necessary to use sc_vector. Another limitation is the index of an sc_vector always runs from 0 to $size-1$. In many cases, it is more convenient if any range can be chosen or even arbitrary data types could be used as the index of the container—comparable with the map container of the C++ STL [154].

For VHDL, special commands exist for the creation of structures of modules [155]. However, they are pretty basic. The provided `for-generate` statement runs through some variables and simulation elements can be instantiated each iteration. It is basically equivalent to a for-loop in C++. Thus, the generation is as manual as the generation of structures by standard C++ features.

6.3.2. Overview of the sc_map library

To overcome the limitations of sc_vector, a library called sc_map has been developed. The library allows the system designer to organize SystemC objects of the same type in a container that is more flexible than sc_vector. Any SystemC object type derived from `sc_object` may be organized in an sc_map container. The SystemC object type is defined by a template argument of the sc_map class.

Similar to the map container of the C++ STL [154], the SystemC objects are stored within the container and accessed by a key. The name of the library originates from the similarities of both classes. As with map of C++ STL, square brackets and the `at()`-function are used to access the elements in the container. Generally, this key may be of any data type. The only restriction is that it is wrapped within the sc_map-specific key class, described in Section 6.3.7.

So far, the functionality corresponds to those of the map container of C++ STL. However, sc_map also facilitates a structure to be assigned to objects within the container. For that purpose, ranges are associated with the container. A range defines which keys are part of the container and defines the order of these keys.

Basically, any structure may be defined by the designer by writing a derived `sc_map_range` class, described in Section 6.3.8. However, the sc_map library

provides some common structures that are ready to use. Two different categories of structures exist currently: regular structures and the list structure.

Regular structures organize SystemC objects in regular, geometrical arrangements of different dimensions. Currently, regular structures for linear, square, cube and 4-dimensional arrangements are implemented in the library, of which the first three are illustrated in Figure 6.4. The functionality of the linear structure is very close to the one of an sc_vector, but with a bit more flexibility. For instance, the index may start with any integer and not only zero, and the counting direction of the index could be either up or down. The other regular structures are multidimensional versions of the linear structure.

The second type is the list-like structure that allows organizing SystemC objects in arbitrary order. Still, the SystemC objects are accessed by a key. However, by knowing the key, it is not apparent which is the next key in the order. The list is a very flexible container and comparable with the functionality of the map container of C++ STL [154].

So far no special library would be necessary as these functionalities could be realized by the standard library containers of C++. However, sc_map goes beyond that and adds SystemC-specific features and the possibility to configure the SystemC objects within an sc_map. For instance, an sc_map of SystemC signals can be directly bound to an sc_map of SystemC ports, which simplifies the modeling of large models enormously. Furthermore, sc_map provides the possibility to write the same value to every signal of an sc_map and to trace a whole list of signals for debugging.

The effect of sc_map on the execution time is negligible. The code of sc_map is only executed during the evaluation phase of the simulation. As the SystemC objects have to be generated anyway, the overhead generated by usage of the sc_map library is small. After calling their constructor, the SystemC objects

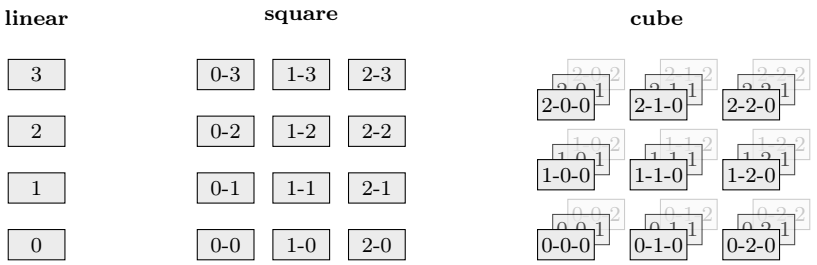


Figure 6.4.: Regular structures within sc_map. Source: [14].

are managed by the SystemC kernel that uses its own system to keep track of processes and signals.

6.3.3. Modeling with sc_map

In this section, the modeling of systems with sc_map is explained on the base of some small examples. Figure 6.5 shows an example system consisting of four sensors, which are connected by individual wires to a processor with four inputs. The sources, the wires and the in-ports of the processor are all modeled by list-like sc_map structures. All three structures are indexed by the same set of keys: north, east, south and west. A vector of strings containing these keys is passed to the constructor of sc_map to initialize the objects.

Listing 6.1 shows the definition of the wires by three different approaches: a list of the standard C++ STL, an sc_vector of the SystemC standard and a list-like sc_map. Obviously, the usage of the C++ STL list is the most cumbersome as a loop is needed. The difference between sc_vector and sc_map is in this case not so big for list-like structures. In cases where the objects may be accessed by an index that is not an integer starting from zero, sc_map provides a more convenient solution as the index can be used directly.

Listing 6.2 shows the usage of the more SystemC-specific features port binding and writing to signals. Port binding works the same for sc_map and sc_vector and is very easy as the structures can directly be connected together. For the C++ STL list, a loop is necessary to iterate over the ports and the signals. For writing to a set of signals, one line is enough for sc_map. As this feature is not directly supported by sc_vector, a for-loop is used as with the C++ STL list.

What has been said about list-like structures also applies to the regular structures: linear, square, cube, 4D. The main difference is the construction. For regular structures, the constructor takes the size of every dimension of the coordinate. Each size of a dimension can be an arbitrary integer, i.e. it can be negative, and a range does no need to start with zero. An example can be seen in the next section about slicing. A square structure of 5×4 ports with coordinates starting at (1,1) is shown in Figure 6.6.

The binding of regular structures works as with list-like structures. However, the simplification of the written code is even larger, as sc_map avoids nested loops.

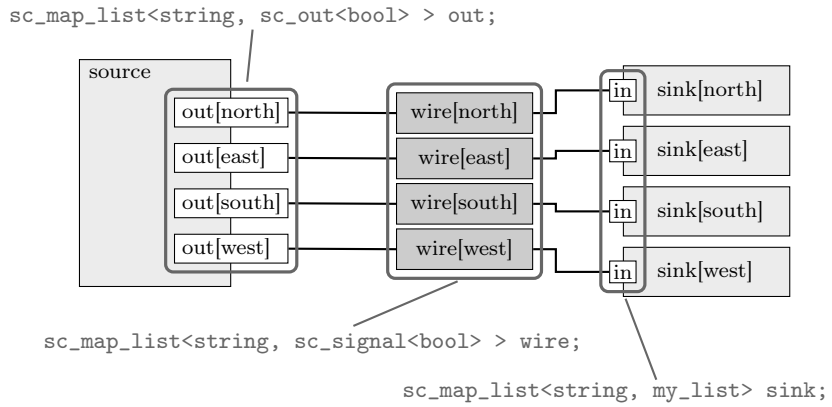


Figure 6.5.: Example system with three list-like sc_map structures all with the same keys (north, east, south, west). A source with four ports (left side) connected to four wires that are also connected to four sinks (right side). Source: [14].

Listing 6.1: Comparison of different approaches to model a set of SystemC signals with the keys: north, east, south and west.

```
// define keys
std::list<std::string> sensor_names {
      "north", "east", "south", "west"};

// using standard C++ ------------------------
std::map<std::string, sc_signal<int>> wires;
for(std::string name : sensor_names) {
   wire[name] = new sc_signal("wire-" + name);
}

// using sc_vector ---------------------------
// strings as keys not possible, using keys: 0-3
sc_vector<sc_signal<int>> wires("wire", 4);

// using sc_map ------------------------------
sc_map_list<sc_signal<int>> wires("wire", sensor_names);
```

Listing 6.2: Comparison of different approaches for port binding and writing to multiple signals.

```
// standard C++ approach ------------------------------
std::map<std::string, sc_signal> wires;
for(std::string name : sensor_names) {
    sink.inputs[name].bind(wire[name]);
    wires[name].write(0);
}

// sc_vector approach ---------------------------------
sink.in.bind(wires);
for(int i = 0; i<4; ++i) {
     wire[i].write(0);
}

// sc_map approach ------------------------------------
sink.in.bind(wires);
wires.write(0);
```

6.3.4. Slicing of sc_map structures

As described in the previous section, sc_map simplifies the creation of large regular structures. However, systems usually do not keep the same complexity over the complete data flow and the number of dimensions changes between the modules of the simulation. For example, the full-mesh interconnect between the different blades, which connects the *data organizers* with the *processor organizers* in the system simulation, is a square structure. Because it connects each blade with all the other blades, it is of size $n_{blades} \times n_{blades}$. However, each data organizer needs only to be connected to the channels leaving from it, which is a linear slice of the entire interconnect. The slicing feature of sc_map facilitates these kinds of tasks.

The slicing feature is realized by iterators [156]. Iterators are a concept widely used by the C++ STL. An iterator is used to access the elements within a container in a certain order. By doing so, the iterator always points to one element in the container, which can be accessed through the iterator. The next element within the container can be accessed by incrementing the iterator.

The iterators of sc_map facilitate very flexible slicing of the elements in the container. Thus, a slice may have the same or a lower number of dimensions than the container to which it is applied. Figure 6.6 and the corresponding Listing 6.3 give an example where two slices of a square structure of ports are

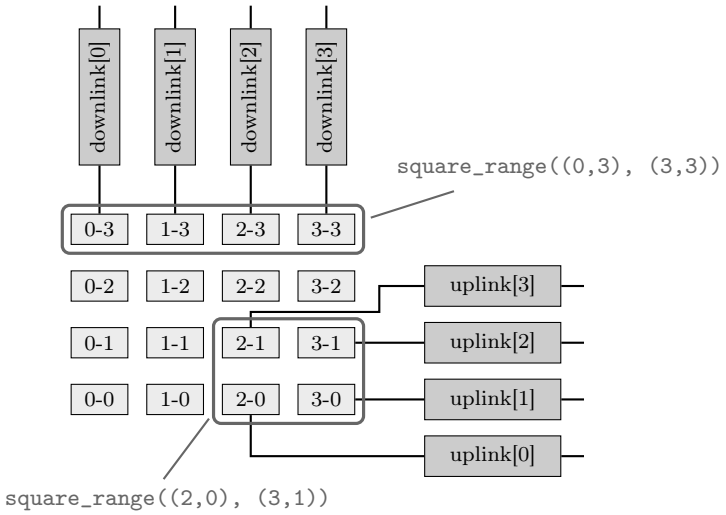


Figure 6.6.: Selection of a linear (top) and a square slice (bottom right) from a 4×4 square sc_map of SystemC ports. The two slices are each bound to a linear sc_map of SystemC signals. Source: [14].

Listing 6.3: Example of slicing of an `sc_map_square` structure.

```
sc_map_square<sc_port<bool> > port_array(4, 4);
sc_map_linear<sc_signal<bool> > downlink(4);
sc_map_linear<sc_signal<bool> > uplink(4);

// binding to a 1-dimensional slice
sc_map_square_key start_key_1(0, 3);
sc_map_square_key end_key_1(3, 3);
port_array(am_board_in_sig(
        start_key_1, end_key_1)).bind(downlink);

// binding to a 2-dimensional slice
sc_map_square_key start_key_2(2, 0);
sc_map_square_key end_key_2(3, 1);
port_array(am_board_in_sig(
        start_key_2, end_key_2)).bind(uplink);
```

bound to channels arranged in linear structures. To create a slice, the container and the selected range are passed to the constructor of an iterator.

The bind function of structures can take iterators, and the iterators also provide a bind function. Thus, the slicing is very flexible, and any combination is possible for binding of ports and signals: structure-to-slice, slice-to-slice and structure-to-structure.

6.3.5. Creation and configuration of structures with sc_map

One benefit of the sc_map library is that the programmer does not have to take care of the creation of the single SystemC objects. However, the SystemC objects often need to be configured, e.g. the size of a memory, the ID of the object or just a name to identify the object. As this configuration is specific for every single object, the sc_map library also includes a concept for the configuration of the objects arranged in the container. An overview of the configuration concept is shown in Figure 6.7

The SystemC objects are created in the heap memory and organized within sc_map. Consequently, an sc_map container is completely configured after the call of its constructor. This may sound like a limitation, but as all SystemC objects must be defined anyway by the start of the simulation, i.e. the call of function `sc_start()`, this is usually not an issue. However, it also implies that all configuration parameters needed to create the objects arranged by an sc_map structure must be available when the constructor of the container is called. Although some parameters may be changed later, it is advised to configure the objects entirely from the beginning to avoid additional iterations over the sc_map container.

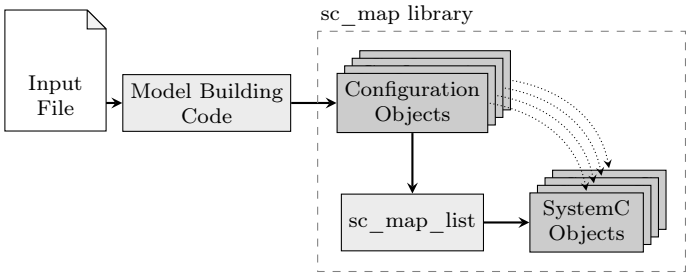


Figure 6.7.: Process of configuration of an sc_map structure out of a file. Source: [14].

It is advised by the SystemC standard to pass a unique name to each SystemC object. Therefore, every sc_map object has to take at least one configuration parameter. This parameter builds the base of the name that is passed to the constructor of every element in the sc_map container. To make the name unique, the key of the individual object is attached to the base name and then passed to the constructor.

Often, it is also necessary to configure more parameters of a module, e.g. the size of a memory element. For the configuration, so-called configuration classes are introduced. A configuration class has all configuration parameters of the corresponding SystemC module as member variables. As one configuration class could be a member of another configuration class, this principle facilitates to configure entire hierarchical systems. For example, the configuration object of the CMS track trigger PRM with sixteen AM Chips mounted on it consists of a variable for the number of memory chips and a vector of configuration objects for each AM Chip. The configuration objects in the vector then configure the individual AM Chips. By loading the configurations from files, it is possible to change parameters from outside the simulation without compiling the simulation again.

The configuration object is passed to the sc_map container as a parameter of the constructor. If a single configuration object is passed, the same configuration is assigned to all the elements. If a C++ STL vector or map of configuration objects are passed, the configuration objects are assigned to the SystemC objects depending on their order or their key.

6.3.6. The core of sc_map structures—sc_map_base

The parent class of all structures that are modeled by the sc_map library is `sc_map_base`, i.e. all classes that model structures must be derived from `sc_map_base`. Figure 6.8 shows the relation between all structures integrated into the sc_map library. This class contains both the container for the SystemC objects and the main logic behind sc_map. Two template arguments specialize the class: `object_T` defines the SystemC object that is stored in the container and `range_T` configures the structure in which the objects are arranged. The most important parts of `sc_map_base` are visualized in the UML diagram, Figure 6.9.

Internally, a C++ STL map container is used to map the keys to the corresponding SystemC objects. The SystemC objects itself are created dynamically in

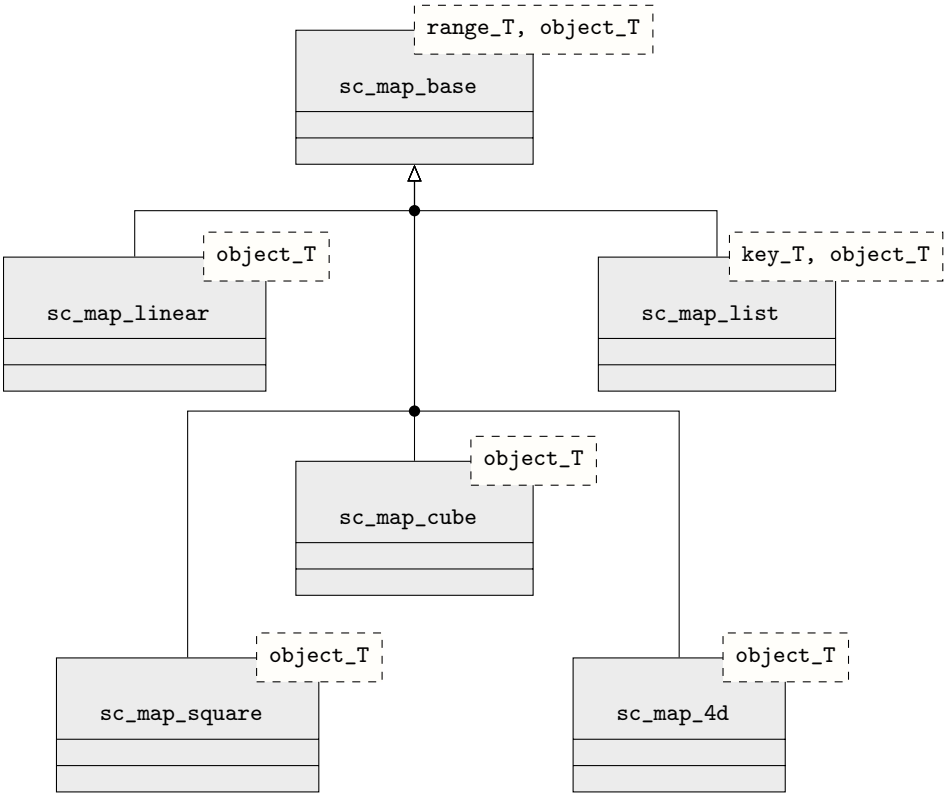


Figure 6.8.: The predefined types of sc_map structures available in the sc_map library. All of them are derived from `sc_map_base`.

the heap memory, and a pointer to them is stored in the map container together with the key to which the element belongs. By this arrangement, sc_map can be adapted very flexibly to any specific structure. Furthermore, it is memory effective as memory is only reserved for the created objects.

Beside the sc_map objects itself, the second central member variable is the range. Its type is defined by the `range_T` template argument and must be derived from the `sc_map_range` class (Section 6.3.8).

The functional part of `sc_map_base` can be divided into four parts: initializing the sc_map, accessing SystemC objects, binding signals and writing collectively to all signals. Basically, this is all the functionality that is necessary for an sc_map object. Usually, only the constructors specific to the structure must be implemented in derived structure classes. However, the constructors are mainly used to convert the structure-specific input parameters to a range, and

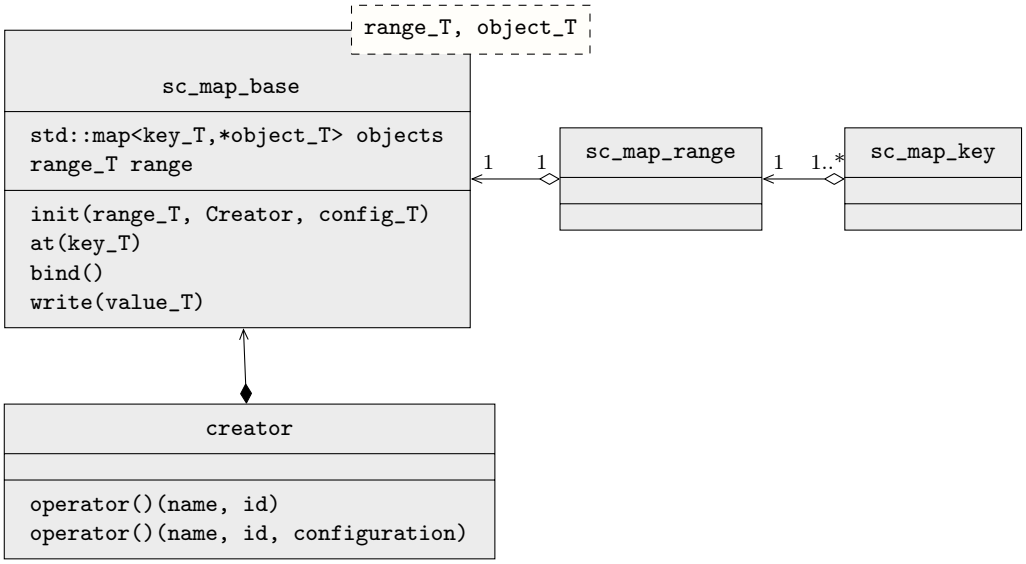


Figure 6.9.: UML diagram of `sc_map_base`. Source: [14].

the actual creation is performed by the init functions of `sc_map_base`. The creation process is described in detail in Section 6.3.5.

The function bind may only be applied to an sc_map object containing SystemC port objects. If this is not the case, a compilation error is thrown. The bind function takes either a single signal, an sc_map object with stored SystemC signals or an `sc_map_iterator` as an argument. In any of these cases, the signals passed to the object are bound to the ports of the sc_map object.

The execution of the write function is only possible on sc_map containers that contain `sc_signal` objects. Otherwise, a compilation error is thrown. With this function, the same value can be written to all signals within the sc_map.

The `sc_map_base` class also contains a creator class that is responsible for the creation of the individual objects within the sc_map container. This concept has been adapted from SystemC's sc_vector class [157]. The `sc_map_base` class provides a creator class that fits most of the cases. The provided creator class does three things: (1) generates the object in the heap memory, (2) generates and applies the name of the object, and (3) applies the configuration object if provided. However, if a custom creator class with extended functionalities is required, it can be passed to the constructor.

6.3.7. Implementation of keys

In an sc_map, a key unambiguously identifies a SystemC object within the container. The key is used for both accessing the SystemC objects as a user of the class and as the key for the C++ STL map container within a `sc_map_base` object.

All keys are derived from the abstract base class `sc_map_key`. Apart from that, the data type representing the key could be of any simple or compound data type. Figure 6.10 shows the implementation of the key used for `sc_map_square`, which has two class members one for each coordinate X and Y. In the case of regular keys, an intermediate inheritance layer exist—`sc_map_regular_key`, which groups all key classes that belong to regular structures.

To define its order among other keys, each key must provide functions to compare it with other keys of the same class. In a specific key class, only the *equality* and *smaller-than* functions have to be implemented, as the other comparison functions are derived from them. Additionally, a sub-class called `Comparator` is part of the `sc_map_key` class. Such a class is required by the C++ STL map container to place an object into it. This `Comparator` class must be derived from `std::binary_function`.

6.3.8. Implementation of ranges

For each type of structure, a corresponding range class must be implemented, which is derived from the templated abstract base class `sc_map_range`. The template parameter of the class configures the key type used to identify the arranged objects. A range class defines which keys are present in an sc_map object and as a consequence determines its size. It also implements functions to find the keys such as the first one, the last one or the next key after a specific key.

As the definitions of ranges may differ widely, the storage of associated keys and most functions are not implemented in the base class, but in the derived classes. Ranges may implement direction flags. These flags define if the keys or the coordinates are accessed upwards or downwards. The implementation of the range type for square structures is shown in Figure 6.11.

To have a regular structure defined, only the start and the end key need to be stored in the range class. These two keys are part of the `sc_map_range_regular` class. The `sc_map_range_regular` class is an intermediate inheritance layer

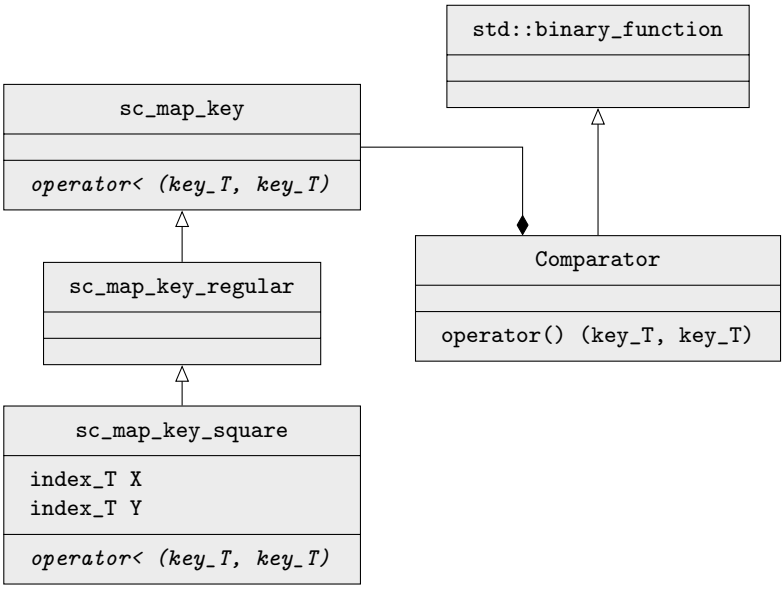


Figure 6.10.: Example implementation of a key: `sc_map_key_square`.

between the `sc_map_range` base class and the specific implementation of a regular structure. The calculation of the next key within the range is performed in the final class, in the example in `sc_map_square_range`.

In the case of list-like structures, the order of the elements is not given by the key itself. Therefore, a C++ STL vector in `sc_map_range_list` is used to put the keys into a certain order.

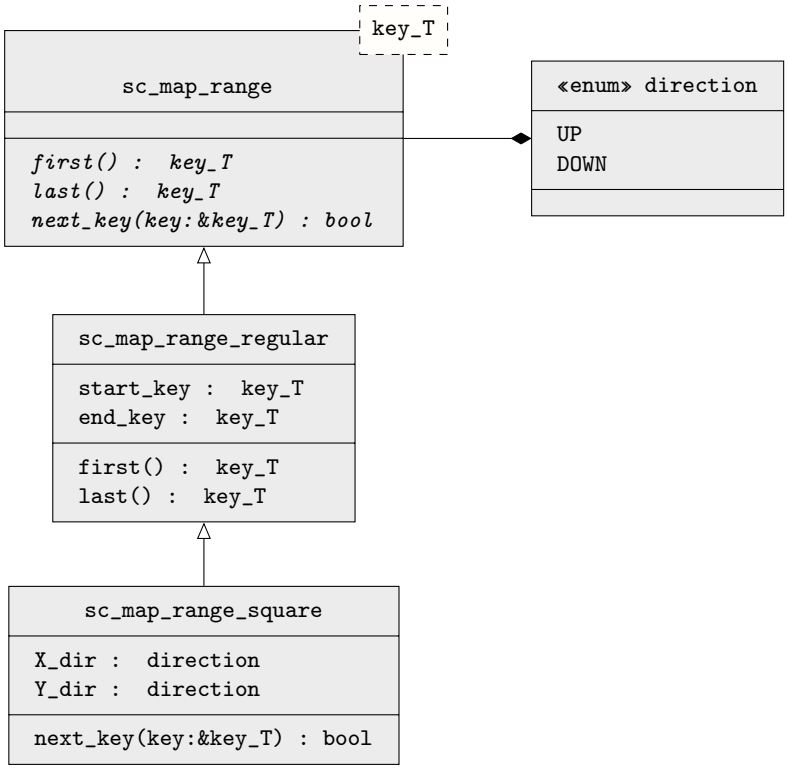


Figure 6.11.: Example implementation of a range: `sc_map_range_square`.

7. System Simulation of the CMS Track Trigger

This section describes the implementation of the system simulation of the AM approach CMS track trigger, as described in Section 5. In this context, the principles presented in Chapter 6 have been applied to the system simulation. The following sections focus mainly on the implementation details and the differences from the general description of the AM track trigger.

One sector of the CMS track trigger is implemented in the system simulation of which Figure 7.1 gives an overview. The simulation is split into an on-detector and an off-detector part. The on-detector part covers the systems that belong to the silicon tracker. The on-detector parts are included in the simulation to generate the data streams that constitute the input to the CMS track trigger. The track trigger processor including the DTCs build the off-detector part. In the off-detector, part the focus is put more on the functionality of the blocks.

The implementation as it is presented here is not complete because by now many details of the CMS track trigger are not determined. While the implementation of most blocks is roughly defined, an accurate estimate for the latencies of the single blocks is not available. Hence, the concept and the basic structure are available; details can be added to the system simulation easily.

7.1. Configuration of the simulation model

The system simulation is configured according to the concept of configuration objects described in Section 6.3.5. The object `track_trigger_config` builds the root of the configuration hierarchy. Within this object, the complete configuration of the simulated track trigger is contained as values and sub-configuration objects. Figure 7.2 shows a part of the hierarchy of the configuration objects. The main configuration object is passed to the SystemC module that is the top module of the track trigger. The specific configuration objects are then passed down the hierarchy.

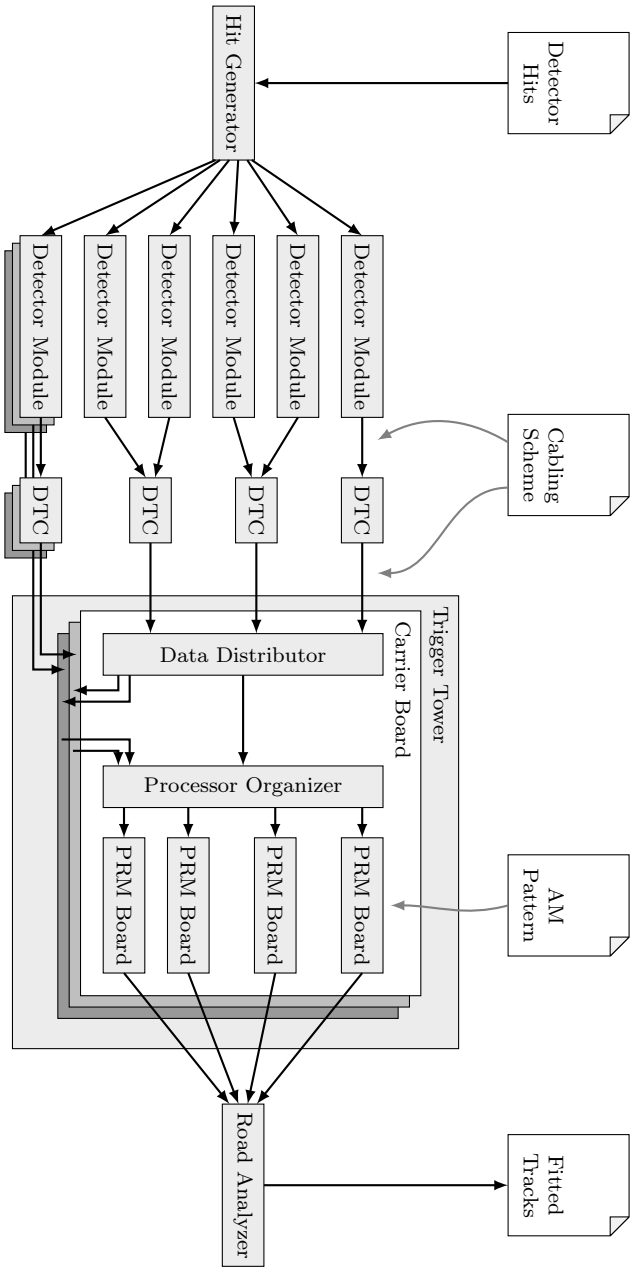


Figure 7.1.: Data flow of the CMS track trigger as it is implemented in the system simulation.

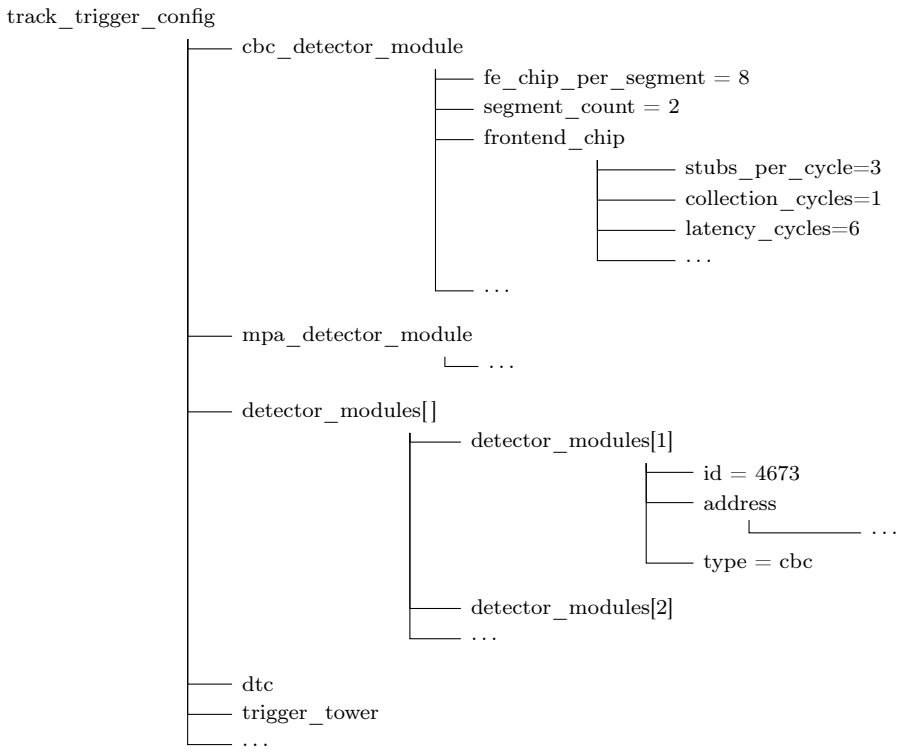


Figure 7.2.: Part of the configuration hierarchy of the system simulation.

Standard settings based on the currently proposed CMS track trigger design are contained in a so-called baseline configuration. A new configuration for a track trigger architecture can be based on this baseline configuration to simplify the generation of the new configuration.

Parts of the configuration are read from files. One file defines which detector modules are part of the simulated sector. Furthermore, it contains the coordinates, types and the number of the DTC to which they are connected. Two more files configure the DTCs and the crates, and define their connections.

7.2. Input data

The input data of the simulation is generated by specific modules, the hit generator. It accepts American Standard Code for Information Interchange (ASCII) files that are human readable and, thus, can also be written by hand

for debugging purposes. However, the input files are usually generated within CMSSW so that the simulation runs with realistic detector data. The format of the input files reflects the data that are assigned to a stub object in CMSSW. It contains event-specific information such as the event number and the stub ID. Furthermore, all information about the detector module is stored in the input file: type (2S or PS), location (barrel or endcap), the module coordinates (layer, ladder[1] and module number). Finally, the internal coordinates of a module are given: z-segment, front-end chip number, strip address, pixel address and the bend. Listing 7.1 shows an excerpt of an input file.

The hit generator reads all the stubs of one event from the input file. The stub address is then checked whether the module, to which the stub actually belongs, exists within the simulation. If the stub does not exist, it is discarded. Otherwise, the stub is pushed to the First-In, First-Out (FIFO) buffer of the readout chip by which the stub is generated. In the simulation, there is a FIFO SystemC signal for each readout chip of the detector modules. After reading a complete event, the next event is processed.

7.3. Modeling of the CMS track trigger components

7.3.1. Detector modules

The purpose of the detector modules within the simulation is mainly the generation of the input streams to the CMS track trigger processor. Therefore, many details of the detector module elements are not included in the simulation model. The model of the detector module contains three elements of the CMS tracker modules: the readout chips (CBC or MPA), the data concentrator (CIC) and the GBT transceiver chip. Its composition is shown in Figure 7.3.

Only one simulation model for both the 2S and PS modules exist. Their specific behavior can be configured by the parameters passed to the detector module model. As the input data from CMSSW already provide the stub address and not the signals from the silicon detectors, many processing steps are omitted in the model, e.g. the cluster and the stub building. The remaining functionality is the selection of the stubs if more stubs are provided by the data generator than the ones that can be transmitted to the data concentrator. Also, the spreading

[1] The ladder corresponds to the φ-coordinate of the module. In the barrel section, the detector modules are mounted on rods—called ladder—which are arranged cylindrically and build the layers in this way.

Listing 7.1: Excerpt of an input file to the system simulation.

# Evnt	S_ID	P	B	E	Ly	Ld	Mo	S	C	St	Px	Bd
1	1	1	1	0	5	14	18	1	6	234	7	0
1	2	1	1	0	6	20	17	0	1	166	10	−1
1	3	1	1	0	7	29	18	0	5	188	6	1
1	4	0	1	0	8	41	18	1	7	151	0	1
1	5	0	1	0	9	52	20	1	1	96	0	0
1	6	0	1	0	10	64	21	0	3	27	0	1
2	1	1	1	0	5	2	14	0	3	35	6	0
2	2	1	1	0	6	3	11	1	2	205	11	0
2	3	0	1	0	8	6	8	1	2	69	0	0
2	4	0	1	0	9	8	7	1	3	233	0	0
2	5	0	1	0	10	10	6	1	5	167	0	1

of the stubs over two time slots, as used by the PS modules, is implemented in the readout chip model. The readout chips are arranged in a 2×8 array (z-segments $\times$ chip number), which corresponds to the arrangement on the real detector modules.

The data from all the readout chips of one of the two z-segments are transferred to one data concentrator. As described in Section 3.5.1, the data concentrator selects a number of stubs, 12 for 2S modules and 10 for the PS modules, with the highest transverse momentum within eight clock cycles. The GBT model then merges the two data streams from the data concentrators into a single one.

7.3.2. Data Trigger Control board (DTC)

As the functionality of the DTC in the AM approach is not exactly defined yet, the model of the DTC implements only the functions that are defined as the minimum requirement. The DTC receives the stubs from multiple detector modules and converts them into the data format used by the trigger towers and merges the stubs from all detector modules into a single frame. Up to the DTC, the module number and the z-segment are coded by the specific physical link on which the stub arrive and the number of the stream on the link. As the stubs are merged, and this information would be lost, the DTC attaches a unique module ID and a bit to indicate from which z-segment it originates to each stub.

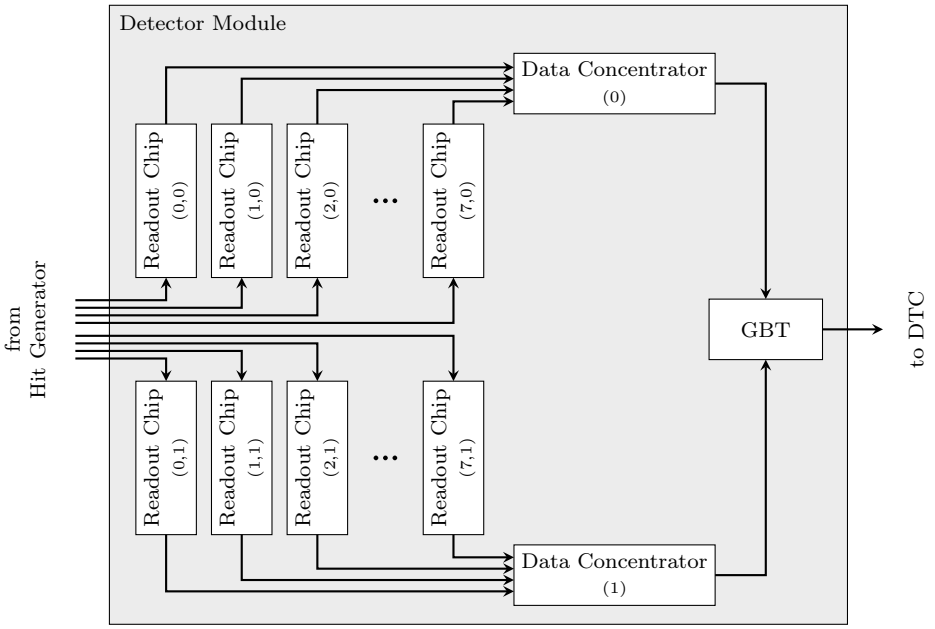


Figure 7.3.: The detector module structure in the system simulation.

7.3.3. Data distribution in the trigger tower

The data organizer is implemented in the FPGA of each carrier board within a trigger tower. Its tasks are to collect and distribute the incoming data and implement the time multiplexing. While it is described as a single part in the general description of the CMS track trigger (Section 5.1), it is split into two parts in the system simulation. The first part is the data distributor while the second part is the processor organizer.

The data distributor receives the data from the DTCs. As the module number is added to the stubs at the DTCs, the number of the DTC and the number of the carrier blade on which the stubs arrive has to be added to the stubs in the data distributor. The stubs are then ordered by the relative bunch crossing number assigned by the data concentrator while spreading the stubs over eight clock cycles. The stubs from the same bunch crossing are packed into a frame and are transmitted to the carrier board, which has the PRM attached that processes this event. This is the first part of the time multiplexing.

The processor organizer is implemented in the same FPGAand realizes the second part of the time multiplexing. It gathers all the stubs that are sent from

the other carrier boards to be processed by one of the PRMs attached to the carrier board of the processor organizer. The stubs are buffered in separate FIFOs one for each detector layer that is processed by this trigger tower. From these FIFOs, the processor organizer reads the stubs, converts their format to the one used at the PRMs and sends the data stream to the PRM that processes the current event.

7.3.4. PRM board

The implemented data flow is shown in Figure 7.4. The model of the PRM board follows the description of the CMS track trigger processing described in Section 5.2. However, the partitioning into functional blocks differs slightly. For instance, the IDB is split into three different blocks to reflect the data flow in the system better. Furthermore, the models of the AM Chip and the superstrip lookup share the pattern bank data. Currently, the track candidate builder and the track fitter are not implemented in the system simulation. As they are just following each other in the data flow, they do not affect the system architecture except for some latency that must be added for these two blocks.

The IDB is split into the three parts: (1) the hit processor that writes the hits to the buffer and the AM Chips, (2) the storage (hit buffer) itself and (3) the road processor that reads back the hits that belong to a road. The hit processor just generates the superstrips from the hits and sends the superstrips to the AM Chip Both the superstrip and the truncated part of the hit are also sent to the hit buffer. The hit buffer is the storage of the hits and provides a write interface towards the hit processor and a read interface towards the road processor. The storage itself is realized as a map container with lists stored in it. Each list stores the truncated part of the full-resolution hits. Each list is linked to a superstrip that is the key of the map, i.e. the list of the hits can be accessed by providing the superstrip to the hit buffer. This is exactly what the road processor does. It sends the superstrip retrieved from the superstrip lookup module and then combines the superstrip with the truncated parts of the hits stored for this superstrip.

The pattern bank is a module that does not exist in the real CMS track trigger where it is part of both the AM Chip and the superstrip lookup. As both the AM Chip and the superstrip lookup contain the same patterns, which are just accessed differently, this part has been extracted from those modules. Therefore, the patterns are managed at a central module, and the simulation needs less memory. The pattern bank includes multiple map containers that allow to

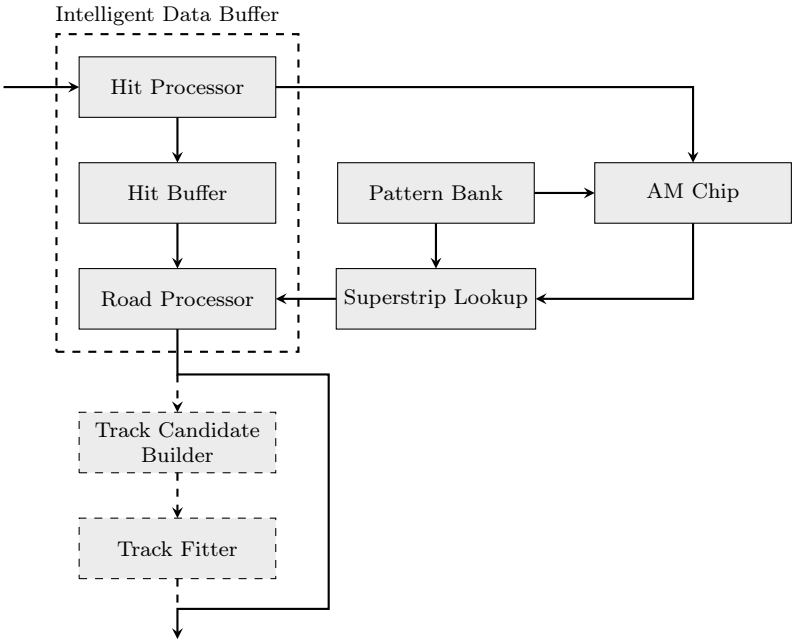


Figure 7.4.: Structure of the PRM board model in the system simulation.

access it either by the road number or a the superstrip value of one layer of a pattern. If the road number is applied, the pattern bank returns the whole pattern. If a superstrip together with a layer is applied, the pattern bank returns the numbers of all the roads that contain this superstrip on this layer. A very important function is the initialization of the pattern bank. A pattern bank can be loaded from either ASCII or binary files. Additionally, the pattern bank module also provides the possibility to generate banks that are well suited for debugging, e.g. a bank that contains only straight paths through the detector.

The AM Chip model takes the superstrips sent by the hit processor and applies them to the pattern bank. The layer is used as the key to access the values in the pattern bank. The pattern numbers returned from the pattern bank are used to fill out a scoreboard in the AM Chip. The scoreboard is a table of binary values where it can be marked whether a superstrip has been found within a pattern on the corresponding layer. In the AM Chip model, this scoreboard is generated dynamically, and only entries exist for patterns in which at least one superstrip has been found. After all the superstrips of an event has been processed, the scoreboard is checked for patterns that have all six, or less if a

different threshold is configured, superstrips activated. The number of these patterns are then sent ahead as the road number.

Most of the functionality of the superstrip lookup model is already implemented in the pattern bank module. Therefore, the superstrip lookup model is basically a wrapper to extract the road number from the data stream and create new streams from the retrieved superstrips.

7.4. Recording of results

Different tools are included into the system simulation to store the calculated results of the simulation and to monitor the simulation process.

The SystemC module road analyzer records the results of the simulated CMS track trigger. In the simulation without the track candidate builder and the track fitter, the recorded results are the hits provided by the road processor. To collect all the hits, the road analyzer is connected to each PRM module. The found hits are stored together with the time into a human-readable ASCII file. Additionally, the road processor is also connected to the hit generator, which provides the number of hits read from the input file and how many hits had to be discarded as they belong to detector modules not included in the simulated CMS track trigger model. At the end of execution, the road analyzer prints a short report with the most important numbers to the console.

Likewise, a report with the execution times of the simulation is generated and printed to the console. Besides the total execution time of the simulation, also the time the building of the model (elaboration) and the time spent in the simulation kernel are reported, for details see Section 6.1.2.

7.5. Results

The system configuration that was used to test the performance of the system simulation corresponds roughly to one trigger tower. However, the configuration of a real trigger tower was not available. Thus, a CMS track trigger configuration has manually been generated whose size and architecture correspond to a possible trigger tower architecture. The configuration of the different elements of the system simulation is based on the currently discussed system.

7.5.1. Size of the system simulation

A diagram showing the structure of the simulation is shown in Figure 7.5. The simulated detector consists of 500 detector modules, 300 for the simulated sector and 200 additional ones from the neighboring sectors. As every DTC could receive data from up to 72 detector modules, seven DTCs would be sufficient in the ideal case. In reality, the data of one trigger tower originates from more DTCs. For the used system, a total of 15 DTCs has been assumed, and roughly the same number of detector modules is connected to each of them. One trigger tower processor contains ten carrier blades with two PRMs connected to each of them. The data from the DTCs is spread to all the carrier boards.

To model the regular structures of the CMS track trigger, the sc_map library has been heavily used in the system simulation. To evaluate the usage of the sc_map library, some code has been included in the simulation that allows the counting of the elements within the system simulation. During performance measurements, this code is deactivated. Table 7.1 shows the number of sc_map containers that are instantiated during the model building of the CMS track trigger simulation. Roughly 18 000 containers organize more than 145 000 SystemC objects of the system simulation. The SystemC objects organized in the containers are modules, signals and ports. Table 7.2 lists how many objects of the different SystemC objects are stored within the sc_map containers.

7.5.2. Execution time

The measurement of the execution time of the system simulation has been performed with the system, as described in the previous section. However, all

Table 7.1.: The number of sc_map containers used in the system simulation of the CMS track trigger. Listed by the type of the sc_map container.

scmap Class	Number
sc_map_linear	14418
sc_map_square	2700
sc_map_cube	784
sc_map_4d	1
sc_map_list	17
Total	17920

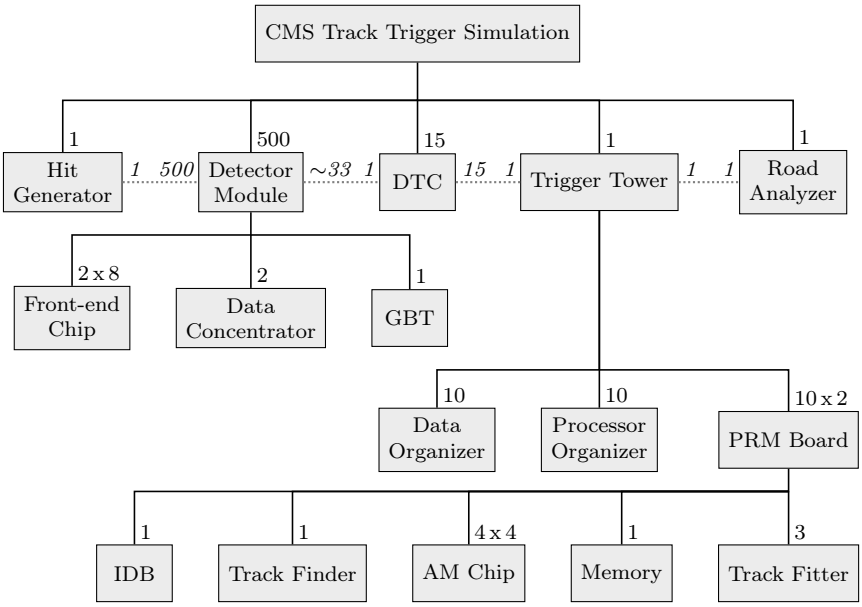


Figure 7.5.: Structure and size of the simulated CMS track trigger.

the code that monitors the simulation, such as the code counting the sc_map objects and logging, has been deactivated. The tests were run on a computer with an Intel Core i7-3777 @ 4×3.4 GHz CPU and 12 GB RAM.

The data set used in the test consists of ten events with 300 hits, which are distributed all over the layers. The hits are chosen as such that they activate patterns in the AM Chips. The hits belonging to this patterns are stored in the output file. This setting emulates a realistic situation similar to the conditions at the final CMS track trigger. The simulation of the CMS track trigger has been running for 5000 ns. As not all the latencies are included to the model yet, these 5000 ns are enough to process all the ten events. The simulation has been run for ten times and the time model building time and the simulation time has been measured.

The time for building the model of the simulation has been measured to be 3.74 s and the time to simulate the model with 4.84 s. The track candidate builder and the track fitter are not yet part of the simulation. Nevertheless, the CMS track trigger can be simulated within a reasonable amount of time. The simulation time is short enough that it is possible to run the system simulation several times with different configurations and, thus, evaluate different system architectures in a short time.

Table 7.2.: SystemC elements that are organized within sc_map containers.

	SystemC class	Number
SystemC Modules:	sc_module	8031
SystemC Signals:	sc_signal	1440
	sc_buffer	32245
	sc_fifo	12372
SystemC Ports:	sc_in	34445
	sc_out	32672
	sc_fifo_in	12382
	sc_fifo_out	12252
Total		145839

7.5.3. Latency of the AM-based track trigger

As not all the parts of the CMS track trigger are implemented or even defined yet, not an exact statement about the latency can be made. However, a rough estimate of the system latency shall be given here as it is a very important requirement of the CMS track trigger. The values presented here are not a result of the system simulation. The list of latencies in Table 7.3 shall give an idea of which values are already known and which parts may be problematic. The values in the table are partially collected from research groups working on parts of the CMS track trigger and, the other values are rough estimates on the base of the actual information on the CMS track trigger.

The measured latencies of the carrier board are based on the implementation of the DO by Fermilab [158]. The current implementation of the PRM firmware provides the latencies from the AM Chip to the track fitter in the "Measured" column of the table [159]. The PRM latencies in the "Estimated" column base on the expected clock frequency with which these blocks will run after optimizations.

All the other values in Table 7.3 are estimates based on the information available on the individual parts of the CMS track trigger. They are written in italic numbers. The latency of the detector modules is caused to a large extent by the accumulation of stubs over eight clock cycles. This collection results in a latency of 200 ns (8×25 ns). Another 30 clock cycles at 160 MHz are added for the stub building and processing in the front-end chips and the processing in the DC. The optical transmission by the GBT system is mainly dominated by the

Table 7.3.: Latencies of the different parts of the CMS track trigger.

	Latency	
	Measured (ns)	Estimated (ns)
Processing on Detector Module		*387.5*
GBT Transfer (100 m)		*600.0*
DTC Processing		*250.0*
DTC → RTM (20 m)		*100.0*
RTM → DO → backplane	600.0	600.0
Backplane → DO → PRM	500.0	500.0
PRM Input → IDB → AM Chip		*1 050.0*
AM Chip		*80.0*
AM Chip to IDB	202.6	100.0
Intelligent Data Buffer	219.9	112.5
IDB to TCB	40.0	26.4
Track Candidate Builder (TCB)	589.9	389.4
TCB → TF	38.8	30.0
Track Fitter (TF)	117.5	94.0
PRM → DO		*100.0*
DO → Gateway Board		*100.0*
Gateway Board → L1 trigger (20 m)		*200.0*
Total		4 719.8

propagation delay of the signal from the detector to the counting room. A cable length of 100 m is assumed which results in a latency of 500 ns. By taking into account the error correction and the serialization, a total transmission latency of 600 ns is assumed.

The amount of data processing at the DTC is low for the AM approach. However, the packet format is changed. Thus, the whole packet is first received and then sent ahead to the trigger tower processors. Receiving the whole packet from the detector modules takes 200 ns. Another 50 ns are added for the other processing tasks. The transmission on the optical link from the DTC to the trigger tower processor takes another 200 ns. The receiving, sorting and distribution to the PRMs have been evaluated. The total latency is 1100 ns.

On the PRM, the hits are transmitted via the IDB more or less directly to the AM Chips. The IDB only generates the superstrips before forwarding the data to the AM Chips and buffers the hits. However, this can run in parallel with other operations. A latency of 50 ns is assumed for this operation. The writing of the superstrips to the AM Chips takes some time as all the hits have to be written to the AM Chips before the processing can start. The AM Chip runs at 100 MHz and up to 100 superstrips may be written on a single layer, the resulting latency is 1000 ns. The latency of the remaining steps on the PRM has been measured on the prototype of the PRM.

After the track fitting, the track parameters need to be transmitted to the L1 trigger. The protocols on this path are not defined yet. It is assumed that the track parameters are streamed and add for each step on the way to the L1 trigger 100 ns. For the link from the crate to the L1 trigger another, 100 ns are added for the propagation delay.

Summing up all these latencies, the total latency from the detection of the event until the first track reaches the L1 trigger is estimated to be 4719 ns. Thus, the estimated latency of the currently proposed CMS track trigger stays below the maximum 5 µs, which are assigned to the track trigger [5].

8. Cost Estimate of the CMS Track Trigger

In the technical proposal for the Phase-II upgrade [5], the costs for the back-end electronics of the silicon outer tracker has been indicated with 11.9 MEUR.[1] An independent study of the costs for an AM-based CMS track trigger has been performed, and the results are presented in this chapter. In a second part, the costs are extrapolated to the year 2022, when the CMS track trigger will be built. The difficulties of such an extrapolation are the open questions about the design of the boards and the progress of the FPGA technology.

8.1. Constraints of the cost estimate

The cost estimate is based on the CMS track trigger system foreseen for the review in December 2016, as described in the Sections 5.3 and 5.4. Therefore, the estimated costs are the costs necessary to build the CMS track trigger today. The cost estimate includes all the parts that directly belong to the CMS track trigger: from the cables that connect the CMS track trigger with the DTCs to the cables that transfer the results to the L1 trigger.

The prices of the components were collected in the beginning of June 2016 and are listed in Appendix D. For the electronics components, the prices were taken from digikey.de [160]. Where possible, the price for 100 pieces of a component has been applied to the cost estimate. Unfortunately, this is not possible for the prices of up-to-date FPGAs for which realistic prices are only accessible by requesting an offer. The FPGA prices in this estimate are based on the prices from digikey.de and can be seen as a conservative estimate. For the prices of the PCBs, the on-line calculation tool of Würth Elektronik [161] has been used. The options for complex boards have been activated and a long delivery time has been chosen. On the Würth Elektronik website, the price can only be calculated for orders up to 16 PCBs. For the cost estimate, the price for 16

[1] 12.4 MCHF at an exchange rate EUR to CHF, 1. June 2015 of: 1.04

pieces has been chosen. However, the price will be lower for the large number of PCBs needed for the CMS track trigger.

8.2. Crates

As described in Section 5.1, the silicon tracker is split into 48 trigger towers. It is expected that one crate is able to process the data of at least one trigger tower. Consequently, a total of 48 ATCA crates is necessary. Most probably, it will be possible to process two trigger towers within one crate. The technological progress may even allow four trigger towers to be processed within one crate. The number of crates would be reduced to 24 or 12 accordingly.

ATCA crates have up to 14 slots for carrier blades. Not all the carrier blades can be used for the data processing, some blades function as gateways for the data transmission to the L1 trigger. The layout that has been discussed in the CMS track trigger community foresees that a crate contains ten processing blades with the PRMs. Thus, four slots remain for the gateway blades. Table 8.1 shows the total of blades needed depending on the number of crates in the system. For the four trigger towers in one crate option, the number of blades is distributed differently: Twelve processing blades and two gateway blades are installed in each crate.

A quote by Pentair from May 2014 [162], states the price of a single ATCA crate at about 6.5 kEUR. This crate is fully equipped with the full-mesh backplane, the power supply and two shelf managers for redundancy.

Table 8.1.: Number of blades depending on the number of sectors processed by one crate.

Trigger Towers per Crate	Crates	Gateway Blades	Computation Blades
1	48	192	480
2	24	96	240
4	12	24	144

8.3. Carrier blades

The cost calculation of the carrier blade is based on the Pulsar IIb blade from Fermilab described in Section 5.3.2. The Pulsar IIb is used as the computation blade in the current design of the AM-based track trigger. As plans for the gateway blade do not yet exist, a blade similar to the Pulsar IIb is assumed and, thus, the costs are the same for both blades. The difference is that no PRMs are plugged to the gateway blades.

A carrier blade is a large PCB, which is according to the ATCA standard 280 mm deep and 322 mm high. Due to the high complexity of the blade, a 16-layer PCB is needed, which costs at this size around 350 EUR. The central part of the blade is a large FPGA. All the communication, from the RTM, between the blades through the backplane and to the processing mezzanines, goes through this FPGA. Thus, an FPGA with enough high-speed transceivers is necessary. The price of the Xilinx Virtex-7 (XC7VX690T-2FFG1927C) mounted to the Pulsar IIb blade is 8072 EUR.

The other main parts on the blades are a board controller,power supply components and the connectors to the backplane, the RTM and the mezzanines. Altogether, the costs of the remaining electronics are estimated to be 1000 EUR.

Table 8.2 lists the components of the carrier blade and the total estimated costs that are around 9450 EUR.

Table 8.2.: Costs of a carrier blade.

Part	Quantity	Price (EUR)
PCB	1	350
FPGA	1	8100
Other parts	1	1000
Total		9450

8.4. Pattern recognition mezzanine (PRM)

The calculation base for the cost of the PRM is the PRM06, as described in Section 5.4.3. However, the goal is to design a PRM with the capacity of 2 million patterns. Therefore, the cost is calculated for a PRM with 16 AM06 chips.

8.4.1. AM chip

As the AM Chip is an ASIC (see Section 5.4.1), the cost of it depends to a large extent on the initial production costs and less on the total number of chips needed. Therefore, the calculation for the price starts from a system-wide view and the price of a single chip is derived from top to bottom. The current generation of the AM Chip (AM06) has been developed by INFN in Pisa and stores up to 128 000 patterns. The chip is produced in a 65 nm process and occupies an area of 150 mm^2. The following calculations are based on these values. Furthermore, the presented value covers just the production costs and not any costs for prototypes, IP blocks, etc.

In the calculations here, a PRM with 16 AM Chips is assumed. In the case of a system with 48 crates and a 20-fold time multiplexing, 20 PRMs are necessary for one trigger tower. Thus, a total of 320 AM Chips are necessary for one trigger tower and 15 360 AM Chips for the complete CMS track trigger.

With such a high number of chips, an own engineering run pays off. An engineering run in the 65 nm Taiwan Semiconductor Manufacturing Company (TSMC) technology, in which the AM Chip is designed, costs around 800 kEUR. Each additional wafer costs approximately 10 kEUR. The diameter of the wafers is 300 mm, and around 400 AM Chips are produced out of one. Consequently, 39 wafers must be processed to get the 15 360 AM Chips for the full CMS track trigger. The costs for one AM Chip c_{AM_chip} is calculated by

$$c_{AM\_chip} = \frac{c_{initial} + n_{wafer} \cdot c_{wafer}}{n_{chip}} \tag{8.1}$$

where $c_{initial}$ are the initial costs that the production of the ASIC costs—in this case the engineering run, n_{wafer} is the number of the produced wafers, c_{wafer} are the costs of a single wafer and n_{chip} the number of AM Chips that

are necessary. By filling the numbers into the equation, the price for a single AM Chip is calculated with about 77 EUR.

$$\frac{800\,\text{kEUR} + 39 \cdot 10\,\text{kEUR}}{15360} = 77.47\,\text{EUR} \tag{8.2}$$

The packaging of ASICs cannot be neglected; it is expected that it costs around 20 EUR per chip. The total costs of one AM Chip are about 100 EUR.

8.4.2. Total costs of the PRM

The FPGA mounted on the PRM06 is a Xilinx Kintex Ultrascale XCKU060 with 726 000 logic cells. This FPGA is not sold yet by digikey.com, the price from avnet.com (USA) [163] is 2363 EUR[2].

The design of the mezzanine bases on the double-wide FMC standard, i.e. two FMC connectors connect the mezzanine to the carrier blade. The PCB itself is slightly longer than defined by the FMC standard and has a size of 149 mm × 149 mm. The PCB is fully packed and has 14 layers. The cost for one PCB is about 190 EUR.

The mezzanine is also equipped with two RLDRAM, a flash memory for the FPGA configuration, power supply components and other small parts. For those parts, another 200 EUR are added to the cost of the mezzanine.

The compilation of all the costs of the PRM is presented in Table 8.3. The total costs of one PRM are about 4350 EUR.

Table 8.3.: Costs of PRM.

Part	Quantity	Price (EUR)
PCB	1	190
FPGA	1	2360
AM Chip	16	1600
Other parts		200
Total		4350

[2] 2624 USD at an exchange rate EUR to USD, 1. June 2016 of: 1.11

8.5. Optical communication

Both the optical links from the DTCs to the track trigger processor and the optical links from the track trigger processor to the global L1 trigger are added to the costs of the CMS track trigger. For the current test stands, QSFP+ cables at a bandwidth of 40 Gbit/s are used for both of them. Most likely, a newer standard with a higher bandwidth will be available for the final system.

The data from the 8424 2S modules are transmitted at 4.5 Gbit/s. From roughly one-third of the 5748 PS modules, the data are transmitted at a data rate of 9.1 Gbit/s. The remaining PS modules transmit data at 4.5 Gbit/s. The total input bandwidth of the connected fibers is approximately 70 Tbit/s. One fifth of the bandwidth is used for raw data. Thus, 56 Tbit/s of trigger data remain. The data from roughly every second sensor is processed by two trigger towers. As the data are duplicated at the DTCs, the total input data to the CMS track trigger is increased by 50 percent. The resulting input bandwidth of the CMS track trigger is about 85 Tbit/s. Assuming an even distribution of the data on the links, 2175 QSFP+ links are needed between the DTCs and the crates of the CMS track trigger. The number would be increased by inefficiencies of the cabling. However, not the full bandwidth is necessary for each event from each sector. It is therefore assumed that the calculated number is roughly the final number of links.

The computed tracks are transmitted to the L1 trigger after the processing by the CMS track trigger. About 125 tracks are found each event. The data set of a track consists of four parameters and four errors. We assume the accuracy of the parameters is 32 bit and, thus, the total output bandwidth to the L1 trigger is 1.3 Tbit/s or 32.5 Gbit/s per tower. To be on the safe side, two QSFP+ cables per trigger tower are included in the cost estimate. The total number of QSFP+ cables that is needed for the output data is therefore 96.

Summing up, 2271 QSFP+ cables are needed for the whole CMS track trigger. The unit price for a QSFP+ cable of 20 m length is 260 EUR. Thus, the cost for all cables is 590 kEUR.

8.6. Rear transmission module (RTM)

The RTM is a long but narrow board that provides the communication infrastructure to the carrier blades. The calculations here are based on the RTM

developed for the Pulsar IIb blade. It provides ten QSFP+ cages, each of them can take a transceiver module capable of transferring 40 Gbit/s. Thereby, the total bandwidth of one RTM is 400 Gbit/s.

The RTM consists only of the PCB, the QSFP+ cages and connectors, the connector towards the blade and few components for controlling the RTM. The PCB with the dimensions 139 mm × 76.5 mm costs about 150 EUR. The QSFP+ cages only provide the mechanical connection, and the transceivers are part of the cable. Hence, the price is roughly 15 EUR per QSFP+ connection. Table 8.4 lists the costs of the components. The total cost of one board is 500 EUR.

8.7. Total costs for the CMS track trigger

The estimated costs of all the individual components for one crate are compiled in Table 8.5. Summing up all the parts, total costs of 232 kEUR results for one crate.

As compiled in Table 8.6, the total costs for the 48 crates, which would be necessary if the CMS track trigger would be built today, would be 11.7 MEUR. Besides the fully populated crates, these costs also include the costs for the optical fibers with the transceivers. Compared with the budgeted costs of 11.9 MEUR, the estimated costs are roughly the same. However, a reduction of the costs can be expected due to the large number of parts that will be ordered for the CMS track trigger.

Table 8.4.: Costs of RTM.

Part	Quantity	Price (EUR)
PCB	1	200
QSFP+ cage	10	150
Other parts		150
Total		500

Table 8.5.: Total costs of a populated crate.

Part	Quantity	Price (kEUR)
ATCA crate	1	7
ATCA blade	14	132
AM board	20	87
RTM	14	6
Total		232

Table 8.6.: Costs of the complete CMS track trigger system.

Part	Quantity	Price (MEUR)
Crate	48	11.1
QSFP+ cable	2271	0.6
Total		11.7

8.8. Extrapolation to 2022

The same system built in 2022 would be cheaper as the prices for electronic components decrease due to the technical progress. However, the technical progress can also be used to reduce the system size and reduce the costs further.

8.8.1. Extrapolation of FPGA technology

As the central and also most expensive component, the development of FPGAs influences the composition of the final CMS track trigger. Figure 8.1 shows the evolution of three key features of FPGAs: the amount of logic, the memory size and the total bandwidth of the high-speed transceivers. The figure is limited to FPGAs from Xilinx. The models chosen are the devices with the most logic units from the corresponding generation. For the Virtex-5, Virtex-6 and UltraSCALE, the devices with the highest transceiver bandwidth have also been included. The date is chosen according to the announcement of the device by Xilinx [164].

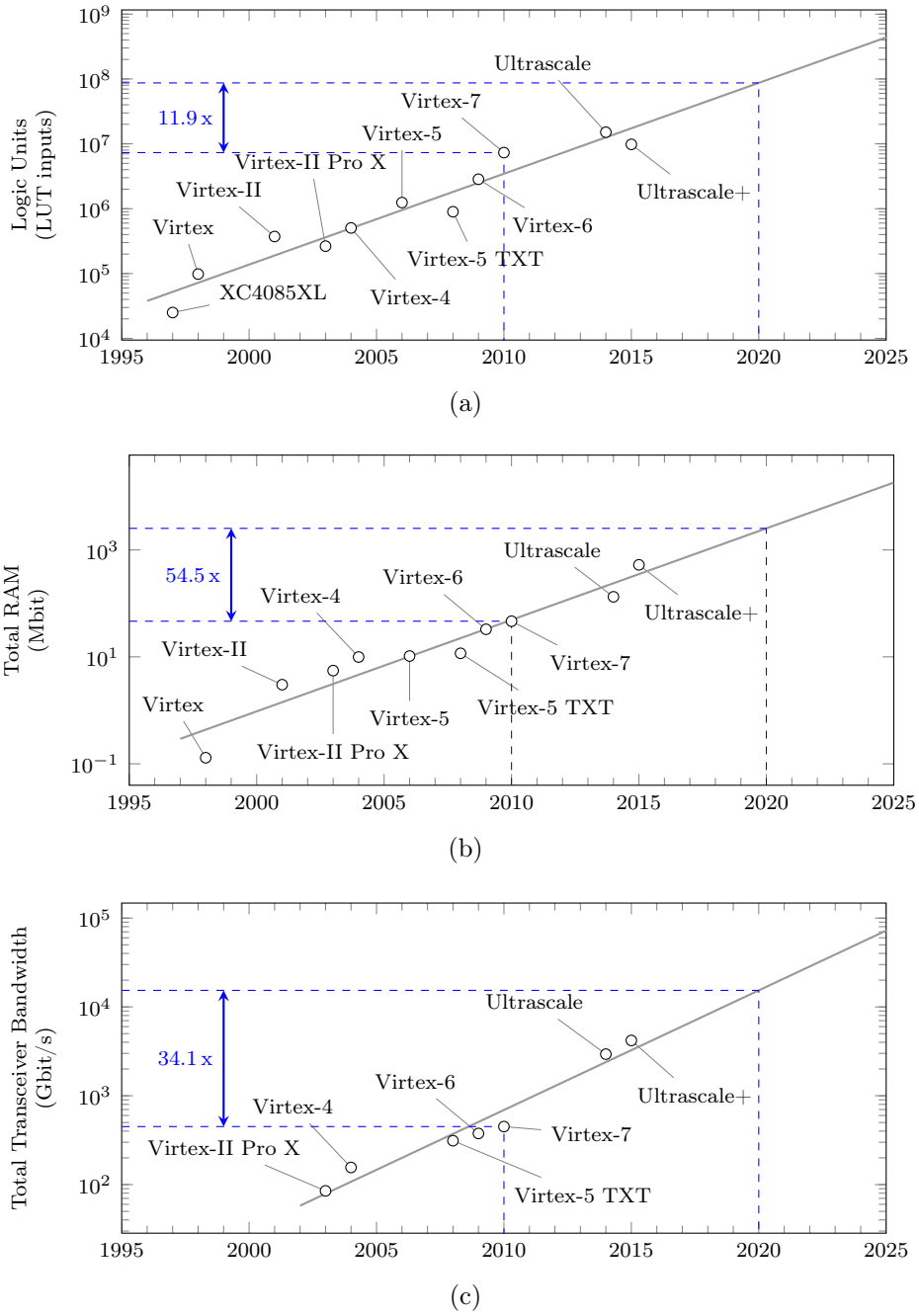


Figure 8.1.: Evolution of features over FPGAs Xilinx generations: (a) the evolution of logic in form of LUTs, (b) shows the size of all RAM resources and (c) the transmission bandwidth of the high-speed transceivers. The change of the values from the currently used Virtex-7 until 2020 is indicated in blue. Sources: [165–181].

The current versions of the hardware for the AM approach utilizes Xilinx FPGAs from generation 7, with the new developments shifting towards the UltraSCALE generation. The final decisions about the hardware for the CMS track trigger will be taken around 2020. Therefore, the expected performance of the FPGAs is extrapolated in Figure 8.1 to the year 2020. By having a look at the intervals between the past generation, one more generation can be expected to be announced and available until the CMS track trigger will be built.

Figure 8.1 (a) shows the evolution of the logic resources in the Xilinx FPGAs. The size of the LUTs in the FPGAs changed from the Virtex-4 to Virtex-5. Therefore, the comparison of the logic resources is based on the total number of LUT inputs that an FPGA provides, i.e. the number of LUTs multiplied by the number of inputs of one LUT. Comparing the logic resources of the Virtex-7 device with the extrapolated value in 2020, an increase by factor 12 is predicted. Taking into account that it is feasible to build the track trigger by today's technology this provides some room for improvements.

Figure 8.1 (b) shows the evolution of the total RAM resources within the Xilinx FPGAs. With the introduction of the UltraSCALE+ generation, Xilinx introduced a new feature the UltraRAM [134]. These are large RAM blocks that are intended to replace external memories. In the graph, the size of the UltraRAM is added to the block RAM resources, which the FPGAs provide for several generations. The integration of the UltraRAM can also be seen in the graph, where the last point in the data series moves up quite a bit compared to the previous point. The prediction indicates that Xilinx FPGAs will contain more than 2 Gbit of RAM by 2020, which is a factor of 55 more than the currently used Virtex-7 FPGAs. Although this value might be too high, 1 Gbit seems to be realistic considering the currently announced UltraSCALE+ FPGAs contain more than 500 Mbit of RAM.

The third investigated feature of FPGA is the bandwidth of the high-speed transceivers of the FPGAs, shown in Figure 8.1 (c). The bandwidth is predicted to increase by a factor of 34 from the currently used Virtex-7 until the year 2020. The input bandwidth of a single FPGA would thus be enough to read the data of roughly a fifth of the CMS tracker. This does not mean that all these data could also be processed by a single FPGA. Another question is the number of high-speed transceivers. For the use within the CMS track trigger, transceivers may not necessarily be the fastest, but it is important that many transceivers be available.

8.8.2. Costs of the hardware platform in 2022

The described system for the AM approach (Section 5.3) and the cost estimate previously in this chapter are based on the system as it is developed for the December review. It is expected that this system will demonstrate the feasibility to build the CMS track trigger with today's technologies. However, the technology will progress until the final CMS track trigger is built around 2022. The additional computing power and communication bandwidth will be mainly used to shrink the system size and to reduce the latency. The use of the newest technologies will certainly also have an effect on the costs of the CMS track trigger. In the following, we assume that an FPGA of a new generation costs about the same as an FPGA of today of a similar class.

No fundamental changes are expected at the crate system. The ATCA standard fits the AM approach well as the full-mesh backplane delivers enough bandwidth to distribute the data of the events to the individual processing boards. The backplane is also the only part of the crate that may change due to technological progress. This could increase the price slightly. However, the crate will most probably become slightly cheaper over time, and this compensates at least partially the backplane cost increase. Thus, costs of the crates are not changed for the estimate for 2022.

The only component on the carrier blade that is essentially affected by new technologies is the FPGA. Currently, a high-end Xilinx Virtex-7 FPGA is mounted on the carrier blade. The requirements on the computational performance of this FPGA are not particularly high. However, the FPGA is the central component implementing the communication between the RTM, PRM and the backplane and it has to provide enough transceivers to fulfill this need. Therefore, a similar FPGA class but on the lower end of logic resources could be part of the new carrier blade. For the projection to 2022, the cost of FPGA is thus reduced to 6000 EUR, and the total costs of the carrier blade become 7350 EUR. Both the costs for the PCB and the other small components will not change substantially. The costs of the RTM belonging to the carrier blade will also not change significantly.

The communication facilities will only undergo a moderate change. A new version of QSFP+ with a bandwidth of 100 Gbit/s is already available [182] but at a very high price. The developments towards 200 and 400 Gbit/s are also ongoing [183]. However, the data are delivered to each trigger tower from many DTCs, and an increased bandwidth does not reduce the number of cables correspondingly. Assuming the installation of 100 Gbit/s fibers and being

conservative with the cable reduction, for the following calculations, 1500 optical fibers are assumed. It is also assumed that the 100 Gbit/s QSFP+ will be available for roughly the same price as the 40 Gbit/s QSFP+ today. Thus, the total costs of the optical communication are reduced to 390 kEUR.

8.8.3. Costs of the PRM in 2022

The technical progress will have the biggest impact on the PRM. For both the AM Chips and the FPGA changes are expected due to the technical progress.

The plan to design a new AM Chip at 28 nm technology may change a lot for the design of the PRM. As the new AM Chip could store up to 500 000 patterns, only four AM Chips are necessary to cover a complete trigger tower. Therefore, fewer chips are necessary, and because of the high initial costs of the chip production, the price per chip will increase. The price of an engineering run at 28 nm is difficult to estimate for 2022. We assume that this technology will still be significantly more expensive than the 65 nm technology today. Thus, the costs of the engineering run are assumed to be 1.5 MEUR. If the number of PRMs per trigger tower does not change a total of 3840 AM2022 chips will be needed. Calculated by Formula 8.1, the cost for a single AM2022 chip with some addition for packaging will be 450 EUR. In the case the time-multiplexing factor could be reduced to ten, only 1920 AM Chips would be necessary, and one would cost 850 EUR.

The reduction of the number of AM Chips from 16 to 4 frees much space on the PRM. Thus, it becomes realistic to reduce the size of the PRM from a double-wide FMC to a standard FMC. However, the power supplies take still much place. The solution, which is used in this cost estimate, is to assume a double-wide FMC with multiple track processors (track finder and track fitter) on it but with a shared power supply. Furthermore, the two RLDRAMs for the pattern lookup can be saved in 2022 by using a Xilinx FPGA with UltraRAM. As described in Section 8.8.1, it is realistic that the future FPGAs contain up to 1 Gbit of RAM. This is more than enough as a whole pattern bank with two million patterns needs a pattern lookup memory of 256 Mbit.

Three cases of PRMs that are possible with the AM2022 chips on a double-wide FMC should be discussed here. The probable layouts are sketched in Figure 8.2. In the first case (layout 1a), two completely parallel track processors are on a mezzanine only sharing the power supply. This means two medium-size FPGAs and eight AM Chips are mounted on the mezzanine. As already today a

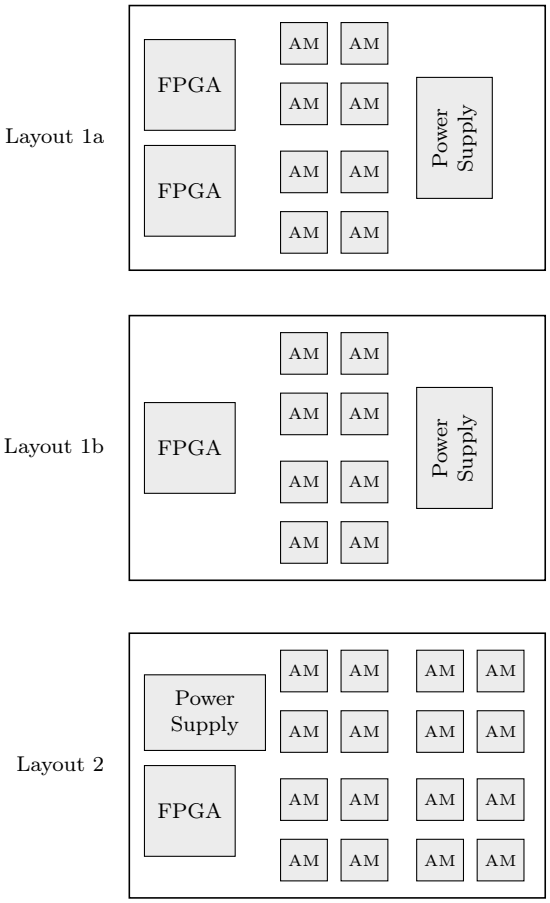


Figure 8.2.: Probable layouts for future PRMs.

medium-size FPGA provides enough resources, an even slightly smaller FPGA of the future generation will suit the needs of the PRM. The advantages of this version are that both processing chains are completely independent. Layout 1b merges the functionality of the two FPGAs into one high-end FPGA. This should be possible by the increase of logic resources and transceiver bandwidth as predicted in Section 8.8.1. However, it is hard to say which option would be cheaper as the cost for two medium-size FPGAs could be lower or higher than for the high-end FPGA. Therefore, only layout 1a is considered in for the further calculations, and the price of one FPGA is assumed to be 2000 EUR.

On the PRM with the AM05 chips, 16 AM Chips are mounted on one FMC. Thus, for layout 2 it is assumed that 16 AM Chips will be mounted on the PRM

together with a large FPGA. This is possible from the available space on the board but might be not possible for another reason. By an increase of the logic resources by the factor of twelve, as predicted in Section 8.8.1, and an increase of transceiver bandwidth, the FPGAs will be powerful enough. However, to access the AM Chip groups individually, many transceivers are necessary, and the number of available transceivers may become a problem. The requirements for the FPGA on such a board would be high. Thus, the costs of a high-end FPGA of 7000 EUR are added to the calculations.

Table 8.7 lists the costs for layout 1 and layout 2 of the future PRM. As the prices of the AM Chips differ largely whether a time-multiplexing factor of 10 or 20 is used in the CMS track trigger, the costs for the layout 1 are calculated with both multiplexing factors. For layout 3, a time-multiplexing factor 10 is unrealistic as this would increase the input bandwidth of a single PRM to 320 Gbit/s. The costs of these new PRMs are at least twice as high as the costs of the actual PRM, and they are going up as far as 14 600 EUR. These high costs are mainly due to the higher price of the AM Chips.

8.8.4. Total costs of the CMS track trigger in 2022

Depending on the options for the PRM described in the previous section, the price for the entire CMS track trigger varies. The number of track processors on one PRM increases with the newly suggested layouts. Consequently, the number of track processors that are hosted in one crate increases accordingly and more than one trigger tower can be processed by one crate. The number of crates also depends on the time-multiplexing factor. Thus, for layout 1 that foresees two track processors on one PRM, 24 crates are needed for 20-fold time multiplexing and 12 crates for 10-fold time multiplexing. In layout 2, one PRM hosts four track processors and 12 crates are sufficient for the expected 20-fold time multiplexing. Additionally, it was defined in Section 8.2 that depending on the number of crates 10 or 12 computation blades are installed within a crate.

Considering these numbers, Table 8.8 assembles the costs of one crate for all three options. The costs of for the layout 1 with 20-fold time multiplexing is just slightly more expensive than the estimated costs of today's CMS track trigger with the 232 kEUR. The higher price of the PRMs is partially compensated by the price reduction of the carrier blades due to the technological progress. In the case of the other two options, the costs of one crate will be significantly higher.

Table 8.7.: Costs of a PRM according to the probable layouts in 2022.

	Layout 1 20-fold TM		Layout 1 10-fold TM		Layout 2 20-fold TM	
Part	Qty.	Price (EUR)	Qty.	Price (EUR)	Qty.	Price (EUR)
PCB	1	190	1	190	1	90
FPGA	2	4000	2	4000	1	7000
AM Chip	8	3600	8	6800	16	7200
Other parts		200		200		200
Total		7990		11190		14590

Table 8.8.: Costs of a populated crate for the CMS track trigger extrapolated to 2022.

	Layout 1 20-fold TM		Layout 1 10-fold TM		Layout 2 20-fold TM	
Part	Qty.	Price (kEUR)	Qty.	Price (kEUR)	Qty.	Price (kEUR)
Crate	1	7.0	1	7.0	1	7.0
Blade	14	103.0	14	103.0	14	103.0
PRM	20	160.0	24	269.0	24	350.0
RTM	14	6.0	14	6.0	14	6.0
Total		276.0		385.0		466.0

Table 8.9 lists the costs of the complete track trigger for all three options. It is shown that the reduction of the number of crates brings some advantage regarding costs. By applying the first option, a cost reduction of 60 percent is predicted. Obviously, the lower number of crates of the second option reduces the costs further. The costs of the third option applying layout 2 lie between the other two. The dense integration of tracker processors on the PRMs causes higher costs than the second option with the same number of crates. This is because a larger FPGA on the PRM will be necessary to implement four track processors.

One of the largest uncertainties are the costs of the ASIC production because the production costs are not publicly available and the future development is

hard to predict. Even if the costs would be 1 million euro higher, the costs of the complete CMS track trigger remain below the cost of the track trigger built by today's technology. To conclude, it seems realistic that the costs of the CMS track trigger built around 2022 will be significantly lower than the budgeted costs. It should also be noted that this estimate does not include the costs of the DTC.

Compared with the other two proposed approaches (TMTT and tracklet), the costs of the AM approach are expected to be higher. The reason is mainly the development and production of the AM Chip.

Table 8.9.: Costs of the complete CMS track trigger extrapolated to 2022.

	Layout 1 20-fold TM		Layout 1 10-fold TM		Layout 2 20-fold TM	
Part	Qty.	Price (MEUR)	Qty.	Price (MEUR)	Qty.	Price (MEUR)
Crate	24	6.6	12	4.6	12	5.6
QSFP+ cable	1500	0.4	1500	0.4	1500	0.4
Total		7.0		5.0		6.0

9. Conclusion

When the HL-LHC will go operational, presumably in 2025, the CMS track trigger will be an essential element of the CMS trigger system. The requirements of the electronic system that forms the CMS track trigger are extraordinary high. Especially, the high input data rates of up to 100 Tbit/s and the low processing time in the range of 4 µs require the application of cutting-edge technologies and novel concepts. For these reasons, the CMS track trigger will be one of the largest and most complex heterogeneous embedded systems in the world. Including the electronics for the particle detection and data collection, the CMS track trigger system will consist of up to 200 000 ASICs and up to 2000 FPGAs.

At the current state of system development and evaluation, different ideas about possible implementations of the CMS track trigger are under investigation. Not only, three completely different approaches are discussed, but also within the AM approach, different research groups investigate diverse aspects of the system. These aspects are: (1) the evaluation of the system properties such as latency and efficiency, (2) the selection of a suitable hardware architecture providing the required performance, (3) and the development of the algorithms and their implementation within FPGAs. Different ideas for all these system aspects circulate. However, these aspects are often investigated in isolation and not within the context of the complete system.

The system simulation presented in this thesis provides a tool to combine the mentioned aspects within a single framework and, therefore, to examine the CMS track trigger as a complete electronic system. With the system simulation, the interaction of the components and the system properties of a proposed CMS track trigger architecture can be evaluated. One sector of the CMS track trigger according to the AM approach has been modeled within the system simulation. We showed that it is feasible to simulate such a large, heterogeneous system with SystemC that consists of more than 8000 components.

Towards the formal review of the competing concepts for the CMS track trigger in December 2016, the exact properties of the key system elements, e.g. latency,

become known. By applying this knowledge to the system simulation, the precise properties of the complete system can be evaluated. Thus, the system simulation provides a valuable tool to evaluate the properties of the complete system. After the review, the system simulation will be used to optimize possible system architectures for the final CMS track trigger before the design is fixed around 2020. For future experiments, we propose to include such a simulation framework in an earlier stage of system design. A simulation of the system architecture, after a first implementation of the algorithm in software but before any hardware development, can guide the direction of the hardware development right from the beginning.

The sc_map library, developed within the activities on the system simulation, facilitates the modeling and configuration of large regular structures within SystemC simulations. The application of this library is not limited to the CMS track trigger, and it can be applied to simulations of other large digital systems, e.g. Systems on a Chip with communication based on a Network on Chip.

With the commissioning and test of the pattern recognition mezzanine (AM05 board), which is the central processing board of the AM approach, we participated in the hardware development for the CMS track trigger. The obtained know-how will be incorporated for the commissioning of the new board (AM06 Board) and the development of any future boards. The test setup presented in this thesis will allow the FPGA developers to access the board remotely from their desks. Thereby, it will be possible for distributed design teams to work with the same board in parallel.

Due to the complexity and size of the CMS track trigger system, the total costs are an important design consideration. A cost estimate of the CMS track trigger has been carried out, which showed that the currently discussed CMS track trigger could be built with the budgeted 11.9 million euro. The extrapolation of the costs showed that the costs due to a possible higher integration of the PRM could be reduced by roughly a factor of two. However, the cost reduction, as well as the system size, depend largely on the technical progress of FPGAs and the development of the AM Chip.

Appendices

A. Logarithmic Fitting

The data sets to which a function is fitted within this thesis are related to the development of semiconductor technology. Therefore, they roughly follow Moore's law [69] which shows an exponential growth. The formula to which the data have been fit is

$$K = K_0 * 2^{\frac{t-t_0}{T_2}} \tag{A.1}$$

where K is the value to which the function is fitted, t is the time in years to which the value is fitted. The constant t_0 is the time offset indicating the starting year of the data row. The free parameters T_2 and K_0 are the evaluated by the fit. T_2 is the doubling time of the fitted value, and K_0 defines the value of the function at time t_0.

In order to achieve a robust fit, the exponential function has been linearized by taking the logarithm on both sides of the function

$$\ln K = \ln K_0 + \frac{\ln 2}{T_2}(t - t_0) \tag{A.2}$$

By applying these substitutions

$$y = \ln K; \quad a = \frac{\ln 2}{T_2}; \quad b = \ln K_0 \tag{A.3}$$

the linear actual fitting function is

$$y = a(t - t_0) + b \tag{A.4}$$

The actual doubling time and start value can be calculated by

$$T_2 = \ln(2)/a \tag{A.5}$$

$$K_0 = \exp b \tag{A.6}$$

B. Evolution of Silicon Tracker - Data and Fit

B.1. Channels of silicon trackers

Table B.1.: Evolution of channel number of silicon trackers in collider experiments. Sources: [5, 7, 24, 51, 53–68].

Experiment	Year	Number of Strip Channels	Number of Pixel Channels
ATLAS - Phase-II	2025	45 000 000	400 000 000
CMS - Phase-II	2025	221 850 000[1]	434 000 000
CMS - Phase-I	2017		123 000 000
ATLAS	2008	6 200 000	140 000 000
CMS	2008	9 600 000	66 000 000
GLAST	2008	1 000 000	
AMS-02	2002	193 000	
DØ	2000	793 000	
Zeus	2000	200 000	
CDF-II	2000	712 000	
BaBar	1999	150 000	
DELPHI (97)	1996	174 080	1 225 728
CDF-I	1992	46 080	
Mark II	1990	18 432	
DELPHI (89)	1989	73 728	
NA11/NA32	1980	2 400	

Table B.2.: Fitted values for the channel number of silicon strip detectors.

	Strip	Pixel
a	0.235	0.172
b	8.115	12.481
T_2	2.951	4.031
K_0	3 343.614	263 276.872

[1] Every channel readout by the readout chips, i.e. includes macro-pixels and both layers of detector modules.

B.2. Active area of silicon trackers

Table B.3.: Evolution of the area of silicon trackers in collider experiments. Sources: [5, 7, 24, 51, 53–68].

Experiment	Start Year	Strip Area (m^2)	Pixel Area (m^2)
ATLAS - Phase-II	2025	193.000	7.000
CMS - Phase-II	2025	205.900	3.200
CMS - Phase-I	2017		1.940
ATLAS	2008	61.100	2.300
CMS	2008	198.000	1.000
GLAST	2008	70.000	
AMS-02	2002	6.300	
DØ	2000	3.000	
Zeus	2000		
CDF-II	2000	6.000	
BaBar	1999		
DELPHI (97)	1996	1.500	0.133
CDF-I	1992	0.220	
Mark II	1990	0.044	
DELPHI (89)	1989	0.420	
NA11/NA32	1980	0.005	

Table B.4.: Fitted values for the area of silicon strip detectors.

	Strip	Pixel
a	0.238	0.108
b	−3.748	−3.163
T_2	2.913	6.397
K_0	0.024	0.042

C. Evolution of FPGAs - Data

C.1. Logic resources

Table C.1.: Evolution of Logic resources in Xilinx FPGAs. Sources: [165–175, 177–181].

Generation	Model	Year	Logic Units	Nr	In	Total Logic LUT Inputs
	XC4025E		1024	2	4	8192
	XC4085XL	1997	3136	2	4	25088
Virtex	XCV1000	1998	6144	4	4	98304
Virtex-II	XC2V8000	2001	46592	2	4	372736
Virtex-II Pro X	XC2VPX70	2003	33088	2	4	264704
Virtex-4	XC4VFX140	2004	63168	2	4	505344
Virtex-5 LX	XC5VLX330	2006	51840	4	6	1244160
Virtex-5 TXT	XC5VTX240T	2008	37440	4	6	898560
Virtex-6	XC6VLX760	2009	118560	4	6	2845440
Virtex-7	XC7V2000T	2010	152700	8	6	7329600
Virtex UltraSCALE	XCVU440	2014	314820	8	6	15111360
Virtex UltraSCALE+	XCVU13P	2015	1636000	1	6	9816000

C.2. RAM resources

Table C.2.: Evolution of RAM resources in Xilinx FPGAs. Sources: [165–175, 177–181].

Generation	Model	Year	BRAM (Mbit)	UltraRAM (Mbit)	Total (Mbit)
Virtex	XCV1000	1998	0.13	0.00	0.13
Virtex-II	XC2V8000	2001	3.02	0.00	3.02
Virtex-II Pro X	XC2VPX70	2003	5.54	0.00	5.54
Virtex-4	XC4VFX140	2004	9.94	0.00	9.94
Virtex-5 LX	XC5VLX330	2006	10.37	0.00	10.37
Virtex-5 TXT	XC5VTX240T	2008	11.66	0.00	11.66
Virtex-6	XC6VHX565T	2009	32.83	0.00	32.83
Virtex-7	XC7V2000T	2010	46.51	0.00	46.51
Virtex UltraSCALE	XCVU190	2014	132.90	0.00	132.90
Virtex UltraSCALE+	XCVU13P	2015	94.50	432.00	526.50

C.3. High-speed serial link bandwidth

Table C.3.: Evolution of high-speed serial link bandwidth in Xilinx FPGAs. BW stands for bandwidth. Sources: [167–169, 171–175, 177–181].

Generation	Model	Year	Nr_1	BW_1	Nr_2	BW_2	Total BW
Virtex-II Pro X	XC2VPX70	2003	20	4.25	0	0	85.0
Virtex-4	XC4VFX140	2004	24	6.5	0	0	156.0
Virtex-5 TXT	XC5VTX240T	2008	48	6.5	0	0	312.0
Virtex-6	XC6VHX565T	2009	48	6.6	24	2.5	376.5
Virtex-7	XC7V2000T	2010	36	12.5	0	0	450.0
Virtex UltraSCALE	XCVU190	2014	60	16	60	33	2940.0
Virtex UltraSCALE+	XCVU13P	2015	128	32.75	0	0	4192.0

D. Prices for CMS Track Trigger Components

Table D.1.: Prices of components for the PRM board. Source: [160].

Component	Manufacturer	Part Number	Price (1)	Price (100)
FPGA Kintex-7	Xilinx	XC7K355T-1FFG901C	1532.50	N/A
FPGA Kintex-7	Xilinx	XC7K355T-2FFG901C	1838.75	N/A
FPGA Kintex-7	Xilinx	XC7K480T-1FFG901C	2908.75	N/A
FPGA Kintex-7	Xilinx	XC7K480T-2FFG901C	3350.40	N/A
FPGA Virtex-7	Xilinx	XC7VX415T-1FFG1927C	3614.69	N/A
FPGA Virtex-7	Xilinx	XC7VX415T-2FFG1927C	4518.63	N/A
FPGA Virtex-7	Xilinx	XC7VX690T-1FFG1927C	N/A	N/A
FPGA Virtex-7	Xilinx	XC7VX690T-2FFG1927C	8072.05	N/A
QSFP+ Cable 20m	Amphenol	ICD040GVP163D-20	297.29	257.64
QSFP+ Cage	TE Connectivity AMP	1888617-1	8.39	7.54
QSFP+ Connector	TE Connectivity AMP	1761987-9	4.17	2.98
RLDRAM 576Mbit	Micron	MT44K16M36RB-107E:A	N/A	36.17

Bibliography

[1] G. Aad, T. Abajyan, B. Abbott, *et al.*, „Observation of a new particle in the search for the standard model Higgs boson with the ATLAS detector at the LHC“, *Physics Letters B*, vol. 716, no. 1, pp. 1–29, Sep. 2012, ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.020.

[2] S. Chatrchyan, V. Khachatryan, A. Sirunyan, *et al.*, „Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC“, *Physics Letters B*, vol. 716, no. 1, pp. 30–61, Sep. 2012, ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.021.

[3] „LHC Machine“, *Journal of Instrumentation*, vol. 3, no. 08, L. Evans and P. Bryant, Eds., S08001, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08001.

[4] O. Brüning and L. Rossi, *The High Luminosity Large Hadron Collider: The New Machine for Illuminating the Mysteries of Universe*, ser. Advanced Series on Directions in High Energy Physics. Singapore: World Scientific, Oct. 2015, vol. 24, ISBN: 978-981-4675-46-8.

[5] The CMS Collaboration, „Technical Proposal for the Phase-II Upgrade of the CMS Detector“, CERN, Geneva, Switzerland, Technical Proposal CERN-LHCC-2015-010, Jun. 2015.

[6] The ATLAS Collaboration, „Letter of Intent for the Phase-II Upgrade of the ATLAS Experiment“, CERN, Geneva, Switzerland, Letter of Intent CERN-LHCC-2012-022, Dec. 2012.

[7] The CMS Collaboration, „The CMS experiment at the CERN LHC“, *Journal of Instrumentation*, vol. 3, no. 08, S08004, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08004.

[8] The CMS Collaboration, „TriDAS project: Technical Design Report, Volume 1: The Trigger Systems“, CERN, Geneva, Switzerland, Technical Design Report CERN-LHCC-2000-038, Dec. 2000.

[9] D. Abbaneo, „Upgrade of the CMS Tracker with tracking trigger“, *Journal of Instrumentation*, vol. 6, no. 12, Dec. 2011, ISSN: 1748-0221. DOI: 10.1088/1748-0221/6/12/C12065.

[10] Cisco Systems Inc., *Cisco Visual Networking Index: Forecast and Methodology, 2015–2020*, 2016.

[11] M. Beretta, A. Annovi, A. Andreani, *et al.*, „Next generation associative memory devices for the FTK tracking processor of the ATLAS experiment“, *Journal of Instrumentation*, vol. 9, no. 03, p. C03053, Mar. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/03/C03053.

[12] The CMS Collaboration. (Jul. 2014). CMSSW application framework, [Online]. Available: https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCMSSWFramework (visited on 04/01/2016).

[13] C. Amstutz, G. Magazzù, M. Weber, and F. Palla, „A simulation framework for the CMS Track Trigger electronics“, *Journal of Instrumentation*, vol. 10, no. 03, P03029, Mar. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/03/P03029.

[14] C. Amstutz and O. Sander, „A Library to Model and Configure Large Regular Structures in SystemC“, presented at the 19th Euromicro Conference on Digital System Design, Limassol, Cyprus: IEEE, Aug. 2016.

[15] P. A. Tipler and R. A. Llewellyn, *Modern physics*, 5th ed. New York, NY: W.H. Freeman, 2008, ISBN: 978-0-7167-7550-8.

[16] P. A. Tipler and G. Mosca, *Physics for scientists and engineers: With modern physics*, 6th ed. (extended). New York: W.H. Freeman, 2008, ISBN: 978-0-7167-8964-2.

[17] A. Ustyuzhanin. (2012). Standard Model, Standard Infographic, [Online]. Available: https://webfest.web.cern.ch/content/standard-model-standard-infographic (visited on 06/03/2016).

[18] P. J. E. Peebles and B. Ratra, „The cosmological constant and dark energy“, *Reviews of Modern Physics*, vol. 75, no. 2, pp. 559–606, Apr. 2003, ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.75.559.

[19] R. Aßmann, M. Lamont, and S. Myers, „A brief history of the LEP collider“, *Nuclear Physics B - Proceedings Supplements*, vol. 109, no. 2-3, pp. 17–31, Jun. 2002, ISSN: 0920-5632. DOI: 10.1016/S0920-5632(02)90005-8.

[20] Nobel Media AB. (2014). The Nobel Prize in Physics 2013, [Online]. Available: http://www.nobelprize.org/nobel_prizes/physics/laureates/2013/ (visited on 04/16/2016).

[21] P. W. Higgs, „Broken Symmetries and the Masses of Gauge Bosons“, *Physical Review Letters*, vol. 13, no. 16, pp. 508–509, Oct. 1964, ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.13.508.

[22] F. Englert and R. Brout, „Broken Symmetry and the Mass of Gauge Vector Mesons“, *Physical Review Letters*, vol. 13, no. 9, pp. 321–323, Aug. 1964, ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.13.321.

[23] O. S. Brüning, P. Collier, P. Lebrun, *et al.*, Eds., *LHC Design Report*, Geneva: CERN, 2004.

[24] The ATLAS Collaboration, G. Aad, E. Abat, *et al.*, „The ATLAS Experiment at the CERN Large Hadron Collider“, *Journal of Instrumentation*, vol. 3, no. 08, S08003, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08003.

[25] The ALICE Collaboration, „The ALICE experiment at the CERN LHC“, *Journal of Instrumentation*, vol. 3, no. 08, S08002, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08002.

[26] The LHCb Collaboration, „The LHCb Detector at the LHC“, *Journal of Instrumentation*, vol. 3, no. 08, S08005, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08005.

[27] The TOTEM Collaboration, „The TOTEM Experiment at the CERN Large Hadron Collider“, *Journal of Instrumentation*, vol. 3, no. 08, S08007, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08007.

[28] The LHCf Collaboration, „The LHCf detector at the CERN Large Hadron Collider“, *Journal of Instrumentation*, vol. 3, no. 08, S08006, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08006.

[29] J. Abraham, P. Abreu, M. Aglietta, *et al.*, „Measurement of the energy spectrum of cosmic rays above 1e18 eV using the Pierre Auger Observatory“, *Physics Letters B*, vol. 685, no. 4-5, pp. 239–246, Mar. 2010, ISSN: 0370-2693. DOI: 10.1016/j.physletb.2010.02.013.

[30] J. L. Pinfold, „The MoEDAL Experiment at the LHC – a New Light on the Terascale Frontier“, *Journal of Physics: Conference Series*, vol. 631, p. 012 014, Jul. 2015, ISSN: 1742-6596. DOI: 10.1088/1742-6596/631/1/012014.

[31] CERN. (Apr. 2016). CMS Luminosity - Public Results, [Online]. Available: https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults (visited on 04/20/2016).

[32] The CMS Collaboration, *Technical proposal for the upgrade of the CMS detector through 2020*. Geneva, Switzerland: CERN, Feb. 2011, ISBN: 978-92-9083-362-8.

[33] W. Barletta, M. Battaglia, M. Klute, *et al.*, „Future hadron colliders: From physics perspectives to technology R&D“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 764, pp. 352–368, Nov. 2014, ISSN: 0168-9002. DOI: 10.1016/j.nima.2014.07.010.

[34] Eleanor Rusack. (May 2011). CMS Collaboration - CMS Experiment, [Online]. Available: http://cms.web.cern.ch/content/cms-collaboration (visited on 04/17/2016).

[35] The CMS Collaboration, „The CMS magnet project : Technical Design Report“, CERN, Geneva, Switzerland, Technical Design Report CERN-LHCC-97-010, May 1997.

[36] The CMS Collaboration, „The CMS electromagnetic calorimeter project : Technical Design Report“, CERN, Geneva, Switzerland, Technical Design Report CERN-LHCC-97-033, Dec. 1997.

[37] C. Grupen and I. Buvat, Eds., *Handbook of Particle Detection and Imaging*, ser. Springer reference. Berlin; New York: Springer, 2012, ISBN: 978-3-642-13270-4.

[38] The CMS Collaboration, „The CMS hadron calorimeter project : Technical Design Report“, CERN, Geneva, Switzerland, Technical Design Report CERN-LHCC-97-031, Jun. 1997.

[39] The CMS Collaboration, „The CMS muon project : Technical Design Report“, CERN, Geneva, Switzerland, Technical Design Report CERN-LHCC-97-032, Dec. 1997.

[40] R. Frühwirth and M. Regler, Eds., *Data analysis techniques for high-energy physics*, 2nd ed, ser. Cambridge monographs on particle physics, nuclear physics, and cosmology 11, Cambridge, U.K. ; New York: Cambridge University Press, 2000, ISBN: 978-0-521-63219-5.

[41] The CMS Collaboration, „The TriDAS Project: Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger. CMS trigger and data-acquisition project“, CERN, Geneva, Switzerland, Technical Design Report CERN-LHCC-2002-026, 2002.

[42] D. Trocino, „The CMS High Level Trigger“, *Journal of Physics: Conference Series*, vol. 513, no. 1, p. 012 036, Jun. 2014, ISSN: 1742-6596. DOI: 10.1088/1742-6596/513/1/012036.

[43] A. Afaq, W. Badgett, G. Bauer, *et al.*, „The CMS High Level Trigger System“, *IEEE Transactions on Nuclear Science*, vol. 55, no. 1, pp. 172–175, Feb. 2008, ISSN: 0018-9499. DOI: 10.1109/TNS.2007.910980.

[44] T. Blank, M. Caselle, M. Weber, *et al.*, „Novel module production methods for the CMS pixel detector, upgrade phase I“, *Journal of Instrumentation*, vol. 10, no. 02, p. C02021, Feb. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/02/C02021.

[45] G. Zito. (Jan. 2012). CMS Tracker Visualization, [Online]. Available: http://zitogiuseppe.com/cms/tvis.html (visited on 05/17/2016).

[46] Jones, J, Hall, G, Foudas, C, and Rose, A, „A Pixel Detector for Level-1 Triggering at SLHC“, presented at the 11th Workshop on Electronics for LHC and Future Experiments, Heidelberg, Germany: CERN, Sep. 2005, pp. 130–134. DOI: 10.5170/CERN-2005-011.130.

[47] D. A. Glaser, „Some Effects of Ionizing Radiation on the Formation of Bubbles in Liquids“, *Physical Review*, vol. 87, no. 4, pp. 665–665, Aug. 1952, ISSN: 0031-899X. DOI: 10.1103/PhysRev.87.665.

[48] F. D. Becchetti, „History of the bubble chamber and related active- and internal-target nuclear tracking detectors“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 784, pp. 518–523, Jun. 2015, ISSN: 0168-9002. DOI: 10.1016/j.nima.2015.01.030.

[49] H. Spieler, *Semiconductor detector systems*, ser. Series on semiconductor science and technology 12. Oxford ; New York: Oxford University Press, 2005, ISBN: 978-0-19-852784-8.

[50] K. G. McKay, „Electron-Hole Production in Germanium by Alpha-Particles“, *Physical Review*, vol. 84, no. 4, pp. 829–832, Nov. 1951, ISSN: 0031-899X. DOI: 10.1103/PhysRev.84.829.

[51] F. Hartmann, *Evolution of silicon sensor technology in particle physics*. Berlin: Springer, 2009, ISBN: 3-540-25094-8.

[52] E. Haase, M. Fawzi, D. Saylor, and E. Velten, „Striped semiconductor detectors for digital position encoding“, *Nuclear Instruments and Methods*, vol. 97, no. 3, pp. 465–469, Dec. 1971, ISSN: 0029-554X. DOI: 10.1016/0029-554X(71)90247-3.

[53] B. Hyams, U. Koetz, E. Belau, *et al.*, „A silicon counter telescope to study short-lived particles in high-energy hadronic interactions“, *Nuclear Instruments and Methods in Physics Research*, vol. 205, no. 1-2, pp. 99–105, Jan. 1983, ISSN: 0167-5087. DOI: 10.1016/0167-5087(83)90177-1.

[54] N. Bingefors, H. Borner, R. Boulter, *et al.*, „The DELPHI Microvertex detector“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 328, no. 3, pp. 447–471, May 1993, ISSN: 0168-9002. DOI: 10.1016/0168-9002(93)90663-3.

[55] P. Chochula, P. Rosinský, A. Andreazza, *et al.*, „The DELPHI Silicon Tracker at LEP2“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 412, no. 2-3, pp. 304–328, Aug. 1998, ISSN: 0168-9002. DOI: 10.1016/S0168-9002(98)00344-1.

[56] G. Bianchi, N. De Maio, and S. Mersi. (). tkLayout - Layout repo, [Online]. Available: http://tklayout.web.cern.ch/content/layout-repo (visited on 05/02/2016).

[57] C. Adolphsen, R. Jacobsen, V. Lüth, *et al.*, „The Mark II silicon strip vertex detector“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 313, no. 1-2, pp. 63–102, Mar. 1992, ISSN: 0168-9002. DOI: 10.1016/0168-9002(92)90086-J.

[58] P. Azzi, „The CDF silicon detector upgrade“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 419, no. 2-3, pp. 532–537, Dec. 1998, ISSN: 0168-9002. DOI: 10.1016/S0168-9002(98)00830-4.

[59] D. Amidei, P. Azzi, N. Bacchetta, *et al.*, „The Silicon Vertex Detector of the Collider Detector at Fermilab“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 350, no. 1-2, pp. 73–130, Oct. 1994, ISSN: 0168-9002. DOI: 10.1016/0168-9002(94)91156-8.

[60] P. Weilhammer, „Experience with Si Detectors in NA32“, in *New solid state devices for high-energy physics*, Berkeley, USA, Oct. 1985.

[61] L. Miller, „Status of the CDF silicon detector“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 518, no. 1-2, pp. 281–285, Feb. 2004, ISSN: 0168-9002. DOI: 10.1016/j.nima.2003.10.082.

[62] The CMS Collaboration, „Projected Performance of an Upgraded CMS Detector at the LHC and HL-LHC: Contribution to the Snowmass Process“, CERN, Geneva, Switzerland, CMS Note CMS-NOTE-2013-002, Jun. 2013.

[63] W. Cooper, „The D0 silicon tracker“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 598, no. 1, pp. 41–45, Jan. 2009, ISSN: 0168-9002. DOI: 10.1016/j.nima.2008.08.065.

[64] A. Polini, I. Brock, S. Goers, *et al.*, „The design and performance of the ZEUS micro vertex detector“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 581, no. 3, pp. 656–686, Nov. 2007, ISSN: 0168-9002. DOI: 10.1016/j.nima.2007.08.167.

[65] C. Sgro, W. Atwood, L. Baldini, *et al.*, „Construction, test and calibration of the GLAST silicon tracker“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 583, no. 1, pp. 9–13, Dec. 2007, ISSN: 0168-9002. DOI: 10.1016/j.nima.2007.08.224.

[66] V. Re, D. Kirkby, M. Bruinsma, *et al.*, „Lessons learned from BaBar silicon vertex tracker, limits, and future perspectives of the detector“, *IEEE Transactions on Nuclear Science*, vol. 52, no. 3, pp. 787–792, Jun. 2005, ISSN: 0018-9499. DOI: 10.1109/TNS.2005.850982.

[67] G. Ambrosi, W. Burger, and A. Oliva, „The AMS-02 silicon tracker: Recent results and current status“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 617, pp. 471–472, 1-3 May 2010, ISSN: 0168-9002. DOI: 10.1016/j.nima.2009.09.109.

[68] T. Barber and U. Parzefall, „Upgrading the ATLAS silicon tracking for the HL-LHC“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 730, pp. 191–194, Dec. 2013, ISSN: 0168-9002. DOI: 10.1016/j.nima.2013.06.085.

[69] G. Moore, „Cramming More Components Onto Integrated Circuits“, *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan. 1998, ISSN: 1558-2256. DOI: 10.1109/JPROC.1998.658762.

[70] L. Caminada, „The CMS Experiment at the LHC“, in *Study of the Inclusive Beauty Production at CMS and Construction and Commissioning of the CMS Pixel Barrel Detector*. Springer Berlin Heidelberg, 2012, pp. 7–22, ISBN: 978-3-642-24561-9.

[71] G. Baulieu, C. Combaret, G. Galbit, *et al.*, „Emulation of a track reconstruction system based on associative memories“, CERN, Detector Note CMS-DN-2015-025, Oct. 2015.

[72] G. Bianchi, „tkLayout: A design tool for innovative silicon tracking detectors“, *Journal of Instrumentation*, vol. 9, no. 03, p. C03054, Mar. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/03/C03054.

[73] International Tennis Federation, *ITF rules of tennis*, 2015.

[74] D. Abbaneo and A. Marchioro, „A hybrid module architecture for a prompt momentum discriminating tracker at HL-LHC“, *Journal of Instrumentation*, vol. 7, no. 09, p. C09001, Sep. 2012, ISSN: 1748-0221. DOI: 10.1088/1748-0221/7/09/C09001.

[75] D. Ceresa, A. Marchioro, K. Kloukinas, *et al.*, „Macro Pixel ASIC (MPA): The readout ASIC for the pixel-strip (PS) module of the CMS outer tracker at HL-LHC“, *Journal of Instrumentation*, vol. 9, no. 11, p. C11012, Nov. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/11/C11012.

[76] G. Blanchot, D. Braga, A. Honma, M. Kovacs, and M. Raymond, „Hybrid circuit prototypes for the CMS Tracker upgrade front-end electronics“, *Journal of Instrumentation*, vol. 8, no. 12, p. C12033, Dec. 2013, ISSN: 1748-0221. DOI: 10.1088/1748-0221/8/12/C12033.

[77] D. Braga, G. Hall, L. Jones, *et al.*, „CBC2: A microstrip readout ASIC with coincidence logic for trigger primitives at HL-LHC“, *Journal of Instrumentation*, vol. 7, no. 10, p. C10003, Oct. 2012, ISSN: 1748-0221. DOI: 10.1088/1748-0221/7/10/C10003.

[78] A. Tricomi, „Upgrade of the CMS tracker“, *Journal of Instrumentation*, vol. 9, no. 03, p. C03041, Mar. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/03/C03041.

[79] G. Hall, M. Pesaresi, M. Raymond, *et al.*, „CBC2: A CMS microstrip readout ASIC with logic for track-trigger modules at HL-LHC“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 765, pp. 214–218, Nov. 2014, ISSN: 0168-9002. DOI: 10.1016/j.nima.2014.04.056.

[80] M. Kovacs, G. Blanchot, A. Honma, A. Kokabi, and M. Raymond, „Flexible front-end hybrids for the CMS outer tracker upgrade“, *Journal of Instrumentation*, vol. 10, no. 01, p. C01046, Jan. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/01/C01046.

[81] P. Moreira, R. Ballabriga, S. Baron, *et al.*, „The GBT project“, CERN, Geneva, Switzerland, 2009.

[82] M. B. Marin, S. Baron, S. Feger, *et al.*, „The GBT-FPGA core: Features and challenges“, *Journal of Instrumentation*, vol. 10, no. 03, p. C03021, Mar. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/03/C03021.

[83] A. Xiang, D. Gong, S. Hou, *et al.*, „A Versatile Link for High-Speed, Radiation Resistant Optical Transmission in LHC Upgrades“, *Physics Procedia*, vol. 37, pp. 1750–1758, 2012, ISSN: 1875-3892. DOI: 10.1016/j.phpro.2012.03.753.

[84] G. Foster, J. Freeman, C. Newman-Holmes, and J. Patrick, „A fast hardware track-finder for the CDF central tracking chamber“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 269, no. 1, pp. 93–100, Jun. 1988, ISSN: 0168-9002. DOI: 10.1016/0168-9002(88)90865-0.

[85] S. Holm, J. Dittman, J. Freeman, *et al.*, „System architecture and hardware design of the CDF XFT online track processor“, in *1999 IEEE Nuclear Science Symposium. Conference Record. 1999 Nuclear Science Symposium and Medical Imaging Conference (Cat. No.99CH37019)*, vol. 1, IEEE, 1999, pp. 267–274. DOI: 10.1109/NSSMIC.1999.842491.

[86] B. Ashmanskas, A. Barchiesi, A. Bardi, *et al.*, „The CDF Silicon Vertex Trigger“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 518, no. 1-2, pp. 532–536, Feb. 2004, ISSN: 0168-9002. DOI: 10.1016/j.nima.2003.11.078.

[87] J. Adelman, A. Annovi, M. Aoki, *et al.*, „The Silicon Vertex Trigger upgrade at CDF“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 572, no. 1, pp. 361–364, Mar. 2007, ISSN: 0168-9002. DOI: 10.1016/j.nima.2006.10.383.

[88] M. Shochet, L. Tompkins, V. Cavaliere, *et al.*, „Fast TracKer (FTK) Technical Design Report“, CERN, Geneva, CERN-LHCC-2013-007, Jun. 2013.

[89] J. Anderson, A. Andreani, A. Andreazza, *et al.*, „A fast hardware tracker for the ATLAS trigger system“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 718, pp. 258–259, Aug. 2013, ISSN: 0168-9002. DOI: 10.1016/j.nima.2012.11.133.

[90] G. Volpi, J. Adelman, P. Albicocco, *et al.*, „The ATLAS Fast Tracker“, presented at the 24th International Workshop on Vertex Detectors, Santa Fe, New Mexico, USA: Proceedings of Science, Jun. 2015.

[91] M. Neubauer, „A fast hardware tracker for the ATLAS trigger system“, in *2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*, IEEE, Oct. 2009, pp. 1876–1879. DOI: 10.1109/NSSMIC.2009.5402177.

[92] A. Andreani, A. Annovi, R. Beccherle, *et al.*, „The Associative Memory Serial Link Processor for the Fast TracKer (FTK) at ATLAS“, *Journal of Instrumentation*, vol. 9, no. 11, p. C11006, Nov. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/11/C11006.

[93] F. Palla, M. Pesaresi, and A. Ryd, „Track Finding in CMS for the Level-1 Trigger at the HL-LHC“, *Journal of Instrumentation*, vol. 11, no. 03, p. C03011, Mar. 2016, ISSN: 1748-0221. DOI: 10.1088/1748-0221/11/03/C03011.

[94] D. Sabes, „L1 track triggering with associative memory for the CMS HL-LHC tracker“, *Journal of Instrumentation*, vol. 9, no. 11, p. C11014, Nov. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/11/C11014.

[95] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed, ser. Springer series in statistics. New York: Springer, 2002, ISBN: 978-0-387-95442-4.

[96] A. Tapper and D. Acosta, „CMS Technical Design Report for the Level-1 Trigger Upgrade“, CERN, Geneva, Technical Design Report CERN-LHCC-2013-011, Jun. 2013.

[97] R. Frazier, S. Fayer, G. Hall, *et al.*, „A demonstration of a Time Multiplexed Trigger for the CMS experiment“, *Journal of Instrumentation*, vol. 7, no. 01, p. C01060, Jan. 2012, ISSN: 1748-0221. DOI: 10.1088/1748-0221/7/01/C01060.

[98] G. Hall, D. Newbol, M. Pesaresi, and A. Rose, „A time-multiplexed track-trigger architecture for CMS“, *Journal of Instrumentation*, vol. 9, no. 10, p. C10034, Oct. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/10/C10034.

[99] G. Hall, „A time-multiplexed track-trigger for the CMS HL-LHC upgrade“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 824, pp. 292–295, Jul. 2016, ISSN: 0168-9002. DOI: 10.1016/j.nima.2015.09.075.

[100] C. Amstutz, F. A. Ball, M. N. Balzer, *et al.*, „An FPGA-based Track Finder for the L1 Trigger of the CMS Experiment at the High Luminosity LHC“, presented at the 20th IEEE Real Time Conference, Padova, Italy: IEEE, Jun. 2016. DOI: 10.1109/RTC.2016.7543102.

[101] C. Amstutz, F. A. Ball, M. N. Balzer, *et al.*, „Emulation of a prototype FPGA track finder for the CMS Phase-2 upgrade with the CIDAF emulation framework“, presented at the 20th IEEE Real Time Conference, Padova, Italy: IEEE, Jun. 2016.

[102] Imperial College. (2013). MP7 - Master Processor, [Online]. Available: http://www.hep.ph.ic.ac.uk/mp7/ (visited on 05/24/2016).

[103] K. Compton, S. Dasu, A. Farmahini-Farahani, *et al.*, „The MP7 and CTP-6: Multi-hundred Gbps processing boards for calorimeter trigger upgrades at CMS“, *Journal of Instrumentation*, vol. 7, no. 12, p. C12024, Dec. 2012, ISSN: 1748-0221. DOI: 10.1088/1748-0221/7/12/C12024.

[104] PICMG, *Micro Telecommunications Computing Architecture Base Specification - PICMG MTCA.0 Revision 1.0*, Jul. 2006.

[105] Avago Technologies, *MiniPOD AFBR-811VxyZ, AFBR-821VxyZ - Product Brief*, Aug. 2013.

[106] J. Chaves, „Implementation of FPGA-based level-1 tracking at CMS for the HL-LHC“, *Journal of Instrumentation*, vol. 9, no. 10, p. C10038, Oct. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/10/C10038.

[107] M. Baber, M. Blake, J. Brooke, *et al.*, „Development and testing of an upgrade to the CMS level-1 calorimeter trigger“, *Journal of Instrumentation*, vol. 9, no. 01, p. C01006, Jan. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/01/C01006.

[108] M. Dell'Orso and L. Ristori, „VLSI structures for track finding“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 278, no. 2, pp. 436–440, Jun. 1989, ISSN: 0168-9002. DOI: 10.1016/0168-9002(89)90862-0.

[109] A. Igor, C. Trevis, and A. Sheikholeslami, „A ternary content-addressable memory (TCAM) based on 4T static storage and including a current-race sensing scheme“, *IEEE Journal of Solid-State Circuits*, vol. 38, no. 1, pp. 155–158, Jan. 2003, ISSN: 0018-9200. DOI: 10.1109/JSSC.2002.806264.

[110] PICMG, *AdvancedTCA Base Specification - PICMG 3.0 Revision 3.0*, Mar. 2008.

[111] A. Karlsson and B. Martin, „ATCA: Its performance and application for real time systems“, *IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 688–693, Jun. 2006, ISSN: 0018-9499. DOI: 10.1109/TNS.2006.873404.

[112] JumpGen Systems - M. Dharm, *Whitepaper: Advantages of a Full-Mesh Fabric Configuration vs. Dual-Star in AdvancedTCA Systems*, Mar. 2010.

[113] Artesyn Embedded Technologies - D. Sandy, and T. Wynia, *100G+ on a Standard Platform*, Oct. 2014.

[114] T. Jezynski, R. Larsen, and P. Le Du, „ATCA/µTCA for physics“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 623, no. 1, pp. 510–512, Nov. 2010, ISSN: 0168-9002. DOI: 10.1016/j.nima.2010.03.053.

[115] J. Olsen, T. Liu, and Y. Okumura, „A full mesh ATCA-based general purpose data processing board“, *Journal of Instrumentation*, vol. 9, no. 01, p. C01041, Jan. 2014, ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/01/C01041.

[116] Y. Okumura, J. Olsen, T. T. Liu, and Y. Hang, „Prototype performance studies of a full mesh ATCA-based general purpose data processing board“, presented at the IEEE Nuclear Science Symposium and Medical Imaging Conference (2013 NSS/MIC), IEEE, Oct. 2013. DOI: 10.1109/NSSMIC.2013.6829449.

[117] Y. Okumura, T. Liu, J. Olsen, *et al.*, „ATCA-based ATLAS FTK input interface system“, *Journal of Instrumentation*, vol. 10, no. 04, p. C04032, Apr. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/04/C04032.

[118] American National Standards Institute and VITA Standards Organization, *ANSI-VITA 57.1-2008 American National Standard for FPGA mezzanine card (FMC) standard.* Fountain Hills (Arizona): VMEbus International Trade Association, 2008, ISBN: 978-1-885731-49-4.

[119] K. Pagiamtzis and A. Sheikholeslami, „Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey“, *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, Mar. 2006, ISSN: 0018-9200. DOI: 10.1109/JSSC.2005.864128.

[120] T.-B. Pei and C. Zukowski, „Putting routing tables in silicon“, *IEEE Network*, vol. 6, no. 1, Jan. 1992, ISSN: 0890-8044. DOI: 10.1109/65.120723.

[121] H. Chao, „Next generation routers“, *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1518–1558, Sep. 2002, ISSN: 0018-9219. DOI: 10.1109/JPROC.2002.802001.

[122] M. Nakanishi and T. Ogura, „Real-time CAM-based Hough transform algorithm and its performance evaluation“, *Machine Vision and Applications*, vol. 12, no. 2, pp. 59–68, Aug. 2000, ISSN: 0932-8092, 1432-1769. DOI: 10.1007/s001380050125.

[123] S. Panchanathan and M. Goldberg, „A content-addressable memory architecture for image coding using vector quantization“, *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2066–2078, Sep. 1991, ISSN: 1053-587X. DOI: 10.1109/78.134438.

[124] A. Annovi, R. Beccherle, M. Beretta, *et al.*, „Associative memory design for the fast track processor (FTK) at ATLAS“, in *2011 IEEE Nuclear Science Symposium Conference Record*, IEEE, Oct. 2011, pp. 141–146. DOI: 10.1109/NSSMIC.2011.6154467.

[125] S. R. Amendolia, S. Galeotti, F. Morsani, *et al.*, „The AMchip: A full-custom CMOS VLSI associative memory for pattern recognition“, *IEEE Transactions on Nuclear Science*, vol. 39, no. 4, pp. 795–797, Aug. 1992, ISSN: 0018-9499. DOI: 10.1109/23.159709.

[126] P. Fischer, „First implementation of the MEPHISTO binary readout architecture for strip detectors“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 461, no. 1-3, pp. 499–504, Apr. 2001, ISSN: 0168-9002. DOI: 10.1016/S0168-9002(00)01283-3.

[127] A. Bardi, S. Belforte, M. Dell'Orso, *et al.*, „A prototype of programmable associative memory for track finding“, *IEEE Transactions on Nuclear Science*, vol. 46, no. 4, pp. 940–946, Aug. 1999, ISSN: 0018-9499. DOI: 10.1109/23.790708.

[128] A. Andreani, A. Annovi, R. Beccherle, *et al.*, „Next generation associative memory devices for the FTK tracking processor of the ATLAS experiment“, in *Proceedings of nuclear science symposium and medical imaging conference*, Seoul, South Korea: IEEE, Oct. 2013, pp. 1–6. DOI: 10.1109/NSSMIC.2013.6829550.

[129] M. Shochet, L. Tompkins, V. Cavaliere, *et al.*, „Fast TracKer (FTK) Technical Design Report“, CERN, Geneva, Switzerland, Technical Design Report CERN-LHCC-2013-007, Mar. 2013.

[130] T. Liu, G. Deptuch, J. Hoff, *et al.*, „Design and testing of the first 2D Prototype Vertically Integrated Pattern Recognition Associative Memory“, *Journal of Instrumentation*, vol. 10, no. 02, p. C02029, Feb. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/02/C02029.

[131] T. Harbaum, M. Seboui, M. Balzer, J. Becker, and M. Weber, „A Content Adapted FPGA Memory Architecture with Pattern Recognition Capability for L1 Track Triggering in the LHC Environment“, presented at the The 24th IEEE International Symposium on Field-Programmable Custom Computing Machines, Washington DC, USA: IEEE Computer Society, 2016.

[132] D. Magalotti, L. Alunni, N. Biesuz, *et al.*, „A Pattern Recognition Mezzanine based on Associative Memory and FPGA technology for Level 1 Track Triggers for the HL-LHC upgrade“, *Journal of Instrumentation*, vol. 11, no. 02, p. C02063, Feb. 2016, ISSN: 1748-0221. DOI: 10.1088/1748-0221/11/02/C02063.

[133] Xilinx Inc., *7 Series FPGAs Overview*, May 2015.

[134] Xilinx Inc., *UltraScale Architecture and Product Overview*, Jun. 2016.

[135] Micron Technology Inc., *Datasheet: RLDRAM 3 - MT44K32M18 / MT44K16M36*, 2011.

[136] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk*. Burlington, MA: Morgan Kaufmann Publishers, 2008, ISBN: 978-0-12-379751-3.

[137] HiTech Global, *HTG-V6-PCIE-xxxx User Manual*, Aug. 2010.

[138] C. G. Larrea, K. Harder, D. Newbold, *et al.*, „IPbus: A flexible Ethernet-based control system for xTCA hardware“, *Journal of Instrumentation*, vol. 10, no. 02, p. C02019, Feb. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/02/C02019.

[139] Fermilab. (2014). Scientific Linux, [Online]. Available: https://www.scientificlinux.org/ (visited on 06/02/2016).

[140] Xilinx Inc., *Vivado Design Suite User Guide - Programming and Debugging (v2015.1)*, Apr. 2015.

[141] Xilinx Inc., *7 Series FPGAs GTX/GTH Transceivers - User Guide*, Aug. 2015.

[142] Á. Hegedüs, Á. Horváth, and D. Varró, „A model-driven framework for guided design space exploration“, *Automated Software Engineering*, vol. 22, no. 3, pp. 399–436, Sep. 2015, ISSN: 1573-7535. DOI: 10.1007/s10515-014-0163-1.

[143] The CMS Collaboration. (2016). Github: CMS Offline Software, [Online]. Available: http://github.com/cms-sw/cmssw (visited on 04/26/2016).

[144] P. Elmer, B. Hegner, and L. Sexton-Kennedy, „Experience with the CMS event data model“, *Journal of Physics: Conference Series*, vol. 219, no. 3, p. 032 022, Apr. 2010, ISSN: 1742-6596. DOI: 10.1088/1742-6596/219/3/032022.

[145] Mentor Grapchics. (2016). ModelSim ASIC and FPGA Design, [Online]. Available: https://www.mentor.com/products/fv/modelsim/ (visited on 04/27/2016).

[146] IEEE Computer Society, *1666-2011 - IEEE Standard for standard SystemC language reference manual*, Jan. 2012.

[147] S. Liao, „Towards a new standard for system-level design“, in *Proceedings of the Eighth International Workshop on Hardware/software Codesign*, San Diego, CA, USA: IEEE, 2000, pp. 2–6, ISBN: 978-1-58113-214-4. DOI: 10.1109/HSC.2000.843697.

[148] D. C. Black, *SystemC: From the ground up*, Second edition, in collab. with J. Donovan, B. Bunton, and A. Keist. New York: Springer, 2010, ISBN: 978-0-387-69957-8.

[149] Accelera - Systems Initiative. (2016). SystemC, [Online]. Available: http://www.accellera.org/downloads/standards/systemc (visited on 04/27/2016).

[150] M. Hosseinabady and J. L. Nunez-Yanez, „Effective modelling of large NoCs using SystemC“, in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, IEEE, May 2010, pp. 161–164. DOI: 10.1109/ISCAS.2010.5538028.

[151] F. James, „Monte Carlo theory and practice“, *Reports on Progress in Physics*, vol. 43, no. 9, p. 1145, 1980.

[152] MentorGraphics, *ModelSim SE User's Manual - Software Version 10.1b*, 2012.

[153] D. Abrahams. (Aug. 2003). Boost.Python - 1.55.0, [Online]. Available: http://www.boost.org/doc/libs/1_55_0/libs/python/doc/ (visited on 10/15/2014).

[154] N. M. Josuttis, *The C++ standard library: A tutorial and reference*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2012, ISBN: 978-0-321-62321-8.

[155] P. J. Ashenden, *The designer's guide to VHDL*, 3rd ed, ser. The Morgan Kaufmann series in systems on silicon. Amsterdam; Boston: Morgan Kaufmann Publishers, 2008, ISBN: 978-0-12-088785-9.

[156] J. Noble, „Iterators and encapsulation“, presented at the 33th International Conference on Technology of Object-Oriented Languages, Mont-Saint-Michel: IEEE Comput. Soc, 2000, pp. 431–442, ISBN: 978-0-7695-0731-6. DOI: 10.1109/TOOLS.2000.848781.

[157] Accellera. (Mar. 2012). Systemc/systemc-2.3: New release of the systemc libraries, [Online]. Available: https://github.com/systemc/systemc-2.3 (visited on 03/29/2016).

[158] J. Olsen, *Personal Communication*, E-Mail, 2016.

[159] G. Fedi, *Personal Communication*, E-Mail, 2016.

[160] DigiKey Electronics. (2016). Distributor elektronischer Komponenten, [Online]. Available: http://www.digikey.de/ (visited on 06/10/2016).

[161] Würth Elektronik. (2016). PCB-Preise, [Online]. Available: http://www.wedirekt.de/de/pcb (visited on 06/10/2016).

[162] Pentair, *Angebot ATCA Crates*, May 2014.

[163] Avnet Inc. (2016). Avnet - Product Finder, [Online]. Available: http://products.avnet.com/ (visited on 06/17/2016).

[164] Xilinx Inc. (2016). xilinx.com, [Online]. Available: http://www.xilinx.com/ (visited on 06/15/2016).

[165] Xilinx Inc., *XC4000E and XC4000X Series Field Programmable Gate Arrays-Product Specification*, May 1999.

[166] „Virtex - Our New Million-Gate 100-MHz FPGA Technology“, *Xilinx Xcell*, no. 27, pp. 4–5, Jan. 1998.

[167] Xilinx Inc., *UltraScale Architecture - Configurable Logic Block*, Feb. 2015.

[168] J. Lazzaro. (2010). Xilinx Part Family History, [Online]. Available: http://www-inst.eecs.berkeley.edu/~cs294-59/fa10/resources/Xilinx-history/Xilinx-history.html (visited on 03/09/2015).

[169] Xilinx Inc., *UltraScale Architecture and Product Overview*, Feb. 2015.

[170] „Hi-Rel Product Roadmap Provides High-Density Leadership“, *Xilinx Xcell*, no. 26, p. 8, Jul. 1997.

[171] Xilinx Inc., *7 Series FPGAs Configurable Logic Block - User Guide*, Nov. 2014.

[172] Xilinx Inc., *UltraScale+ FPGAs Product Selection Guide*, Feb. 2015.

[173] Xilinx Inc., *Virtex-II Platform FPGAs: Complete Data Sheet*, Apr. 2014.

[174] Xilinx Inc., *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet*, Jun. 2011.

[175] Xilinx Inc., *Virtex-4 Family Overview*, Aug. 2010.

[176] Xilinx Inc., *Virtex-5 Family Overview*, Aug. 2015.

[177] Xilinx Inc., *Virtex-6 Family Overview*, Aug. 2015.

[178] M. Santarini, „Targeted Design Platforms Put Innovation on Fast Track“, *Xilinx Xcell*, no. 68, pp. 8–12, Jul. 2009.

[179] Xilinx Inc., „Xilinx Redefines State of the Art with New 7 Series FPGAs“, *Xilinx Xcell*, no. 72, pp. 6–11, Jul. 2010.

[180] Xilinx Inc., „Virtex-5 Special Edition“, *Xilinx Xcell*, no. 59, Sep. 2006.

[181] Xilinx Inc., „Virtex-4 Special Edition“, *Xilinx Xcell*, no. 52, Jan. 2005.

[182] Cisco Systems Inc., *Cisco 100GBASE QSFP-100G Modules - Data Sheet*, 2016.

[183] QSFP-DD. (Mar. 2016). Thirteen Industry Leaders Collaborate to Increase Pluggable Module Speeds to 400 Gbps, [Online]. Available: `http://www.qsfp-dd.com/13-industry-leaders-collaborate-to-increase-pluggable-module-speeds-to-400-gbps/` (visited on 06/15/2016).

[184] G. Aad, B. Abbott, J. Abdallah, *et al.*, „Combined Measurement of the Higgs Boson Mass in pp Collisions at sqrt(s) = 7 and 8 TeV with the ATLAS and CMS Experiments“, *Physical Review Letters*, vol. 114, no. 19, May 2015, ISSN: 0031-9007, 1079-7114. DOI: `10.1103/PhysRevLett.114.191803`.

[185] G. Hall, „A time-multiplexed track-trigger for the CMS HL-LHC upgrade“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Oct. 2015, ISSN: 0168-9002. DOI: `10.1016/j.nima.2015.09.075`.

[186] P. A. Hartmann, „sc_vector: A flexible container for modules, ports and channels“, 23rd European SystemC User's Group Meeting, Grenoble, France, Mar. 2011.

[187] D. I. Kazakov, „The Higgs boson is found: What is next?“, *Physics-Uspekhi*, vol. 57, no. 9, pp. 930–942, Sep. 2014, ISSN: 1468-4780. DOI: 10.3367/UFNe.0184.201409j.1004.

[188] C.-E. Wulz, „CMS physics overview“, *Czechoslovak Journal of Physics*, vol. 52, no. 3, pp. C155–C170, Mar. 2002, ISSN: 1572-9486. DOI: 10.1007/s10582-002-0107-z.

[189] PICMG, *Advanced Mezzanine Card Base Specification - PICMG AMC.0 Revision 2.0*, Nov. 2006.

[190] CERN. (May 2015). Cactus project, [Online]. Available: https://svnweb.cern.ch/trac/cactus (visited on 06/20/2016).

[191] R. Frazier, G. Iles, M. Magrans de Abril, *et al.*, *The IPbus Protocol: Version 2.0*, Dec. 2013.

[192] Xilinx Inc., *Integrated Logic Analyzer v6.1 - LogiCORE IP Product Guide*, Apr. 2016.

[193] C. Adorisio, G. Ambrosio, R. B. Appleby, *et al.*, *The High Luminosity Large Hadron Collider: The New Machine for Illuminating the Mysteries of Universe*, O. Brüning and L. Rossi, Eds., ser. Advanced Series on Directions in High Energy Physics. Singapore: World Scientific, Oct. 2015, vol. 24, ISBN: 978-981-4675-47-5.

[194] CERN. (2016). The Large Hadron Collider, [Online]. Available: http://home.cern/topics/large-hadron-collider (visited on 04/14/2016).

[195] V. Khachatryan, A. M. Sirunyan, A. Tumasyan, *et al.*, „Search for supersymmetry with photons in pp collisions at sqrt(s) = 8 TeV“, *Physical Review D*, vol. 92, no. 7, Oct. 2015, ISSN: 1550-2368. DOI: 10.1103/PhysRevD.92.072006.

[196] R. Bailey, E. Belau, T. Böhringer, *et al.*, „A silicon strip detector telescope for the measurement of production and decay of charmed particles“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 226, no. 1, pp. 56–58, Sep. 1984, ISSN: 0168-9002. DOI: 10.1016/0168-9002(84)90165-7.

[197] M. Krammer, „The silicon sensors for the Inner Tracker of the Compact Muon Solenoid experiment“, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 531, no. 1-2, pp. 238–245, Sep. 2004, ISSN: 0168-9002. DOI: 10.1016/j.nima.2004.06.011.

[198] D. Kotliński, „Status of the CMS Pixel detector“, *Journal of Instrumentation*, vol. 4, no. 03, P03019, Mar. 2009, ISSN: 1748-0221. DOI: 10.1088/1748-0221/4/03/P03019.

[199] J. Hoff, M. Johnson, R. Lipton, *et al.*, „Design for a L1 tracking trigger for CMS“, *Journal of Instrumentation*, vol. 8, no. 02, p. C02004, Feb. 2013, ISSN: 1748-0221. DOI: 10.1088/1748-0221/8/02/C02004.

List of Publications

[1] C. Amstutz, G. Magazzù, M. Weber, and F. Palla, „A simulation framework for the CMS Track Trigger electronics“, *Journal of Instrumentation*, vol. 10, no. 03, P03029, Mar. 2015, ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/03/P03029.

[2] C. Amstutz, F. A. Ball, M. N. Balzer, *et al.*, „Emulation of a prototype FPGA track finder for the CMS Phase-2 upgrade with the CIDAF emulation framework“, presented at the 20th IEEE Real Time Conference, Padova, Italy: IEEE, Jun. 2016.

[3] C. Amstutz, F. A. Ball, M. N. Balzer, *et al.*, „An FPGA-based Track Finder for the L1 Trigger of the CMS Experiment at the High Luminosity LHC“, presented at the 20th IEEE Real Time Conference, Padova, Italy: IEEE, Jun. 2016. DOI: 10.1109/RTC.2016.7543102.

[4] C. Amstutz and O. Sander, „A Library to Model and Configure Large Regular Structures in SystemC“, presented at the 19th Euromicro Conference on Digital System Design, Limassol, Cyprus: IEEE, Aug. 2016.

Collaboration Papers

[1] G. Aad, B. Abbott, J. Abdallah, *et al.*, „Combined Measurement of the Higgs Boson Mass in pp Collisions at sqrt(s) = 7 and 8 TeV with the ATLAS and CMS Experiments“, *Physical Review Letters*, vol. 114, no. 19, May 2015, ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.114.191803.

[2] V. Khachatryan, A. M. Sirunyan, A. Tumasyan, *et al.*, „Search for supersymmetry with photons in pp collisions at sqrt(s) = 8 TeV“, *Physical Review D*, vol. 92, no. 7, Oct. 2015, ISSN: 1550-2368. DOI: 10.1103/PhysRevD.92.072006.