VRIJE
UNIVERSITEIT
BRUSSEL

Proef ingediend met het oog op het behalen van de graad van master in de natuurwetenschappen

FIRMWARE DEVELOPMENT AND CHARACTERIZATION OF CMS PHASE II OUTER TRACKER PROTOTYPE MODULES

Master Thesis

Louis Dehennin

June 8, 2018

Professor S. Lowette, Jarne De Clercq

Science & Bio-Engineering

All my gratitude goes towards Jarne en Steven. They both have been incredibly helpful and supportive from the very beginning. Jarne especially was pretty much always available for consulting or a casual conversation. It was a pleasure working with you. Good luck for next year! Steven's guiding and efforts in and outside of the thesis project were also very much appreciated. I wish you the best.

Personal Contributions

From the start the goal of the thesis was to implement the CBC2 calibration procedure on an FPGA. Software procedures were already designed and there was an interest to see if the calibration could be sped up, as a large amount of chips will have to be calibrated at the same time in the CMS tracker.

The major initial hurdle was to learn how and what it meant to program hardware. As these topics are not covered in our curriculum, starting from the very basics and quickly learning the skills to work on dedicated firmware for a custom, complex chip was challenging. The development of the CBCs and the supporting soft- and firmware is still fairly recent. At the time I started, no comprehensive documentation of the firmware structure was available. A great deal of time was spent exploring the different blocks' functionalities and intricacies.

A major difference between programming hardware and coding software is the lack of feedback. When something doesn't work, it doesn't and that's all you get. As this is highly impractical, a solution is to work with simulations. Instead of running the firmware on the FPGA, the computer interprets and calculates the behaviour of the firmware design for a number of clock cycles, allowing to track the values of signals over time. The drawback is that the simulation can not realistically communicate with a CBC or receive IPbus input from the computer, these communications have to be programmed and simulated separately. Thankfully a simulation mode for CBC3 was already present in the firmware, serving as an excellent template to implement the CBC2 simulation. The programming of the simulation was a great exercise to better understand the content of the firmware but also the inner working of the CBC2 chip.

It is now finally possible to start designing the calibration procedure. I was fully responsible for the design of the implementation, although it was heavily inspired by pre-existing software calibration procedure. Once parts of the procedure were ready to be tested on the FPGA, a new hurdle showed up: the FPGA is operated with custom (undocumented) software code. It is fair to conclude that a steep learning curve is present before even achieving a first, basic result.

The implementation of the calibration was successful and results have been presented internally at the IIHE tracker meeting, but also over video conference at the CMS OT upgrade DAQ meeting during the tracker week in May 2018. Results were well received and the feedback was all around very positive. After the initial hurdle of learning all the different aspects, contributing to the study of the CBCs was a lot easier. Although not mentioned in the thesis, some extra work has been performed. Notably on measuring the shape of a test pulse and the interpretation of measurements on a purposefully defect hybrid.

Contents

Abstract

In 2024 wordt de operatie van de Large Hadron Collider stilgezet om werken te beginnen aan een nieuwe fase van het experiment. De luminositeit wordt substantieel verhoogd met het oog op het meten van statistische zeldzame interacties. Deze staan toe om de eigenschappen van het Brout-Englert-Higgs (BEH) deeltje preciezer te verkennen, en eventueel nieuwe fysica te ontdekken buiten het Standaard Model. Samen met de upgrade van de versneller, moeten ook de detectoren aangepast worden om het verhoogde aantal aan interacties te behandelen. In deze thesis wordt specifiek gekeken naar een uitlees chip in de Outer Tracker van de Compact Muon Solenoid detector. In het bijzonder naar een procedure om de chip te calibreren, en de implementatie ervan in een FPGA. De performantie van de calibratie wordt bestudeerd en vergeleken met reeds bestaande procedures, met veelbelovende resultaten als gevolg.

Abstract

2024 marks the start of a scheduled upgrade of the Large Hadron Collider, aiming to substantially increase the luminosity at which the accelerator operates. The upgrade is motivated by statistically rare interactions allowing to further improve knowledge of the Brout-Englert-Higgs particle and probe for new physics beyond the Standard Model. As the accelerator will be upgraded, so will the detectors measuring the interactions. In this thesis we are looking at a newly designed read-out chip for the Outer Tracker of the Compact Muon Solenoid detector. In particular, a procedure to calibrate the chip and the implementation of the procedure on an FPGA. The performance of the calibration is then studied and compared to existing procedures, exhibiting promising results.

1 LHC and CMS

1.1 Large Hadron Collider

The Large Hadron Collider is a circular particle accelerator famously known for being the largest nad most powerful accelerator in the world with a circumference of 27 km. One of its main purposes was to discover the Brout-Englert-Higgs (BEH) Boson, which has been observed in two distinct LHC experiments, ATLAS and CMS, in 2012 [1][2].

Construction of the LHC was conducted and realized by the CERN organization, the European Organization for Nuclear Research. The organization was founded in 1949 and has since performed many experiments related to nuclear and particle physics. In fact, the LHC is built as a replacement of the LEP in the same tunnel. The accelerator complex (fig. 1) is a succession of machines each accelerating the particles up to a certain energy before injecting them into the next machine in the chain. Usually proton-proton collisions are studied at the LHC, although there is the possibility to also collide heavy ions. Protons start their journey in a bottle containing hydrogen. Electrons are stripped and the remaining protons are injected in a linear accelerator, LINAC2. The following stages all consist of increasingly larger, more powerful circular accelerators: Proton Booster, Proton Synchrotron, Super Proton Synchrotron. Finally the protons are injected both in clockwise and anti-clockwise direction into the LHC.

Rather than a continuous beam, particles are squeezed together in discrete, individual packets called bunches. When peak energy is reached, these bunches are aimed to cross and collide four interaction points on the accelerator. In the current configuration of the LHC, bunches cross at a rate of 40 MHz at each interaction point.

In a beam collision experiment with a probe and a target, the scattering cross section σ is the effective size of a single target particle as seen by the probe. It is expressed in a unit of area called *barn*, equal to 10^{-28} m^2. For a single probe particle, the probability of collision is estimated as the fraction of the effective size of all target particles n_t, divided by the total target area A_t:

$$P_{\text{collision}} \approx \frac{n_t \sigma}{A_t}.$$

The average number of collisions is equal to this probability multiplied with the total number of probe particles:

$$n_{\text{collision}} = \frac{\sigma n_t}{A_t} n_p.$$

In terms of proton bunches in the LHC acting as both probe and target, the number of target and probe particles is (ideally) the same. The target area is the

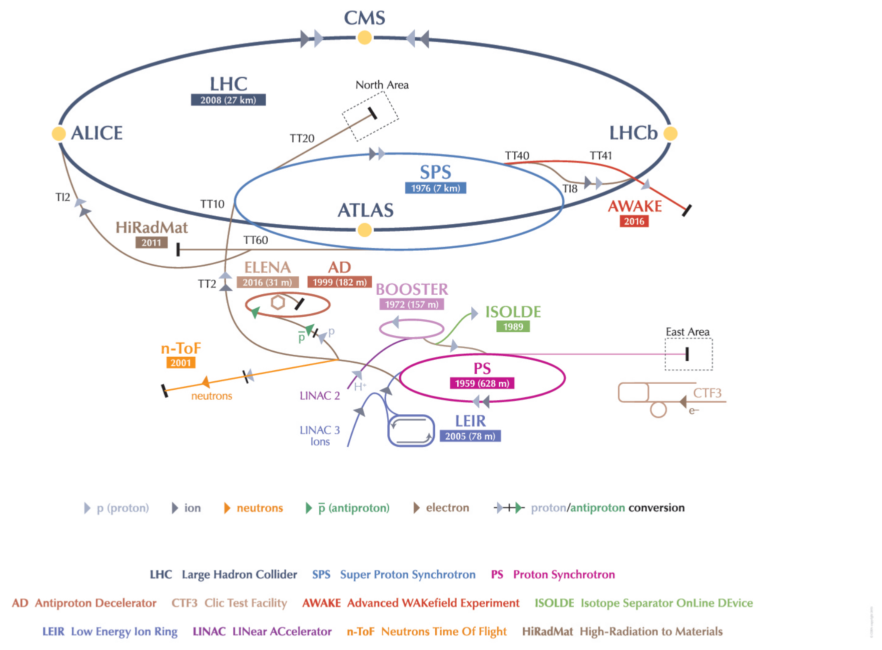


Figure 1: Accelerator complex, showing the different stages before particles reach the LHC. The journey of protons can be followed starting at LINAC2, where bottled hydrogen is injected and stripped from its electrons, before being accelerated into a succession of accelerators. Image provided by courtesy of the CERN organization brochure.

cross-sectional area of the bunch or beam. Previous equation describes the average number of proton collisions when two bunches cross. Total collision rate is then simply the bunch crossing rate f multiplied with $n_{\text{collision}}$,

$$R_{\text{collision}} = \sigma_{proton} \frac{fn^2}{A_{beam}}.$$

The quantity fn^2/A is called the *luminosity* L. It is a measure for the potential amount of collisions per unit time per unit area. Integrated luminosity, $\int Ldt$, is a useful quantity to convey how much data has been collected over the course of an experiment. It gives the number of collisions of a particular reaction when multiplied with that reaction's scattering cross-section. The total number of proton collisions for example is simply $\sigma_{proton} \int Ldt$. Often in literature the amount of integrated luminosity is expressed in inverse femtobarn (fb^{-1}). By 2024, the LHC experiments are estimated to have collected about 300 fb^{-1} of integrated luminosity [3].

1.2 Compact Muon Solenoid

1.2.1 Description

The Compact Muon Solenoid (CMS) is a general purpose detector located at one of the four interaction points on the LHC. The task of the detector is to record trajectories and energy from the particles originating from the interactions occurring at its core. To capture the rich and diverse content of these interactions the detector is built as a barrel with end caps consisting of several layers or subdetectors, each serving a different purpose. A cross-section of the detector is displayed in figure 2.

Charged particles interact in the inner layers of the detector with silicon sensors. The ionizing particles deposit energy creating electron-hole pairs in the charge depleted region of the sensors. Particle hits are recorded by collecting these charges. Electrons and photons are then fully absorbed in the following layer, the electromagnetic calorimeter (ECAL). The function of the calorimeter is to measure the total energy of the particle. Hadrons typically reach beyond the ECAL and are absorbed in the hadron calorimeter (HCAL). The last layer is designed specially for the detection of muons, explaining part of CMS's name. The "compact" in CMS refers to the very dense design of the detector, despite its dimensions (15m radius, 28m long). An essential part of the detector is its magnetic field, bending charged particles. The tranverse momentum p_T of a particle, the momentum in the plane transverse to the beam, can be derived from the curvature of its reconstructed track.

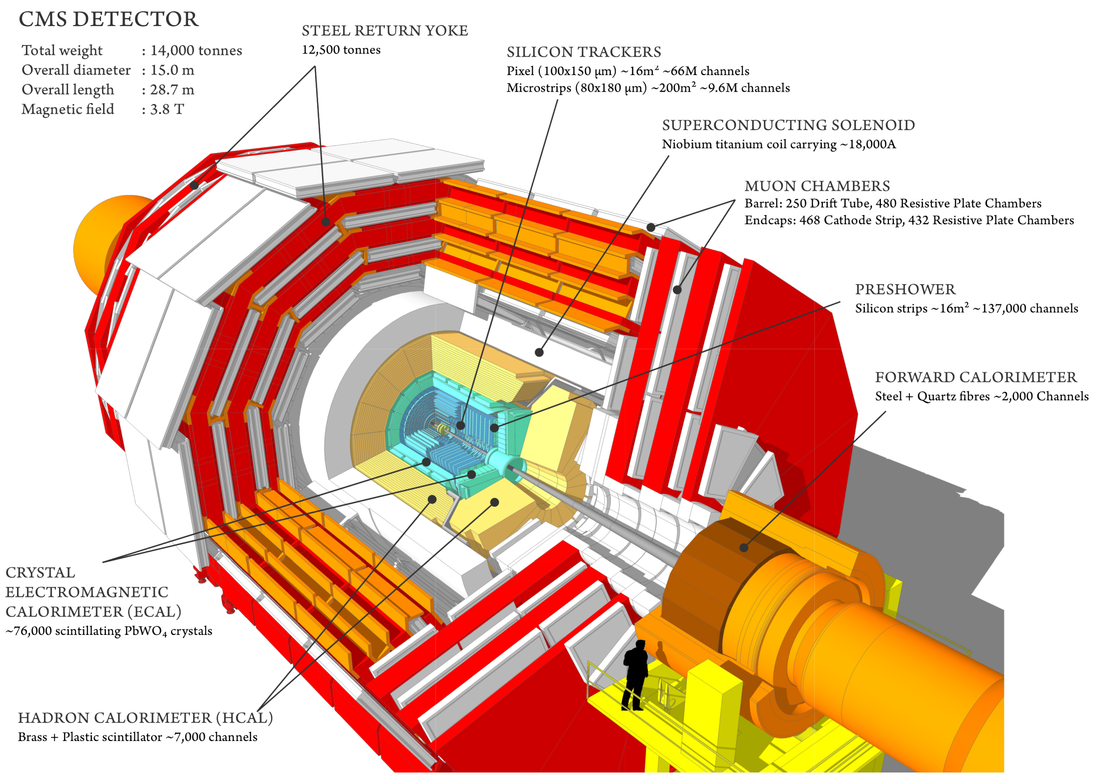

CMS DETECTOR
Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T
STEEL RETURN YOKE
12,500 tonnes
SILICON TRACKERS
Pixel (100x150 μm) ~16m² ~66M channels
Microstrips (80x180 μm) ~200m² ~9.6M channels
SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying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

Figure 2: Compact Muon Solenoid cross-section, showing the layered structure of the detector.

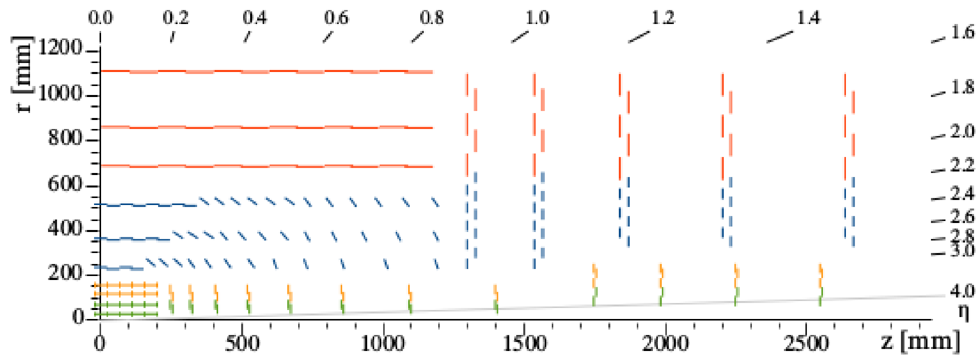


Figure 3: Quarter view of the silicon sensor placement in the Phase-II tracker. The axis z is placed along the beam. The perpendicular $r\phi$ plane is cylindrically symmetric in the ϕ coordinate. The barrel of the tracker extends to $z \approx 1250$ mm, where the end caps region starts. Orange and green lines represent pixel modules. Blue lines are PS modules and red lines are 2S. [3]

1.2.2 Tracker

The tracker refers to the innermost subdetector close to the interaction point. Modules containing silicon sensors are placed in concentric layers recording hits at multiple points for charged particles. The recorded data is used to reconstruct the particles' full tracks, from which the momentum is estimated.

Measuring momentum and energy is key to reconstruct a full, accurate picture of an event. Neutral particles such as neutrinos are inferred from missing momentum and energy. Hence the need of a very accurate tracking device. A breakdown of the silicon sensors placement and type in the target Phase-II tracker design is shown in figure 3.

At the core, sensors are segmented into microscopic pixel-channels that can be read-out individually. Each bunch crossing can produce multiple collisions, each producing multiple particles with distinct tracks. The track density is very high in the inner region, high resolution is thus needed to recognize separate tracks. As particles move outwards track density decreases. The required resolution is lower and sensors can be segmented into strips instead of pixels. There are three types of modules housing a combination of strip and pixel sensors depending on their proximity to the center. The inner tracker consists of pixel modules with a radial position of up to 200 mm. Modules outside of this region are part of the outer tracker. Pixels are partially replaced with strips in the Pixel-Strip modules (PS), while the outermost modules ($r > 600$mm) consist of two silicon strip (2S) layers.

In order to accurately reconstruct a particle's track, it needs to interact with a minimal amount of layers in the detector. For particles with a lot of forward momentum it can happen that a low amount of layers in the barrel is crossed. To avoid losing these, modules have also been placed in the end caps.

1.2.3 Trigger and Data Acquisition

At peak performance, around one billion proton-proton collisions per second occur inside the CMS detector. The amount of data generated is so huge it is absolutely unthinkable to extract and store all of it. A two-fold triggering system is used to quickly select worthy events [4].

The level 1 trigger (L1) is the first filtering stage operating at the hardware level. Based on relatively simple criteria the trigger reduces the original event rate of 40 MHz to 100 kHz only. The current L1 trigger is designed to decide based on information from the calorimeters and muon chambers only. Examples of criteria can be exceptionally high energies or muon count. For the Phase-II upgrade, the tracker is required to contribute to the L1 trigger. When data is considered interesting for read-out, a trigger signal is sent to all modules initiating transmission of all data.

The triggered data is directed to a computer farm where the high level trigger (HLT) occurs. The allotted processing time per event is significantly larger, allowing complex tests to further reduce the data down to about 500 selected events per second.

1.3 Phase-II upgrade to HL-LHC

2024 marks the end of LHC's "Run 3", where an estimated integrated luminosity of 300 fb^{-1} will have been acquired inside CMS. The LHC is then scheduled for a long shutdown of approximately two and a half years during which the accelerator will be substantially upgraded to accommodate for much larger luminosities. The scheduled update is referred to as "Phase-II".

The need for larger luminosities is motivated by rare and statistically limited processes of the Standard Model (SM) or Beyond Standard Model (BSM). Processes of interest are, for example, the SM Brout-Englert-Higgs (BEH) boson's Yukawa couplings to fermionic matter particles, or the BEH boson's self-coupling.

Evidently, CMS will have to be upgraded accordingly. The Phase-II Upgrade of the CMS Tracker is described in the Technical Design Report [3] and discusses the challenges to tackle for the upgrade of the tracker specifically. The most important ones being:

- *Contribution to the Level-1 trigger:* selecting promising events becomes increasingly difficult at high luminosities. CMS has opted for an increased trigger latency and the requirement of data contributions from the tracker. The trigger rate will also be increased from 100 kHz to 750 kHz.

- *Radiation tolerance:* radiation degrades the tracker over time affecting functionality. The current tracker is expected to be essentially dysfunctional by the end of 2024. The upgraded tracker is required to be fully efficient up to a target integrated luminosity of 3000 fb^{-1}.

- *Reduced material in the tracking volume:* the performance of the tracker and calorimeters is affected by the amount of material the particles encounter. Reducing the material budget will greatly benefit the exploitation of high luminosity conditions.

- *Improved granularity*: the channel occupancy - the number of hits compared to the total amount of channels - must remain low to ensure efficient tracking. Since more collisions per bunch crossing will happen and raise the occupancy, the resolution of the tracker must be improved.

- *Improved pattern recognition and two-track separation:* the current tracker has limited track finding performance in high pile-up environment, i.e. a large amount of interactions per bunch crossing. Notably at the level of the HLT, algorithms need to be improved to handle the increased rate of complex incoming data.

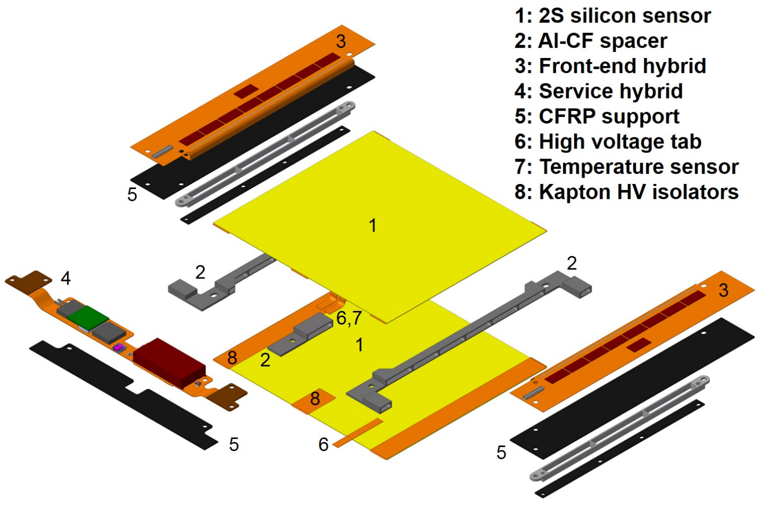


Figure 4: Exploded view of the 2S module design [].

2 2S Module

The Two-Strip (2S) modules are electronic units designed to detect and handle ionizing particle hits in the CMS outer tracker. They are made out of various components responsible for specific tasks in the data acquisition chain. Figure 4 shows an exploded view of the module, labelling each component. A key aspect is the very modular design, leaving lots of freedom for the individual components.

The defining feature of the module are the central strip silicon sensors with active areas of 90 cm^2. Strip sensors are silicon wafers segmented into channels along one axis in the plane, as a result they have changing resolution depending on the of angle of incidence. The requirements on resolution are less strict in the outer parts of the tracker compared to the core, where sensors are segmented in both directions of the plane into microscopic pixels. The silicon sensor technology is discussed in section 2.1. A new feature compared to current read-out modules is the addition of a second sensor layer. Information from both layers together allow for early, local track reconstruction, where the tracker will provide data to the L1 trigger.

The front-end hybrids, electronic circuits consisting of multiple components, labelled on figure 4 are read-out hybrids wire bonded to the sensors. Eight read-out CMS binary chips (CBC) and one concentrator chip (CIC) are bump-bonded to the hybrid, connecting each CBC to 127 sensor channels of the top layer and 127 channels of the bottom layer. The concentrator chip aggregates and formats the data from all 8 read-out chips. This is the goal design of the hybrids, in practice hybrids with varying amount of chips are used for the purpose of testing the chips, hybrids and module functionality. No hybrid with a CIC is available since the chip is not yet produced.

A service hybrid (SEH) is also present hosting all services to and from the counting room, supplying low and high voltage to the chips and sensor respectively, and acts as the I^2C[1] master of the module. Additional chips providing monitoring functionalities, such as temperature readings, are also present on the SEH. The lp-GBT chip, for example, converts electric transmission to optical. While the CIC already provides a data formatting stage, a second one is performed at the level of the SEH merging the data from the left and right CIC. Like the CIC, the SEH is also not yet produced.

2.1 Silicon strip sensor

Silicon sensor technology makes use of the semi-conductor properties of silicon. As in all solid state materials, a band gap of forbidden energy levels separates the valence band (VB) from the conduction band (CB). The occupation of states of energy E in these is determined by the Fermi-Dirac distribution:

$$f(E) = \frac{1}{1 + exp^{\frac{E-E_f}{k_B T}}}.$$

Where the Fermi energy E_f is the energy level where half of the available states are occupied, k_B is the Boltzmann constant and T the temperature. At temperatures of 0 K, all states below E_f are fully occupied, while all (available) states above are left empty. For semi-conductors, the band gap and Fermi energy are such that the VB is full at 0 K, while the CB is left empty.

Acceleration of charges in a band requires accessible, higher energy states. In this regard, semi-conductors at 0 K are isolators. When raising the temperature however, some of the electrons from the VB can jump to the CB. The electrons in the CB have lots of free energy states, they are suitable for conduction. While jumping to the CB they also left an open state in the VB, a so-called hole h^+ that can be modelled as a positively charged particle. This particle also has available energy levels in the VB and contributes to the electric current in the same direction as the electrons in the CB.

Another way of increasing conductivity is to add donor and acceptor impurities to the silicon crystals. Elements with a valence electron more (donor) or less (acceptor) are introduced with energy levels in the band gap of the semi-conductor. The occupation of these energy levels is also determined by the Fermi-Dirac distribution. When unoccupied, the electron (hole) of the donor (acceptor) doesn't

[1]I^2C is a communication protocol adopted into the design of the CBC read-out chips, further discussed in section 2.2

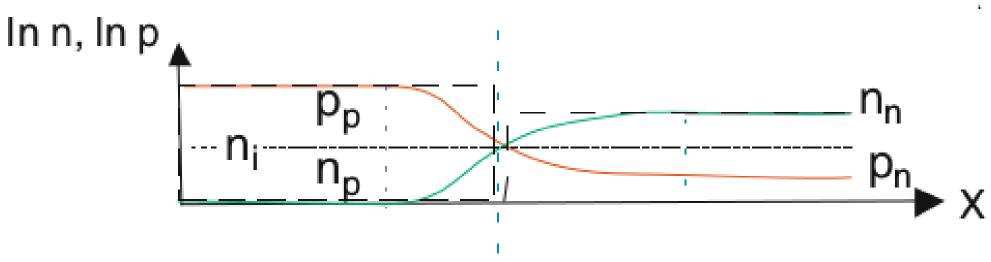


Figure 5: Log scale of the charge densities in a pn junction. [5]

magically disappear, it has moved to an energy level in the VB or CB. The intrinsic occupation distribution is disturbed and this is reflected in the changing Fermi energy when adding impurities. The end result is an increase in free charges, electrons for donors, and holes for acceptors. Substrates with a majority of donors is referred to as n-doped. A majority of acceptors gives a p-doped substrate.

Charged particles from interactions inside the CMS detector intersect with silicon sensors, depositing energy and creating electron-hole pairs. In an intrinsic silicon substrate the size of a standard silicon sensor used in high-energy physics there are $\sim 10^9$ free charge carriers but only $\sim 2 \times 10^4$ generated electrons induced by an ionizing particle [5]. In order to measure the charges induced by the ionizing particle a technique is needed to remove the natural charges present in the semiconductor.

2.1.1 pn-junction

A pn-junction is formed when an n-doped silicon substrate is brought in contact with a p-doped substrate (Figure 5). When contact is established, charges freely flow from one substrate to another to form a new thermal equilibrium. Diffusion moves the electrons from the n-substrate to the p-doped one, leaving behind a (static) positive ion from the doping procedure. Similarly, holes move to the n-substrate leaving behind negative ions. The electrons (holes) move to a region with absence of donors (acceptors) and a surplus of holes (electrons) in the VB (CB). Recombination of electron-hole pairs will spontaneously occur, effectively eliminating free charges.

The ions which are left behind form an electric field which slowly builds up and stops the natural diffusion of the free charges. The result is a substrate with a p-doped region and an n-doped region separated by a region depleted of free charges. A considerable depletion region can not realistically be obtained by diffusion alone. By applying an external voltage, in reverse-bias, the depletion region can be increased up to covering the full sensor thickness.

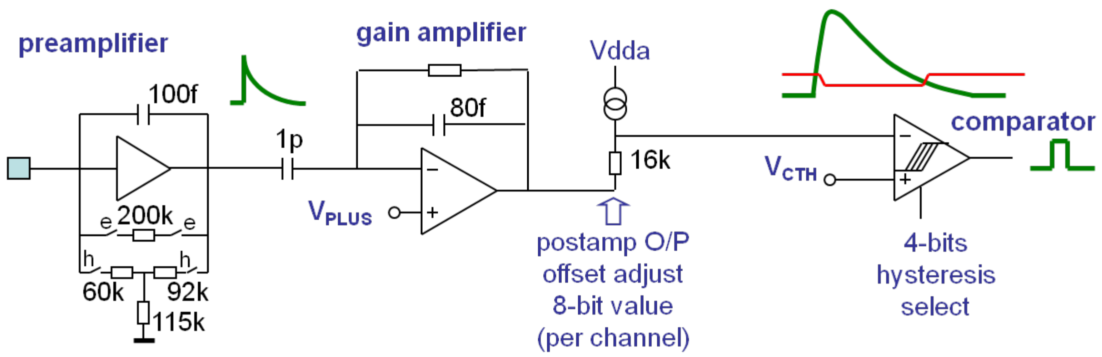


Figure 6: Front-end circuit of the CBC2. A circuit is available for each channel, integrating charges from the sensor to produce an analogue signal proportional to the number of charges collected. The signal is then digitized in the comparator when exceeding a certain threshold. [6]

2.2 CMS Binary Chip 2

2.2.1 Introduction

The CMS Binary Chips are digital, read-out chips for the outer tracker 2S modules, each connected to 254 channels of silicon strip sensor The design choice of the chips aims at reducing the amount of data that needs to be transmitted compared to the analogue predecessor. More importantly it needs to fulfil the demanded requirements of the tracker upgrade in view of the HL-LHC, one of which is the contribution of the tracker to the L1 trigger.

The CBC is currently still in R&D phase and there are three different versions in circulation with slightly different architectures, each one a progression of previous versions. The version of interest for this thesis is CBC2, but occasional comments will be made about the next generation CBC3. The following sections describe the general features of the CBC2 [6].

2.2.2 Analogue Front-end

The front-end of the chip (Figure 6) handles charge collection from the sensor and digitization of signals over threshold, for each channel independently. The first stage of the circuit collects and integrates charges from the sensor at a rate compatible with the bunch crossing rate of the accelerator. An analogue signal with amplitude proportional to the amount of collected charge is produced and amplified in a second post-amplifying circuit logic.

The signals leaving the post-amplifiers are adjusted independently with an offset bias, and then fed to a comparator which will send out a digital ’1’ when

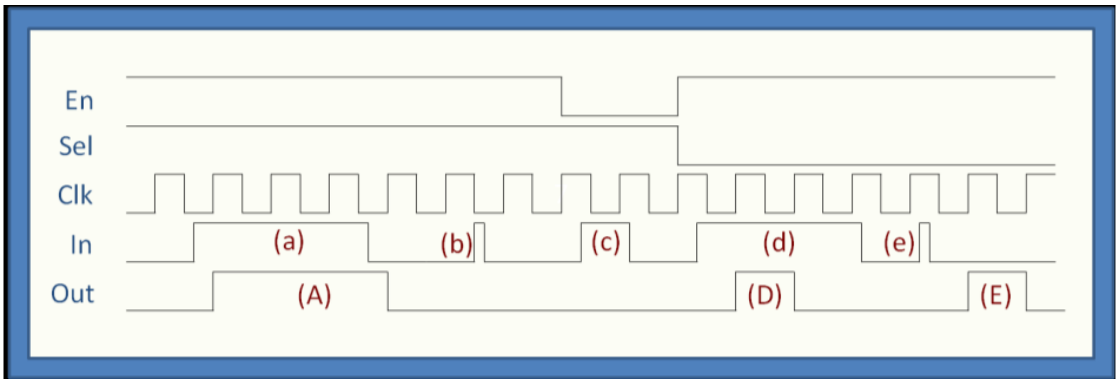


Figure 7: Depiction of the hit detect logic synchronizing input from the comparator to the clock. The signal *Sel* chooses the synchronization mode, *En* completely disables the logic when low. [6]

the incoming signal exceeds a selectable threshold V_{CTH}. By design, the post-amplifiers emit a continuous baseline signal V_{plus}, any input is added on top of the baseline. What the comparator really measures then is signals exceeding the difference V_{CTH} - V_{plus}. When designing CBC2 it was not yet fully decided what type of sensor would be used, so the option of operating in different polarities - hole mode or electron mode - is available. V_{plus} and V_{CTH} need to be adjusted accordingly. In the next iteration, CBC3, the operating mode is set to electron mode.

2.2.3 Digital logic

The chip logic is fully binary after the comparator stage. Hit detect logic is required to synchronize the comparator output to the clock. Two modes are available where each signal over threshold produces a binary "1" for the duration of either exactly one clock cycle, or the duration over which the signal exceeded the threshold (Figure 7). Data is then sent down two different paths: a stub finding logic circuit and pipeline RAM. Stubs are the data format that is ultimately transmitted to the L1 trigger and are discussed in section 2.2.6.

The pipeline RAM stores the channel content of the last 256 bunch crossings. It is 254 bits wide, matching the channel count, and 256 cycles deep allowing for a trigger latency up to 6.4 μs (@40MHz). Read-out is controlled by the pipeline control logic; a selectable delay corresponding to the trigger latency determines which entry is to be read when the read-out is triggered. Data is then transferred to a 32 cycles deep buffer RAM where the data is serialised. The buffer is needed as the transmission takes 270 40 MHz clock cycles (Figure 8), while multiple triggers can occur during it.

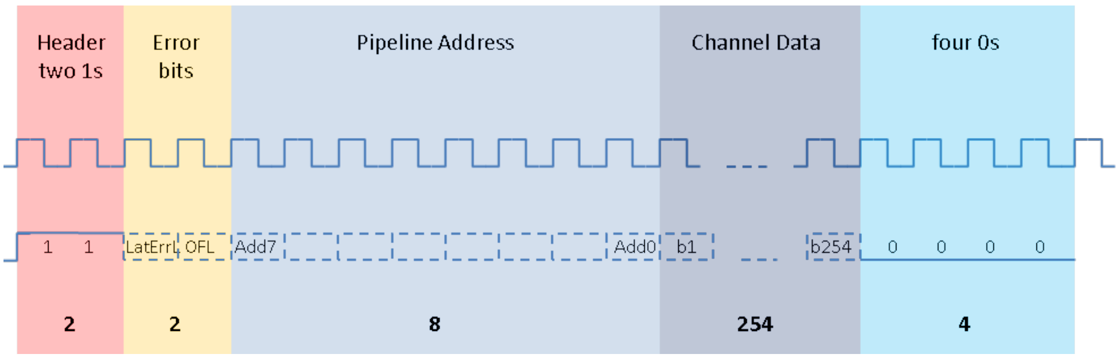


Figure 8: Data format in the buffer RAM. The first two bits are invariantly "11" forming the header of the data. They are followed by two error bits indicating trigger latency mismatch and buffer RAM overflow. Then comes the pipeline address followed by 254 bits of channel content. [6]

2.2.4 Slow Control

The CBC has a total of 511 configurable 8-bit registers. The function of the register entries varies from adjusting parameters ($V_{CTH}, V_{plus}, ...$) to choosing operating modes (polarity, hit detect logic, ...) or enabling certain features (test pulse). A communication protocol with the CBC2 is needed to programme the register, therefore an I^2C [7] architecture was chosen.

The component driving the transaction is referred to as the I2C master, and can communicate with many slaves. In this case the slaves correspond to the register entries and are each identified by a unique address. The master generates the clock and sends a start signal initiating back and forth communication between master and slave in a specific format. Data is sent bit by bit over a single common bus line. Figure 9 shows a write (as opposed to read) transaction. Both the master line (SDAM) and slave line (SDAC) connect to the common data line (SDA). When the I2C is inactive the clock and data lines are high. A transaction is initiated by taking the data line low while the clock line is high. The clock then starts running and all subsequent data must be sent when the clock is low. The first 7 bits correspond to the chip address followed by an 8th bit, "0" for a write transaction and "1" for a read. In the case of the CBC all chip addresses are covered within 5 bits, so the first 2 bits are hard-wired to "10". The master component then releases the data line and waits for the slave to acknowledge the reception of data. The register address and the data written to the register then follow in a similar format. Finally, the transaction is ended with a stop signal with the master line going high when the clock is high. Read transactions are similar although somewhat longer. They start as a write transaction with write/read bit set to 1, but the transaction stops after acknowledging reception of the chip and

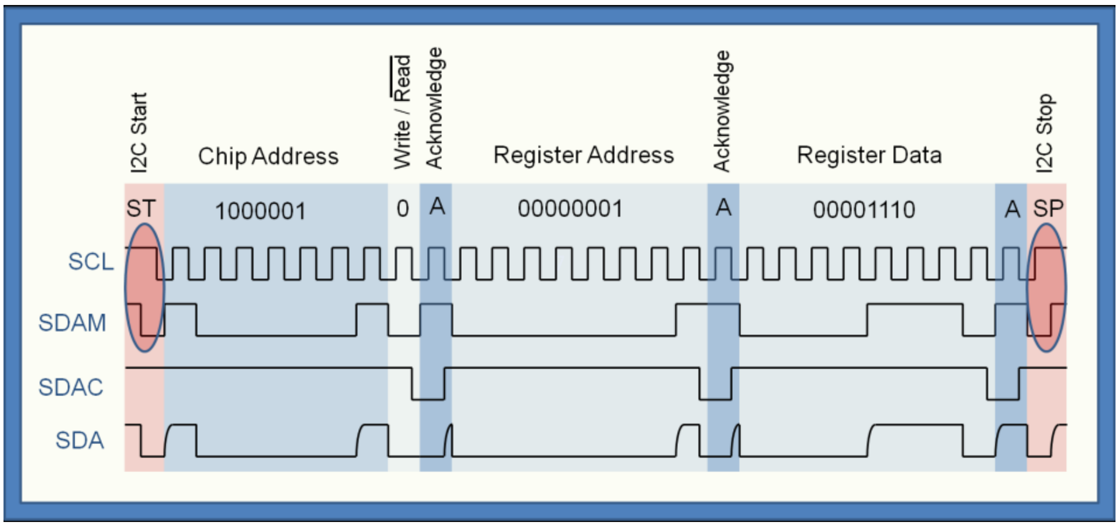


Figure 9: An example of an I2C write transaction. The master line SDAM and slave line SDAC are 'high' by default. They in turn grab the common SDA data line following a set communications protocol in order to reprogramme a user specified register entry. [6]

register address. A new transaction is now started with chip address and read bit set to zero. The slave now drives the data line and sends its content to the master (fig. 10).

An advantage of the I^{2}C architecture is the simplicity of the pin connection between master and slaves. Only 2 lines, the data and clock lines, are required for any number of slaves. The speed of transactions is however limited by this design, data is sent bit by bit over a single line, and chip and register addresses have to be repeated every time. In the example above 28 clock cycles were needed to write 8 bits of data. A read transaction is slightly longer with 36 cycles.

It is worth noting that 8 bits for register addresses only covers 256 unique addresses, whereas a total amount of 511 was mentioned earlier. The solution implemented into the CBC is a paging system. Register 0 has a special unique status and is always accessible with regular I^{2}C communication. The most significant bit (MSB) of this register acts as a page, a 9th bit for all other addresses. Registers on different pages can have the same address, but the chip will know which one to access. For example, V_{plus} and $V_{offset,ch11}$ both have the same register address "0000 1011". If one wishes to read both of these, and the initial active page is zero, one should first send a read request for register address "0000 1011". This will return the current value of V_{plus}. A write transaction to register "0000 0000" is then necessary, setting the page to 1. The following read request for address "0000 1011" will now give the value for $V_{offset,ch11}$. The downside of this paging system

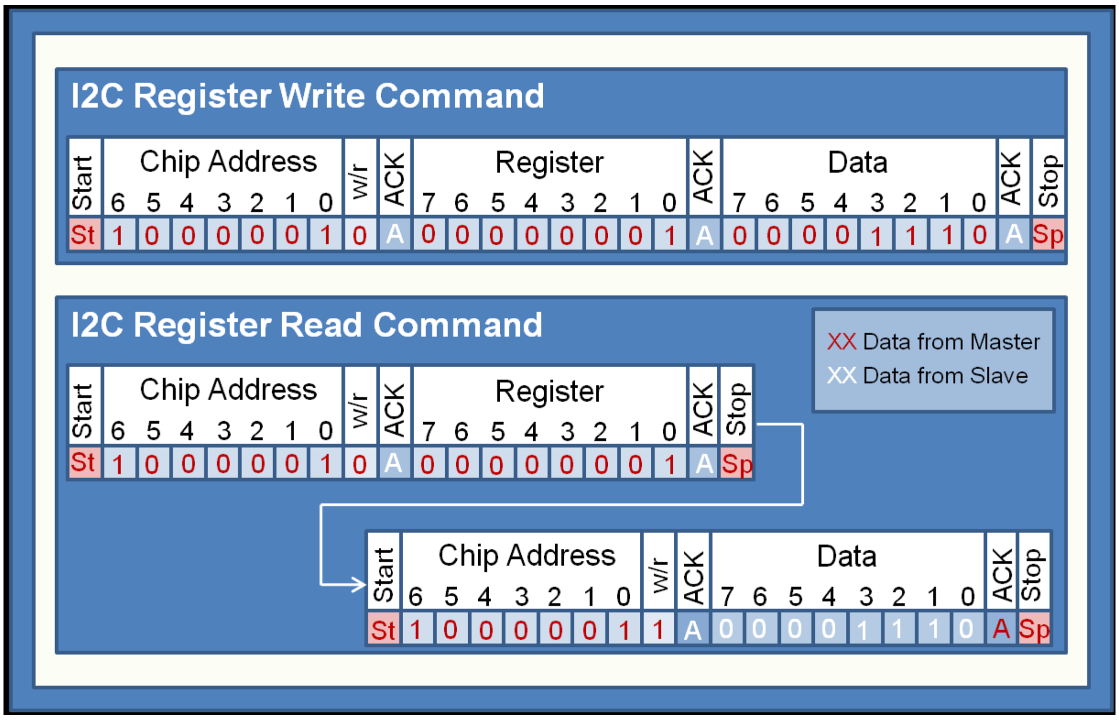


Figure 10: Schematic of both read and write I^{2}C transactions. [6]

is the requirement to read which page is active for every transaction, slowing down an already slow process.

2.2.5 Fast Command

Some actions need to be communicated quickly to the CBCs in cases where timing is critical. CBC2 has 4 lines operating at 40 MHz, each dedicated to the transmission of a specific signal. Among these is the line sending trigger signals to the pipeline RAM, activating read-out of a specific bunch crossing.

In the next version, CBC3, the fast commands are transmitted over a single line operating at 320 MHz. The 8 bits per 40 MHz clock cycle allow for an increased number of fast commands.

2.2.6 Stub logic

One of the set goals of the tracker upgrade is to include tracker data into the decision making process of the L1 trigger. The CBC2 identifies and sends out the presence of so-called stubs for each bunch crossing. Stubs are identified locally in each module in a two stage check.

In a first instance the stub logic looks for valid clusters of hits in the bottom and top sensors of the module. High p_T tracks come in perpendicular to the modules, while low p_T tracks bend and cross the sensors at an angle. Cluster width discrimination then analyses adjacent hits in the sensors and rejects wide clusters associated to low p_T tracks. The maximum cluster width can be set by writing to a register via I^{2}C.

The Φ-shift - the polar angle in the transverse $r\phi$ plane perpendicular to the beam and magnetic field - and correlation logic starts in the center of a valid cluster in the seed (bottom) layer. It then looks for a valid cluster in the (top) correlation layer in a coincidence window with programmable width and position relative to the seed cluster. The coincidence window is another manifestation of a p_T cut, where low p_T tracks are expected to fall outside of the window because of the larger bend of these particles in the magnetic field. However, the relationship of the window width and p_T cut depends on the position of the module and separation of the layers. In the $r\phi$ plane, charged particles in the magnetic field have circular trajectories. Equating the centripetal force to the Lorentz force gives an expression for the radius of curvature ρ of the trajectory of a particle with mass m and charge q, in a magnetic field B:

$$\begin{aligned} \rho &= \frac{mv_T}{qB} = \frac{p_T}{qB}, \\ \Rightarrow \quad p_T &= qB\rho. \end{aligned}$$

Only the components of the velocity or momentum transverse to the magnetic field need to be considered. The relationship between the R coordinate and the (change in) angle ϕ of a recorded hit with track curvature ρ is derived from figure 11.

$$\begin{aligned} \frac{R}{2} &= \rho \sin(\phi), \\ \Rightarrow \quad p_T &= \frac{R}{2} \frac{qB}{\sin(\phi)}. \end{aligned}$$

For a pair of closely separated hits the above expression becomes [8]:

$$p_T = \frac{qB}{2} \frac{\sqrt{R_1^2 + R_2^2 - 2R_1R_2\cos(\Delta\phi)}}{\sin(\Delta\phi)}.$$

To assure a uniform p_T cut over the whole tracker, the coincidence window width should be chosen with previous relation in mind depending on the position of the detector. The 2S modules also come in two variants with different layer separations.

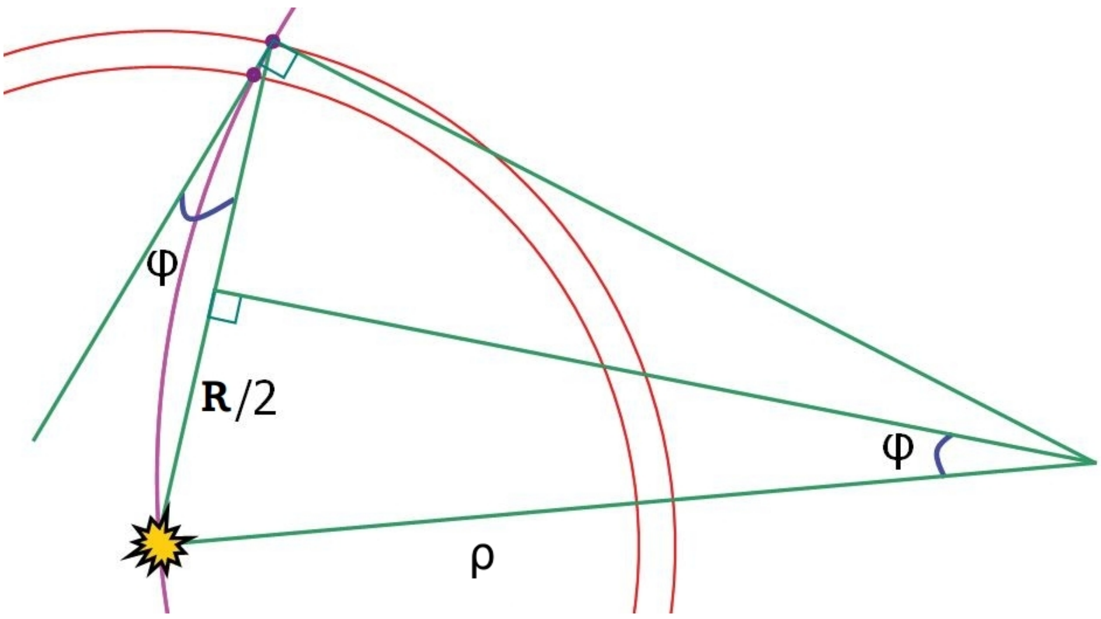


Figure 11: Relationship between the track curvature radius ρ, and the $r\phi$ coordinates of a recorded hit.

If all conditions are met, a stub is found. The CBC2 shares the observation of stubs the L1 trigger over a single line at bunch crossing rate. CBC3 has five lines dedicated to informing L1, allowing for a more detailed account of the stub features.

I_1	I_2	O
0	0	0
0	1	0
1	0	0
1	1	1

(a) **AND**

I_1	I_2	O
0	0	0
0	1	1
1	0	1
1	1	1

(b) **OR**

Table 1: Logic table of operators. The I_i are separate inputs forming an O(utput) depending on the logic operator applied.

3 Field Programmable Gate Array

3.1 FPGAs

A field programmable gate array (FPGA) is a set of electronic circuits designed to be reconfigurable after manufacturing. The FPGA consists of many individual configurable logic blocks (CLB), which can be interconnected in any way specified by firmware design, usually with the aid of Hardware Description Languages (HDL). This allows to create large, complex circuitry by combining simple logic circuits.

To understand the exact composition of these building blocks a short reminder on circuit logic is proposed. The most fundamental elements are logic gates: small circuits which generally take 2 inputs and have one output, and behave like mathematical logic operators. The **AND** logic gate is defined by the logic table 1a. There are many ways to implement this as an electronic circuit, but the most common way is to use a transistor. One of the inputs acts as a switch and lets the other input through the transistor when the switch is closed, and blocks it when the switch is open. The only way to have an active output is to have both inputs active. Another example is the **OR** logic gate (tab. 1b), which requires only one of the inputs, or both, to be 1 for an output of 1. By combining logic gates it is possible to construct larger circuits with complex functionality.

Multiplexers have several input lines and a selection signal. The selection signal determines which input is forwarded to the single output. The number of input lines is unrestricted, while the number of lines for the selection signal is constrained by the number of input lines. The reverse of a multiplexer is a demultiplexer, where a single input line is forwarded to one of many output lines decided by a selection signal.

Latches are components that can remember the state of an input. A typical example is the Set-Reset (SR) latch: when the input S is 1, the state and output of the latch becomes 1. This does not change when S goes back to zero. The input R can reset the latch to 0 by feeding an active input, but similarly can't put it back to state 1.

LookUp-Tables (LUTs) are similar to multiplexers, but the input lines are static. The static values are either hard-wired or can be programmed with a set of latches.

Serializers/Deserializers provide a way to transform parallel data to serial and vice-versa. Logic circuits have the ability to manipulate a finite amount of data at the same time (parallel). Sometimes however the data needs to be shared over a single line, bit by bit. This is where serializers come into play.

Adders and multipliers circuits perform binary addition or multiplication.

Again these circuits can be combined to form larger structures. Digital Signal Processors (DSPs) combine multiple adders and multipliers to perform fast mathematical operations. The DSP is said to be programmable, i.e. the function of the circuit is not set in stone at the manufacturing stage. A set of multiplexers allow to select the behaviour of the circuit. With the same DSP one can for example perform the operation $R = A + B + C + D$ but also $R = A * B - C * D$.

FPGAs contain various general-purpose combinations of components called slices. On the Xilinx Kintex-7 FPGA, used on the FC7 boards[2], basic slices are made from a combination of 8 LUTs, 16 flip-flops (latches), 2 arithmetic and carry chains (adders and multipliers), 256 bits of distributed RAM and 128 bits of shift registers (serializers) [9]. Each slice can be programmed to have wildly different functionality. A logic block consists of two slices, connected to a switch matrix which controls the interconnections between slices (Figure 12).

3.2 Hardware Description Language

Depending on model number, a single Kintex-7 FPGA can have up to 74.650 slices [9]. It is unthinkable that the user would have to configure every slice manually and provide connections between them. In practice the firmware is designed by use of a Hardware Description Language, which generates the hardware design during compilation. The language used in this thesis is VHDL (VHSIC Hardware Description Language). The typical lay-out of a VHDL file looks as follow:

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

[2]Electric boards compatible with μTCA standard, see section 4.1

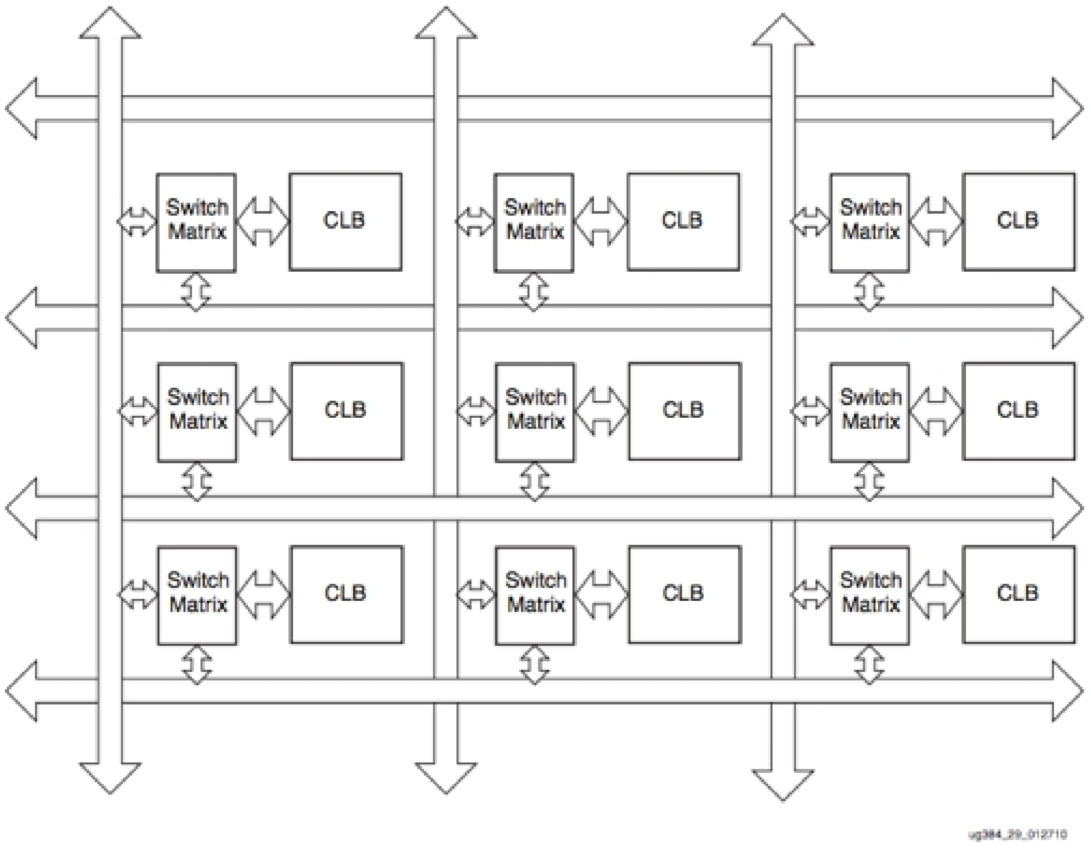


Figure 12: Logic blocks connected to a switch matrix. The matrix defines the connections between logic blocks, essentially creating a circuit from programmable building blocks.

```
entity basic_module is
port(   first_input         : in std_logic;
        second_input        : in std_logic;
        first_output        : out std_logic
        );
end basic_module;

architecture Behavioral of basic_module is

        signal temp : std_logic;
begin

        temp <= first_input or second_input;
        first_output <= temp;
end Behavioral;
```

The entity is the name of the (text)file, and also the place where in and output of the file can be declared. At the top level these signals can be thought of as physical, accessible pins. When the design is completed and translated to hardware components, the user can feed signals to those pins and measure the output. *std_logic* denotes the type of the signal, it can either be “1” or “0”, representing a binary signal.

The internal logic is described in the architecture. Additional internal signals can be defined to help creating the logic, or store values. In this particular case, the signal *temp* behaves like an **OR** operator acting on both the inputs. It also forwards the signal to the output.

An important distinction with common programming is the order in which these statements are evaluated. Usually the program is read top to bottom, evaluating the value of *temp* first before passing it to the output. In VHDL however each statement is permanent and are understood to function at the same time. This makes sense when translating the code to hardware. Once the electric circuit design is set, it doesn’t change and stop behaving like it was described to. In this sense designs are rather static and don’t lend themselves to step-by-step procedures.

Adding dynamic to the circuit is solved with the use of a clock signal and latches. Clock signals are periodic block signals oscillating between a high and low state (binary 1 and 0 respectively). The transition between low to high state is called the rising edge, high to low is the falling edge. By combining a set of latches and a clock in a particular way it is possible to store data that refreshes on every transition of the clock. This allows to introduce sequential logic in the design, and a sense of timing. In VHDL this is done with the use of a process environment:

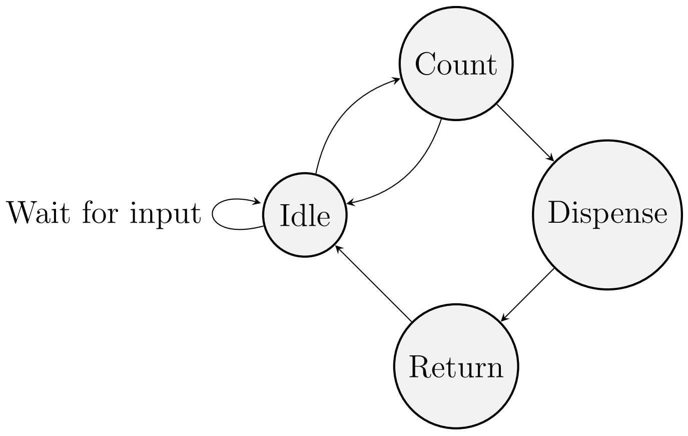


Figure 13: State machine of toy soda dispenser.

```
process(clock_input)
begin
        if rising_edge(clock_input) then
                -- insert logic here
        end if;
end process;
```

The content of a process is evaluated sequentially, e.g. when multiple statements assign a value or logic to the same signal, only the very last assignment will be considered. With the use of `if` conditions signal values and behaviours can be modified depending on timing or output of a certain signal. This leads to the very important concept of a state machine.

3.2.1 State machines

State machines are ways to describe sequential behaviour in firmware design. The idea is to define a set of states each describing a certain signal behaviour and navigation between each other. An illustration probably tells more; consider a soda machine accepting money. When a certain amount of money is exceeded, a soda can is dispensed and excess money is returned. Diagram 13 expresses this scenario in terms of a state machine.

The machine starts in the *idle* state, which points to itself waiting for user input. In this simplified example the dispenser only takes 0.50 eur coins, a soda costs 1.20 eur. When a coin is inserted, the state machine moves to the *count* state. It adds 0.50 to an internal counter signal (which was previously at zero). Here two paths are available. Unless the counter exceeds 1.20 the machine points back to idle, waiting for additional coins. Otherwise it moves to the *dispense* state instead. Here an output signal is set to “1” activating the dispensing mechanism, and 1.20 is subtracted from the counter. The next state returns the remaining money and sets the counter to zero. More importantly maybe it also resets the dispensing signal to “0”. Finally the state machine returns to *Idle*, waiting for its next client.

State machines are easily implemented in VHDL with the use of (clock triggered) processes, IF/ELSE conditions and an enumeration type. The enumeration type will represent states, and the IF/ELSE conditions are used to navigate.

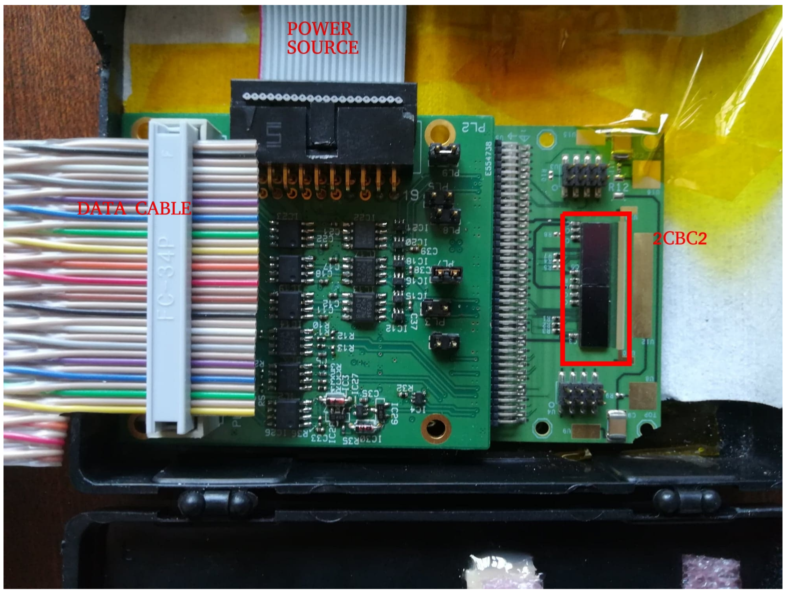


Figure 14: Top view of an uncoupled 2CBC2 hybrid.

4 Set-up

While the target design of the 2S modules is ready, no fully assembled unit currently exists. The concentrator chip hasn't even been fully designed yet. There are however parts and/or prototypes available of the 2S module in the labs at IIHE.

4.1 μTCA and FC7

Micro Telecommunications Computing Architecture is an open standard computing specification created by PICMG [10] and adopted by the CERN collaboration. For the purpose of this thesis, the μTCA crate acts as a hub for FC7 boards, providing Gigabit Ethernet connections to them.

FC7 boards house the Kintex-7 FPGA to which the firmware design is uploaded. It slides into the μTCA crate connecting the backplane, where it makes the Ethernet IPbus connection. Two slots are available for connecting external devices, such as a 2CBC hybrid (Figure 15).

4.2 Uncoupled 2CBC2 hybrid

The uncoupled 2CBC2 hybrid is a set of 2 chips bump-bonded to a hybrid, other components from the 2S design are not present yet. A picture is shown in figure 14. Two connections are available, one for the power supply and the other for external communication. The hybrid is connected to an FC7 board as displayed

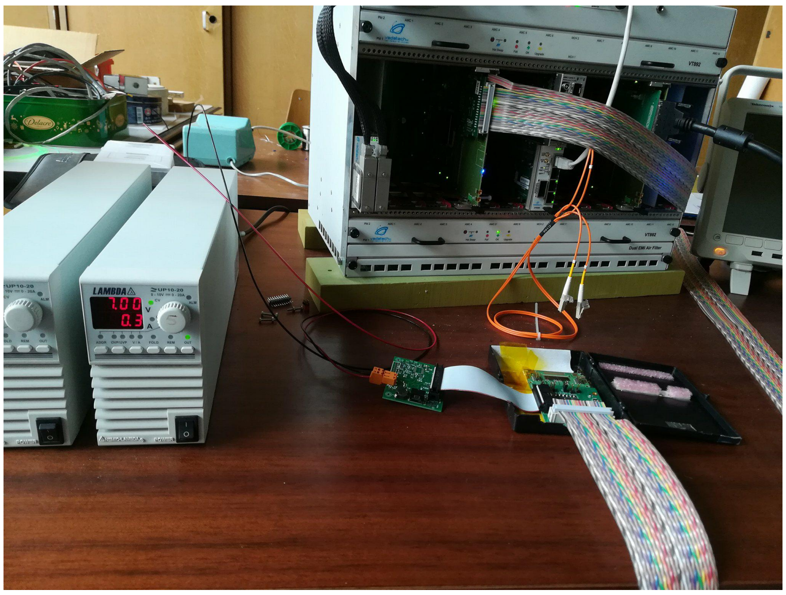


Figure 15: In the black box lies the 2CBC2 hybrid connected to a power source and twisted pair IDC data cable. A better view of the black box content is provided in figure 14. On the left the power source is seen operating at 7 V connected to a custom voltage divider board distributing the voltage over several pins. In the back stands the μTCA crate with FC7 boards, to which the hybrid is connected.

in figure 15. In this particular set-up the power is supplied by a single low voltage unit at 7 V, connected to a custom designed board distributing the voltage over several pins.

The same hybrid will also be tested connected to a different power supply. Here the power is supplemented by 3 distinct power units set at the correct voltage of 5 V, 3.3 V and 1.25 V. The units are wired to the relevant pins of the power connector (Figure 16).

4.3 Prototype 2CBC2 mini-module

Figure 17 shows a top view of the 2CBC2 mini-module, which consists of a 2CBC2 hybrid wire-bonded to a (mini-)sensor. The mini-module allows to test the full data acquisition chain with the use of a radioactive source and scintillators emulating a trigger signal.

The set-up for the 2CBC2 mini-module is very similar to the 2CBC2 hybrid set-up. The module supports two CBC2 chips which need powering from the same power units used previously (fig. 16). There is however an additional connection required to supply the high voltage power to bias the sensor.

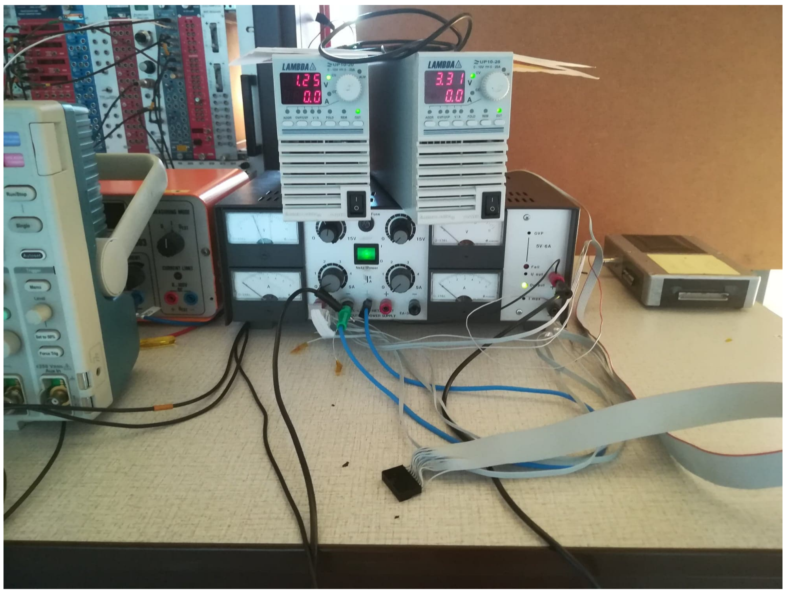


Figure 16: Alternative power source. This one is used because the custom board from figure 15 is not compatible with other set-ups. In front of the three power units lays the black power connector. The box to the right of the power supplies shows the closed container of the 2CBC2 mini-module.

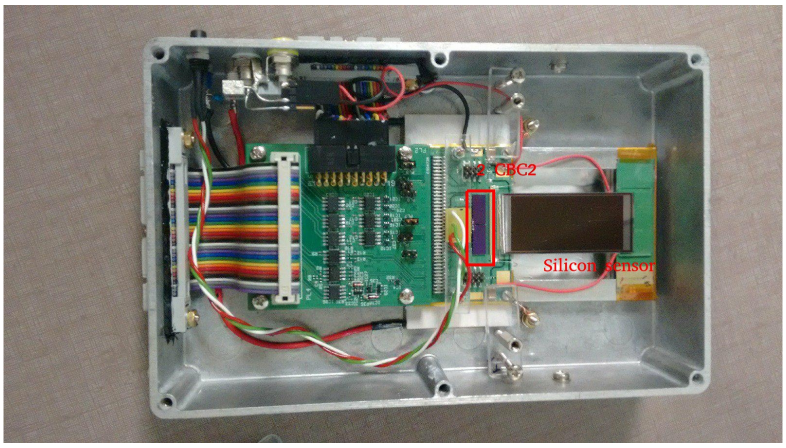


Figure 17: Top view of the prototype 2CBC2 mini-module. The sensor is wire-bonded to the hybrid underneath. When operating, the case is closed, blocking out any light to interfere with the sensor.

5 Calibration

5.1 Motivation

In the analogue front-end of the CBC2, each channel reaches a comparator stage where input signals greater than a selectable, global threshold V_{CTH} produce a digital “1”. The input to the comparator is the sum of a constant, global base-line V_{plus} and signals from the post-amplifier V_{signal}. The constraint to produce a digital “1” is then

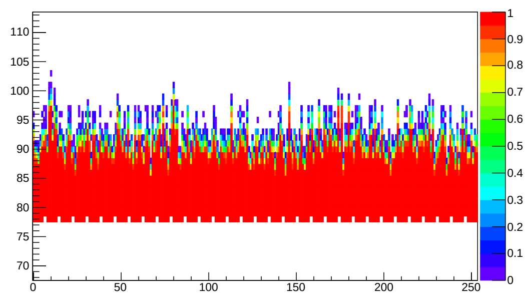
$$V_{plus} + V_{signal} > V_{CTH}.$$

Due to manufacturing limitations, the output of the post-amplifier V_{signal} feeding the comparator can be subject to variation from channel to channel. Since the threshold and base-line is set globally (I.e. for all channels) a third parameter V_{offset} is available for each channel independently, adding voltage to the input signal of the comparator. Digitization now requires to fulfil following inequality:

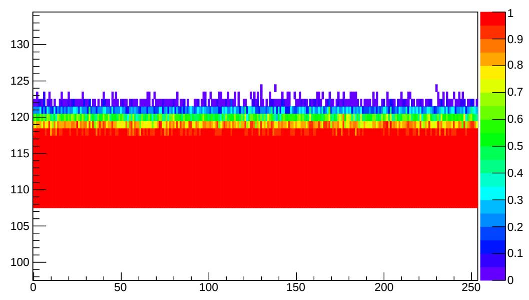
$$V_{plus} + (V_{offset} + V_{signal}) > V_{CTH}.$$

Offsetting the base-line compensates for the variations in V_{signal}.

For a chosen target threshold (V_{CTH}) value, the calibration procedure measures global occupancies to determine a global baseline (V_{plus}) with dynamic range. After finding a suitable value, occupancies of each channel are measured to tune the baseline (V_{offset}) independently such that V_{CTH} corresponds to 0.5 occupancy. The result of the calibration is illustrated in figure 18.



(a) Before calibration.



(b) After calibration.

Figure 18: Pedestal alignment before and after calibration. The x-axis is channel number; y-axis is expressed in V_{CTH} units; and colour scale is the measured occupancy.

Figure 18a shows an S-curve measurement of each channel with default V_{plus} and V_{offset} values. The misaligned transition regions from high to low occupancy indicate varying effective thresholds for each channel. Figure 18b displays the same measurement taken after calibration of the chip with a target threshold of 120 in DAC units, performed by the procedure outlined in section 5.3. The transitions are now aligned and the meaning of the threshold is the same for all channels.

5.2 S-curves

The occupancy (occ.) of a channel is defined as the number of recorded hits divided by the total amount of measurements taken (which is equal to the number of triggers sent):

$$\text{occ.} = \frac{\#\ \text{hits}}{\#\ \text{triggers}}.$$

For clarity a distinction will be made between a local channel occupancy, and the global CBC-occupancy. The channel occupancy is measured for one specific channel and is defined exactly as above. Taking the mean of all channel occupancies gives the global occupancy, it is usually determined by counting all recorded hits and dividing by the amount of triggers multiplied with the number of channels:

$$\text{global occ.} = \frac{\text{total}\ \#\ \text{hits}}{254 \times \#\ \text{triggers}}.$$

The result of an occupancy measurement will evidently depend on the selected threshold. For low thresholds, every trigger will record a hit due to noise inherent to electronics. At high thresholds no signals will pass and the occupancy is inevitably zero. The region of interest is the transition from high to low occupancy. It can be obtained by performing a threshold scan: start measuring at low threshold and repeat after incrementally raising the threshold. The resulting curve is called an S-curve, illustrated in figure 19. The S-curve is an important tool to study noise. Indeed, measuring the occupancy at a particular threshold gives an approximation of the odds of finding a signal exceeding the threshold:

$$\text{occ.}_{V_{CTH}} = P(S > V_{CTH}).$$

When rewriting one recognizes the definition of the cumulative distribution function

$$1 - P(S > V_{CTH}) = P(S \leq V_{CTH}) = \int_{-\infty}^{V_{CTH}} f_S(x)dx.$$

As a result, simply deriving the obtained S-curve (with respect to V_{CTH}) will yield the signal distribution f_S of the noise in units of V_{CTH}.

The point of the curve where occupancy equals 0.5 is of particular interest because it determines the position of the curve. This point is referred to as the pedestal of the curve. It can be determined by measuring an S-curve and extracting

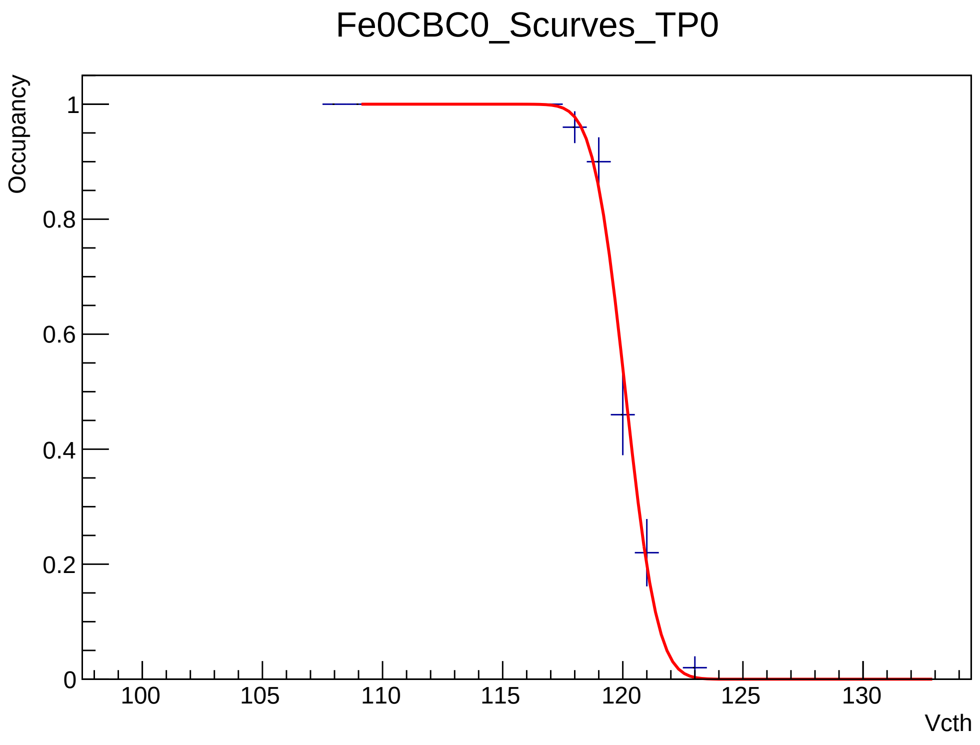


Figure 19: S-curve measurement on channel 42, from CBC # 1 on the 2CBC2 hybrid. The error bars are estimated according to a binomial distribution and the data is fitted with a properly normalized Error Function (Erf(x)).

the result from a fit with a sigmoid function. There are plenty of functions that model the shape of an S-curve, but one in particular is motivated by physics. If the noise is assumed to be normally distributed, the S-curve is then expected to be the cumulative distribution function (cdf) of the distribution, which can be expressed with an appropriately normalized Error function Erf(x):

$$\begin{aligned} Erf(x) &= \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} dt \\ &= \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2} dt. \end{aligned}$$

The cdf is:

$$\mathrm{cdf} = 0.5 + 0.5\mathrm{Erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)$$

The function has, like the normal distribution, 2 parameters μ and σ that can be evaluated by fitting the results. The pedestal is then simply the parameter μ. This leads to an obvious alternative method for determining the pedestal: the parameter μ of a normally distributed sample can be estimated by taking the mean. The sample of an S-curve measurement is not normal, but can be transformed to one as discussed above by differentiation.

5.3 Software calibration procedure

There currently exists a software procedure performing the calibration of the chip. The required slow control commands are sent from the computer to the FC7 boards via IPbus. There the necessary protocols are performed to write to the CBC and collect channel content, which is sent back to the computer. Occupancy is computed from the data and determines how to modify the parameters for the next iteration.

The method of determining an optimal value of V_{plus} efficiently, called binary bisection method, relies on a clever use of its binary representation, illustrated in the following pseudo-code:

```
void ExtractVplus(){
        byte Vplus = 0000 0000; // binary representation of Vplus
        float occupancy = 0;
        EnableTestgroup(-1); // enable all channels
        for(int bit = 7; bit >= 0; bit-1){
                Vplus(bit) = 1;
                Write(Vplus);
                occupancy = MeasureOccupancy();
                if (occupancy > 0.5){
                        Vplus(bit) = 0;
                }
        }
}
```

The idea is to start with the MSB set to 1 and the others left at zero (which is half the range of V_{plus}). By measuring the occupancy it is determined if the current value is too big or too small and the MSB is flipped or left unchanged accordingly. The next bit is now set to 1 and the process is repeated until all bits have been evaluated. With each iteration the occupancy moves in the direction of 0.5 with incrementally smaller steps, inevitably converging to said value. The strength of this approach is to find a suitable value in at most[3] 8 iterations. The found value will not necessarily be the value with occupancy closest to 0.5 due to the implementation of the procedure. Assuming sufficient triggers are used to have a good estimate of the occupancy, found values will be at most 1 DAC unit away from the "correct" unit. This should be kept in mind if the precision of the alignment is required to be to the unit.

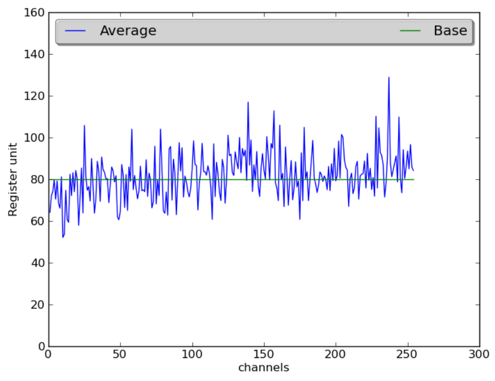
After determination of a suitable global baseline it is time to tune the offset for each channel. The approach is very similar and uses the same iterating solution. V_{plus} is now kept constant instead of the V_{offset}, and the latter is being iterated. Operating the chip at high occupancies (all channels enabled, low threshold) will induce undesired resonance modes in the chip. To avoid this, the calibration of V_{offset} is divided in 8 test groups, i.e. a group of 32 non-neighbouring channels. In principle this should also be done while calibrating V_{plus}. However, there are multiple advantages in not doing so. The precision of the V_{plus} calibration is also somewhat unimportant, since deviations will ultimately be corrected in the succeeding V_{offset} calibration.

In next sections, the consistency and quality of the calibration is evaluated for various set-ups by performing successive calibrations.

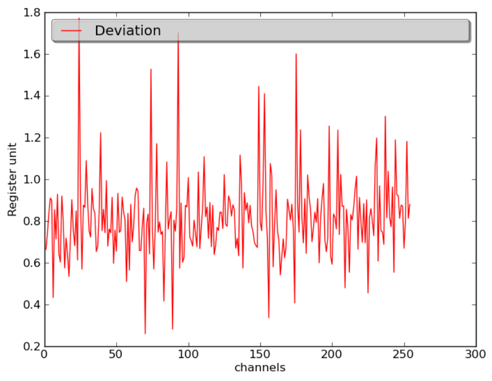
5.3.1 2CBC2 hybrid

Since the calibration is a statistical procedure -the tuning relies on occupancy which is inherently subject to variation- it is interesting to see how much one calibration differs from another. For each channel, the average V_{Offset} is computed from a sample of 100 calibrations and plotted in figure 20a. The standard variation of the sample is also included as it is a measure of consistency of the calibration.

[3]There is the possibility to stop the process early if the occupancy of current iteration is within a user specified range from the target value of 0.5.



(a) Mean



(b) Standard deviation

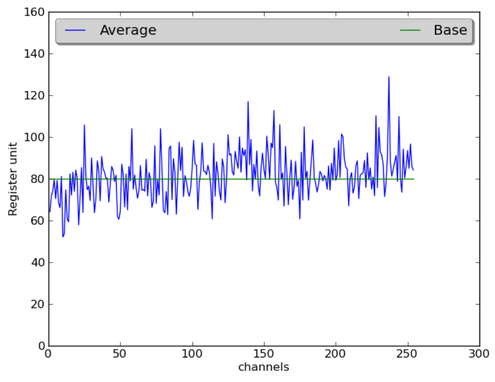
Figure 20: Resulting average and standard deviation of the offset per channel, after 100 successive calibrations on a bare 2CBC2 hybrid at target threshold 0x78 - $(120)_{10}$. The number of triggers per iteration is $N = 50$.

As mentioned before, the task of the offset is to correct differences arising from manufacturing limitations. Consequently, the shape of the histogram will be different from chip to chip. Figure 21 shows the offsets histogram of the second chip, from the same data sample as figure 20. In the remainder of this section, results will consistently be extracted from the same, first chip.

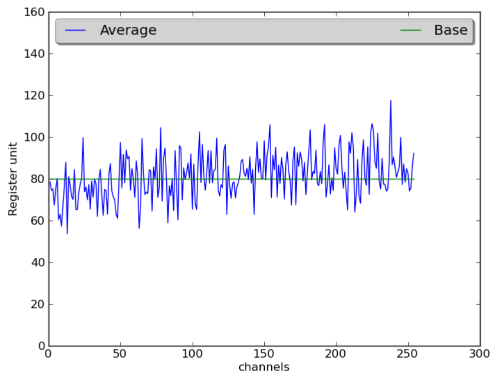
The same experiment can be repeated with different parameters for comparison. What would happen for example when the target threshold V_{CTH} is significantly raised? Only a change in V_{plus} is expected, the shape of the offsets histogram should remain unchanged. Which turns out to be correct as seen in figure 22a. Another option is to vary the number of triggers N per iteration and study its impact on the standard variation of the sample. Measurements for $N = 10, 50, 300$ have been made and are displayed in figure 23.

As expected the overall deviation decreases with increasing number of triggers. Evidently the number of triggers per iteration will increase the total time of the calibration routine. As time is an important constraint, a question arises concerning the optimal number triggers minimizing time but maximizing consistency. It is not immediately clear how much uncertainty is allowed. [11] discusses the acceptance regions of the threshold for high efficiency single and successive hit identification, in function of sampling time and mode (sample or latched). Variations of a single to a few V_{CTH} units doesn't seem too impactful.

An interesting exercise is to realize that the calibration can technically be performed with a single trigger per iteration. Since the occupancy can only ever be one or zero, the quality of the calibration is not expected to be any good.

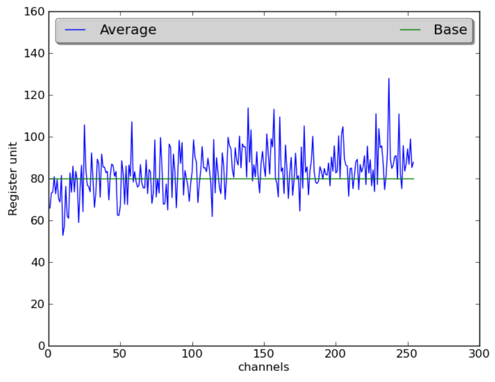


(a) Chip # 1

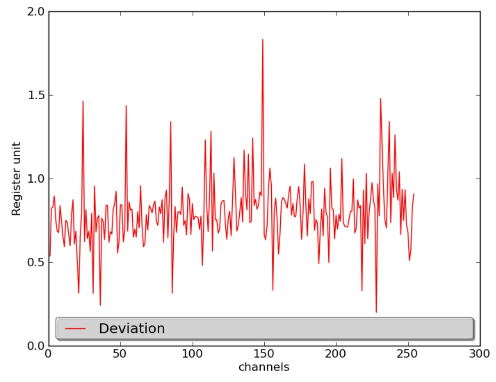


(b) Chip # 2

Figure 21: Comparison of the offset values after calibration from the two chips on the hybrid.



(a) Mean



(b) Standard deviation

Figure 22: Resulting average and standard deviation of the offset per channel, after 100 successive calibrations on a 2CBC2 hybrid at target threshold 0xC0 - $(192)_{10}$ and $N = 50$ triggers per iteration.

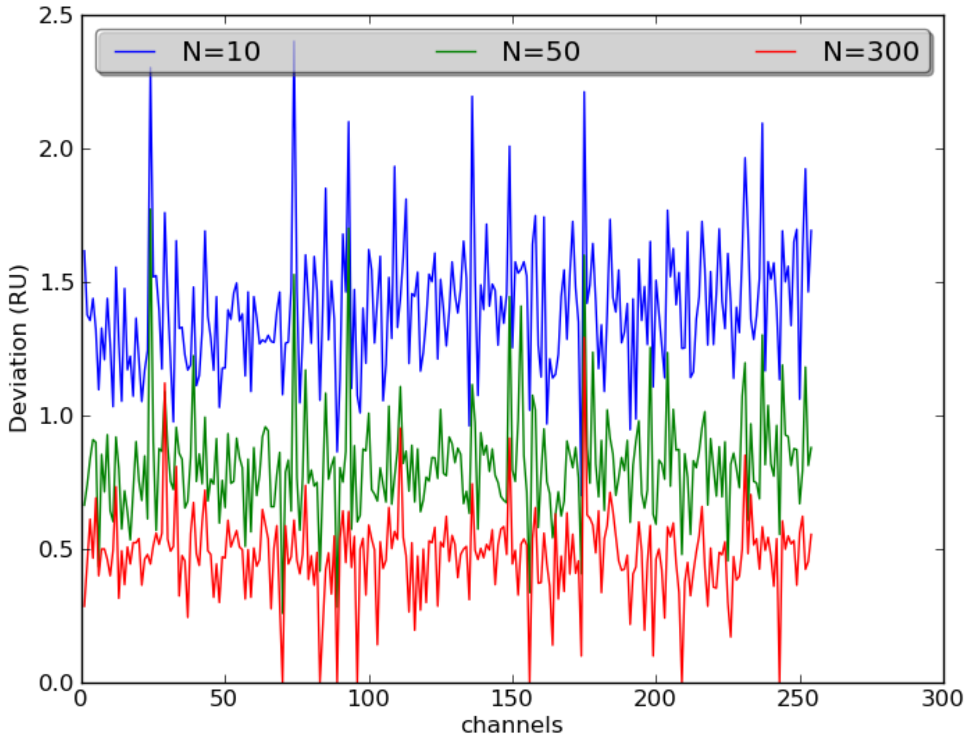
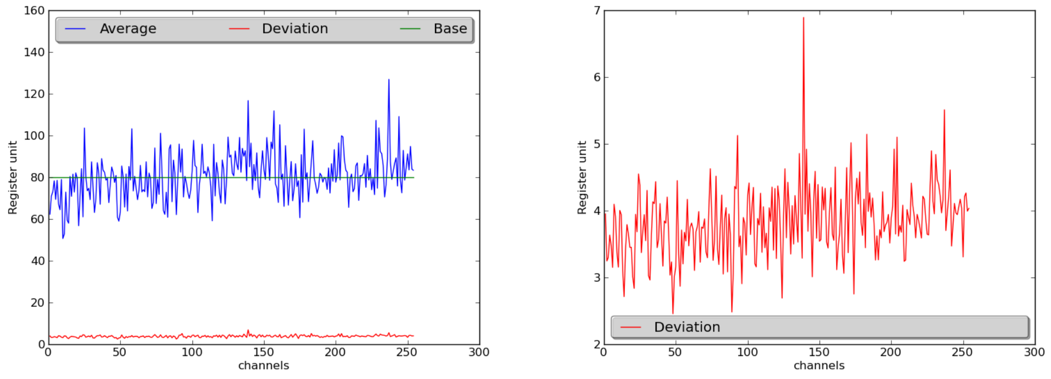


Figure 23: Standard deviations of the offset values from a sample of each 100 calibrations at target V_{CTH} 0x78 - $(120)_{10}$. The difference resides in the number of triggers per iteration, $N = 10, 50, 300$.

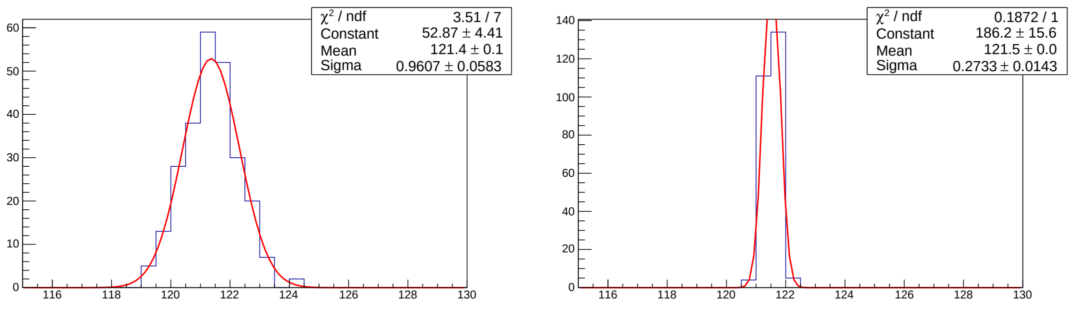


(a) Mean (b) Standard deviation

Figure 24: The mean and standard deviation of each channel's offset values from a data sample of 100 calibrations at target V_{CTH} 0x78 - $(120)_{10}$ with a single trigger per iteration.

However, the averaged values of successive calibrations does give an interesting result. As seen in figure 24, it converges to the same values of a calibration with many triggers.

A measure for the quality of a single calibration is provided by the distribution of the pedestals. Arguably this quantity is more important than the consistency of the calibration: in the end one wants to calibrate the chips once and be assured to have a useful result. Remember that the goal of the calibration is to align the pedestal of all channels, around the target threshold V_{CTH}. Data of the calibrations alone is not sufficient to determine this distribution, a threshold scan is required to construct S-curves for each channel. To determine the pedestal from the S-curves one could pick the value of V_{CTH} with occupancy closest to 0.5. However due to the relative small width of the S-curves and the integer values of V_{CTH}, pedestals will often be badly estimated with this method. As discussed in section 5.2, there are two other methods that will give more reasonable results. Figure 25 shows a plot of the pedestal distribution with configurations determined by a calibration with 1 and one with 300 triggers. The differentiation method for estimating the pedestals was used. Interestingly, the mean of both distribution are quite similar, especially when considering the actual resolution is a full DAC unit. Attentive readers might notice that while the procedure calibrated the chip for a target pedestal of 120, the mean appears to be somewhat higher ($\sim$ 121.5 V_{CTH} units). This is an artefact of the implementation of the procedure. It was purposely implemented to calibrate around occupancy 0.56, while the S-curves really determine the pedestal at occupancy 0.5. The difference between 1 trigger



(a) 1 trigger per iteration (b) 300 triggers per iteration

Figure 25: Pedestal distribution estimated by differentiation of the S-curves. The S-curve measurements were performed after calibration of the chip with a) 1 trigger per iteration and b) 300 triggers per iterations.

	N = 1	N = 10	N = 50	N = 300
Time (s)	3.676 ± 0.390	4.460 ± 0.476	7.250 ± 0.396	25.370 ± 0.355

Table 2: Average time of execution for samples of 100 calibrations, repeated for varying amount of triggers per iteration.

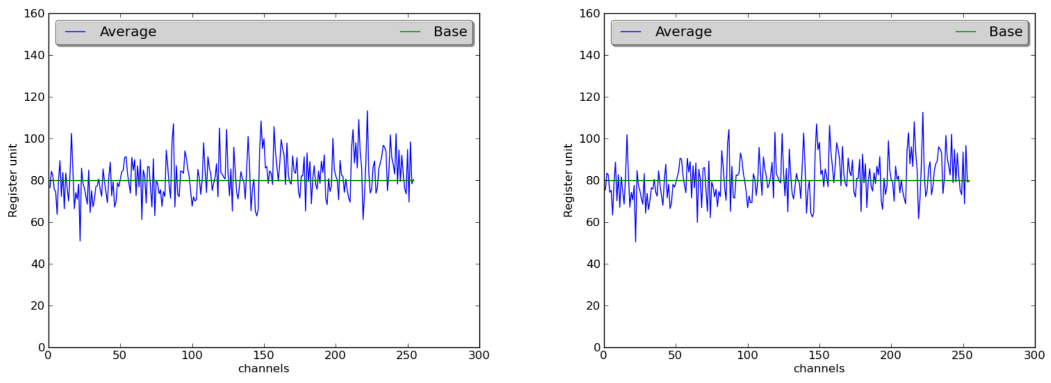
per iteration and 300 triggers is the spread of the distribution. As expected, the spread for low triggers is larger than for high triggers. The same questions remains as to how much of a spread is acceptable.

It's been mentioned a few times already, but an important parameter of the calibration is of course the time of execution. Table 2 shows the average time for each calibration in previous samples. While not really reflected in these numbers, the time to perform a calibration with a fixed amount of triggers can actually vary quite a lot, depending on the load of the computer. Here the calibrations are successive, so the load is somewhat constant for all calibrations.

5.3.2 2CBC2 mini-module

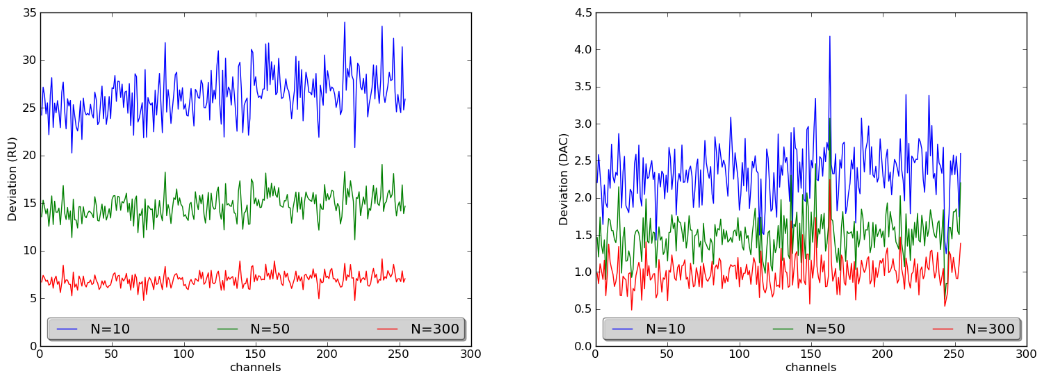
To meaningfully compare results with previous section, samples with identical parameters are acquired. Figure 26 shows the offsets histogram of CBC # 1 on the 2CBC2 hybrid connected to two sensors, measured with and without bias on the sensor side to side. The second chip is omitted in all following results because it is non-functional. Figure 27 displays the offset's standard variations for $N = 10, 50, 300$.

The shapes of the offsets histograms is naturally different from previous measurements because a different chip was calibrated, the important result is the iden-



(a) Unbiased sensor (b) Reverse biased sensor at 260 V

Figure 26: Average offset per channel computed from 100 calibrations at target threshold 0x78, with 300 iterations per trigger. The results are displayed for the first chips only from the 2CBC2 hybrid of the mini-module, operated with and without bias on the sensors.



(a) Unbiased sensor (b) Reverse biased sensor at 260 V

Figure 27: Standard deviations of offset for samples of 100 successive calibrations at target V_{CTH} 0x78 - $(120)_{10}$ with varying amount of triggers per iteration. The results are displayed for the first chips only from the 2CBC2 hybrid of the mini-module, operated with and without bias on the sensors.

	N = 10	N = 50	N = 300
Time (s) - unbiased	3.103 ± 0.275	5.677 ± 0.244	23.999 ± 0.310
Time (s) - biased	3.068 ± 0.170	5.748 ± 0.201	23.887 ± 0.205

Table 3: Average time of execution for samples of 100 calibrations of the 2CBC2 mini-module, repeated for varying amount of triggers per iteration.

tical offset histogram whether the sensor is biased or not. Not unexpectedly, the standard variations are quite different depending on the sensor bias. An unbiased sensor naturally has a large amount of free charges available for collection by the chip. The purpose of the bias is to mostly eliminate these. Even when biased the deviations are slightly higher compared to measurements on the 2CBC2 hybrid. Whatever produced noise on the hybrid is also present on the mini-module, the sensor adds noise on top of that.

Table 3 shows the average time execution of the calibration procedure. One has to be very careful when comparing these numbers with the 2CBC2 hybrid numbers. Here only one chip is calibrated, as opposed to two for the hybrid. Naively doubling these numbers - emulating two chip calibration - shows a substantial growth in the execution time compared to the uncoupled hybrid. One reason is that multiple chips are not, in fact, calibrated completely sequentially. Another reason is the same as last section, performance is quite heavily affected by the load of the computer. As these calibrations were successive, the load was constant and the variation is relatively small.

5.4 Moving to firmware

5.4.1 Motivation

Previous results have all been obtained by controlling the calibration procedure from within the software. The goal of this thesis is to implement a similar procedure running completely within the FPGA.

There are a few reasons why one would want to move the calibration procedure completely to hardware. A first one that comes to mind is to remove the unnecessary traffic between the FPGA and the computer. Another advantage of FPGAs is the use of combinatorial logic, potentially allowing multiple parts of the procedure to be run in parallel. The algorithm for calibrating a single chip doesn't really lend itself to run in parallel, though the true advantage appears when scaling up the system.

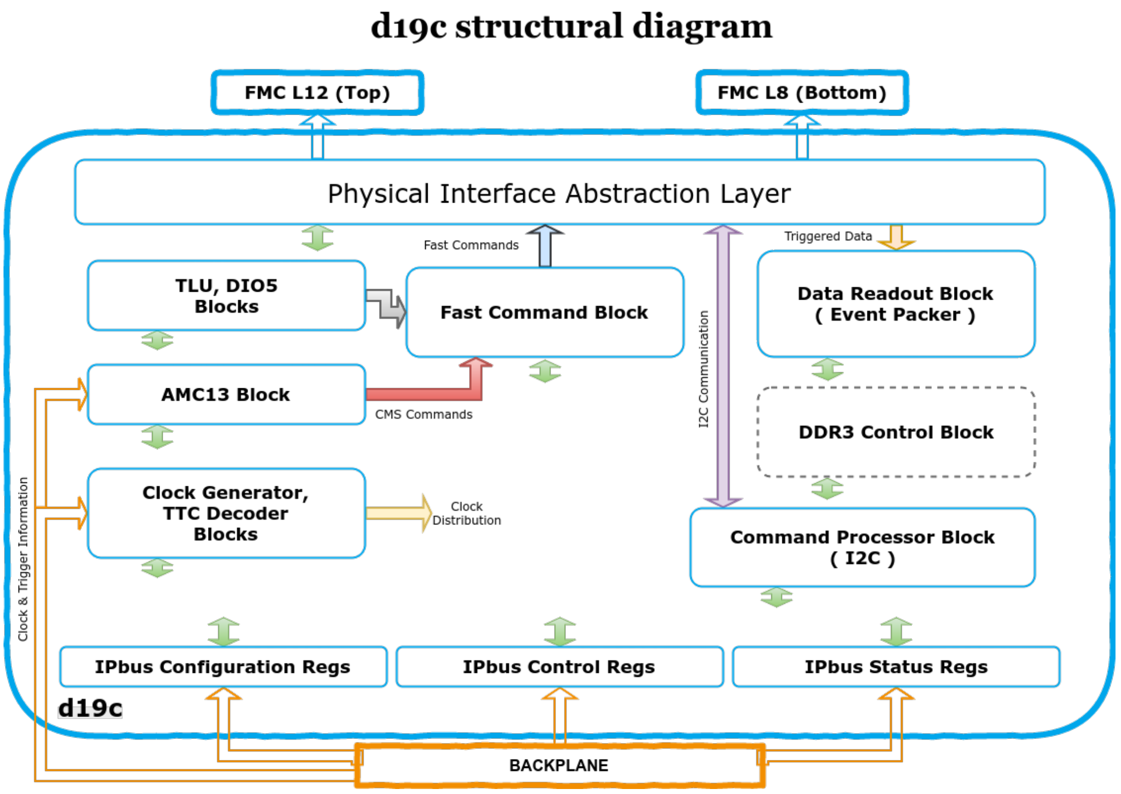


Figure 28: Diagram representation of the μDTC (formerly d19c) firmware structure.

5.4.2 μDTC firmware design

The FPGA design of the calibration is implemented in an existing framework, called μDTC, which acts as middleware between the software and the chips. A standard computer is for example not equipped to handle I^{2}C transactions without some kind of intermediary device. The μDTC design consists of multiple blocks carrying out specific tasks concerning features of the CBCs. These blocks are communicating with each other, passing commands and sending back results after execution. Figure 28 shows a diagram of the firmware structure. Some of the blocks' functions relevant to calibration are discussed in following paragraph.

The IPbus connects to the backplane and makes the connection between hardware and software. The latter controls reading or writing to registers on the FPGA. Three distinct blocks are available each fulfilling a different task. The configuration registers allow to configure aspects of the design after it is already loaded, e.g. number of triggers per iteration or trigger source. Status registers contain information about the status of other blocks; are there any

I^2C requests pending, is a calibration running? The control block is used to take actions; initiate procedures, request I^2C transactions, ... The IPbus data is decoded and commands are sent to appropriate blocks where they are executed.

The command processor handles all slow control. Each transaction is fairly slow, while requests can come in at a faster rate. The requests are queued up in the command processor, decoded and sent to the physical layer at an appropriate rate where the transmission happens following the I^2C protocol.

The fast command processor controls the management of fast commands. Test pulse or trigger requests can be time-sensitive and are best handled in synchronization with a precise clock. When appropriate, the actual fast commands are sent to the physical layer where they will reach the CBCs.

The physical interface abstraction layer describes the behaviour of the actual physical lines connected to the CBCs or any other device connected to the FPGA. This includes the two I^2C lines, requiring implementation of an I^2C master behaviour on the physical interface. Collection of channel content is also handled here as well as the physical transmission of the fast commands.

To implement the calibration, a new calibration block is added to the structure of μDTC. The block fulfils a similar role to the command processor and fast command block, managing signal requests in a particular order and timing. Calibrating requires sending slow control commands and triggers in order to collect data in different configurations. Thus the calibration block will be in communication with both the command and fast command blocks. Channel content arrives in the physical interface layer, and is directly forwarded to the calibration block. On the user side, the IPbus blocks are used to start the calibration procedure, configure target V_{CTH} or $N_{triggers}$, or read if the calibration is still proceeding.

5.4.3 Implementation

The binary bisection method is conserved implemented as a state machine in communication with the appropriate blocks from the μDTC firmware structure. A separate state machine is designed for both the V_{plus} extraction and V_{offset} extraction separately. Figure 29 shows a diagram for the determination of V_{plus}.

V_{plus} is calibrated with all channels active at the same time, saving quite a bit of time. The inaccuracy in the calibration of V_{plus} is corrected during the offsets calibration. The state machine starts by default in the *Idle* state where it waits

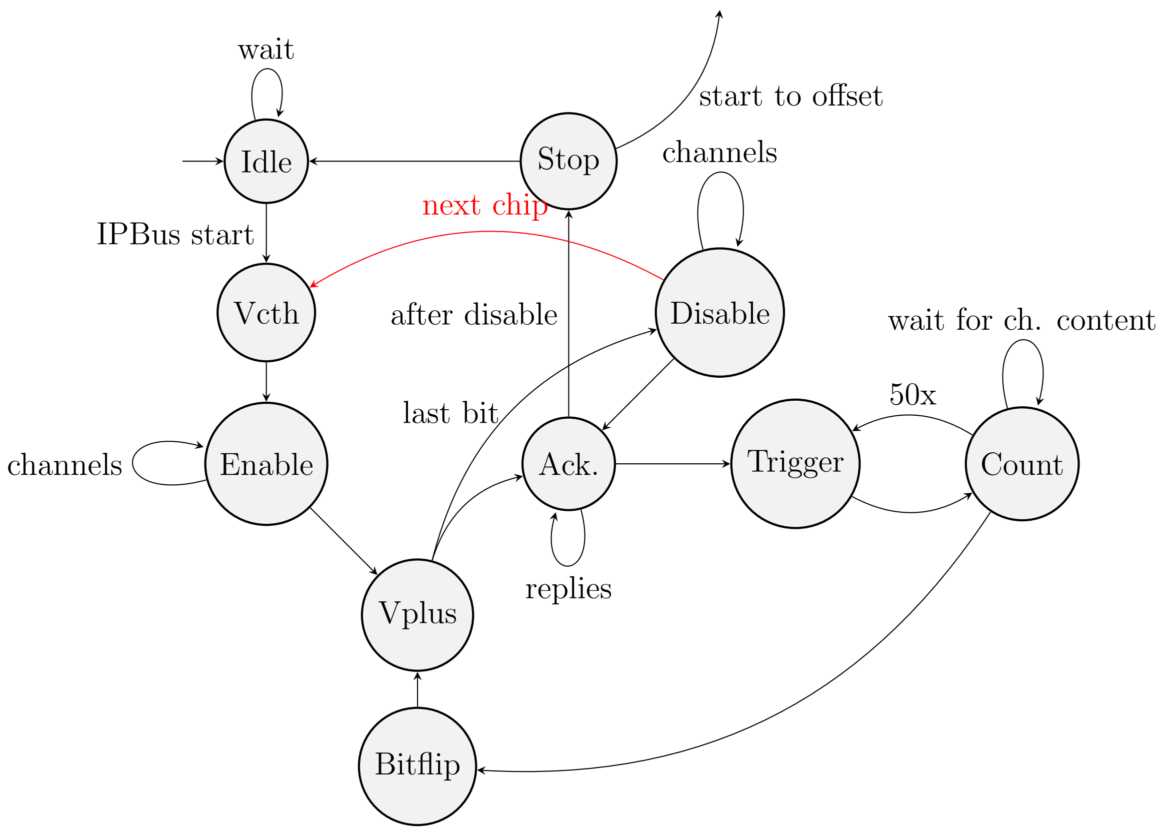


Figure 29: Schematic of the state machine for the extraction of V_{plus}. The initial state *Idle* waits for an external input to start the procedure. The states with a loop pointing to themselves are stages where an operation has to be repeated many times (e.g. for all channels) or a general idling state waiting for a confirmation to proceed (I^{2}C reply). When multiple paths are available it should be clear from context which to follow.

for a signal from the IPbus control registers to begin the procedure. When fired, the next state *init* initializes internal signals to their correct default values and sends an I^{2}C request to the command processor block to set the target threshold value V_{CTH}. *Enable* requests a write transaction for every channel, setting the offsets to 0x50. The main loop of the procedure starts at *Vplus*, requesting the initial value "1000 0000" for the first iteration of measurements. As discussed before, I^{2}C communication is quite slow and the measurements shouldn't start before confirmation that all (relevant) registers are configured. The *Ack*nowledge state counts the replies from the command processor. When it matches with the number of requested transactions, the calibration can continue. A *trigger* state is followed by a *counting* stage. Again, the transmission of data is not instantaneous, in fact it takes at least 266 cycles, the time to serially transmit all data from the buffer RAM, before reaching the calibration module. The state points to itself until data is received, after which it points back to the trigger state. This loop is repeated until sufficient hit data is collected, in this case 50 times. Then the *Bitflip* state compares the total hit count to a theoretical half-full occupancy count ($0.5 \times N_{triggers} \times N_{channels}$), flipping the bit if necessary and moving on to the next one. It marks the end of the first iteration, which is repeated for all bits of V_{plus}. After all iterations are cleared, all channels are disabled (offset set to 0x00) in anticipation of the offsets tuning. When all I^{2}C replies of the final configuration have been counted, the last state *stop* is reached, where a start signal is sent to the offsets state machine. The V_{plus} machine sits in *idle* state until another calibration request is sent.

In principle it is possible to exactly count the number of state transitions and multiply it with the frequency of the driving clock. If every state represents one clock cycle only, the V_{plus} extraction would require: $8 \times (3 + 2 \times 50) + 6 = 830$ clock cycles. At a frequency of 40 MHz this would take 0.020750 ms to perform. As discussed however some states point to themselves, repeating an operation multiple times or waiting for feedback to continue. *Enable* and *Disable* each send 256 requests for I^{2}C transactions, two clock cycles are needed for each request. The biggest contributions are however from the *Ack* and *Count* states. The *Ack* waits and counts the number of replies from the expected transactions. Each transaction takes at least 64 clock cycles, 36 to read the page and 28 to write, at a frequency of only 1 MHz. Accounting for another $(8 + 512) \times 64 \times 40 = 1331200$ cycles. The *Count* state waits for serialization of the channel content, which takes at least 266 cycles, and is repeated 400 (8×50) times. A complete overview of the time consumption per state is summarized in table 4. The diagram represents an extraction for a single chip, and amouonts to a total of 36 ms. It can be expanded to a sequential calibration of n chips with the addition of the red arrow state transition. This would set the time performance to $n \times 36$ ms.

State	Cycles	Frequency (MHz)	Time (ms)
Config.			
Enable	512	40	0.0128
Disable	512	40	0.0128
Ack.	512×64	1	32.768
Others	4	40	0.0001
Subtotal	**1311748**	40	**32.7937**
Main loop			
Trigger	50	40	0.0013
Count	50×266	40	0.3325
Ack.	1×64	1	0.0280
Other	2	40	0.0001
Subtotal	15912	40	0.3978
$\times 8$ iterations	**127296**		**3.1824**
Total	**1439044**	40	**35.9761**

Table 4: Estimate of the time consumption for the extraction of V_{plus} per state. The individual contributions from the main loop are summed and multiplied by 8 for each iteration of the procedure. The total is the sum of the product and individual contributions of the configuration part of the procedure. The clock cycles operating at 1 MHz are normalized to 40 MHz before being taken into account for the total number of cycles.

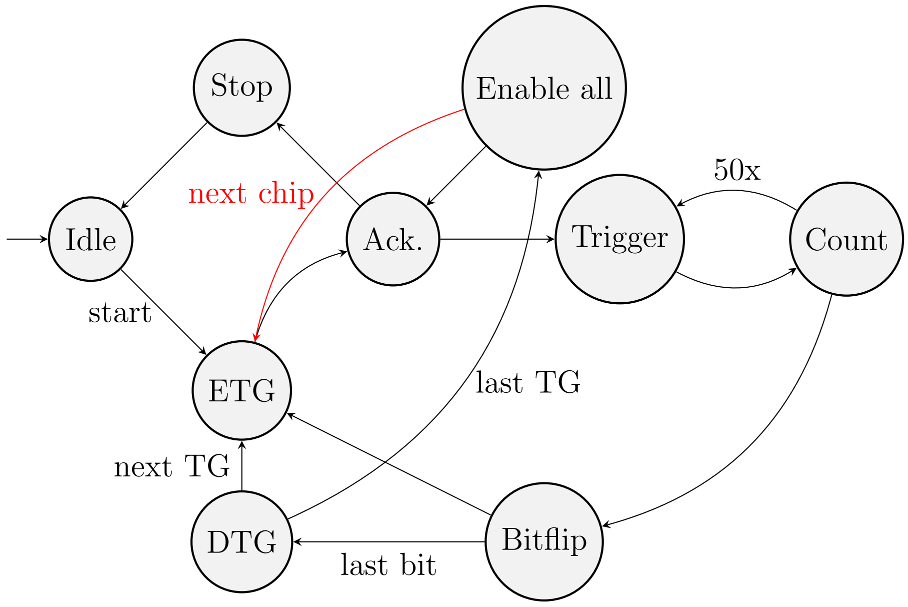


Figure 30: Schematic overview of the offsets calibration. The initial state *Idle* waits for an external signal to start the procedure. In the figure above, TG stands for Test Group, while the prefix E and D is short for Enable and Disable respectively.

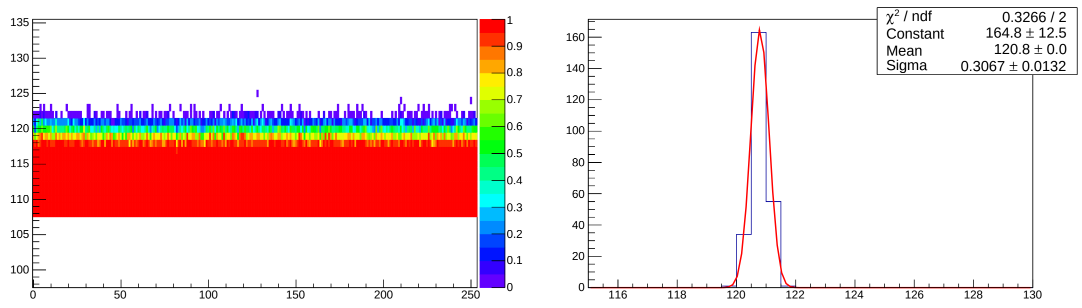
The state machine for the offsets is very similar to V_{plus} and displayed in figure 30. Major distinctions are the iterations over the test groups, counting channel by channel and of course varying V_{offset} instead of V_{plus}. These changes are reflected in the main loop; an extra state is introduced to disable a test group before enabling the next one. The structure has now 8 bit iterations nested in 8 test group iterations. Evidently the timing of the procedure will change, table 5 shows a breakdown of each state. Here also the diagram represents offsets calibration of a single chip. A sequential calibration is implemented with the addition of the red arrow state transition, with an expected time performs of $n \times 169$ ms for n chips. The increase in time is mostly explained with the large increase of I^{2}C transactions required to iterate every channel, the separation into test groups also requires more counting stages. A complete calibration, counting both V_{plus} and $V_{offsets}$ extractions, would then take $n \times 205$ ms.

5.4.4 Results

For meaningful comparison, the firmware procedure should do the same as the software calibration. Both procedures tune V_{plus} with all channels enabled, but the software calibration does acquire data for all chips at the same time. More importantly, the software procedure reads back I^{2}C registers after writing to them,

State	Cycles	Frequency (MHz)	Time (ms)
Config.			
Enable all	256×2	40	0.0128
Others	2	40	0.0001
Ack.	256×64	1	16.3840
Subtotal	**655874**	40	**16.3969**
Main loop			
ETG	32×2	40	0.0016
Ack.	32×64	1	2.0480
Trigger	50	40	0.0013
Count	50×266	40	0.3325
Others	1	40	0.0000
Subtotal	95335	40	2.3834
$\times 8$ bits	762680	40	19.0670
$\times 8$ test groups	**6101440**	40	**152.5360**
Total	**6757314**	40	**168.9329**

Table 5: Estimate of the time consumption for the offsets extraction per state. the main loop now iterates over 8 bits, for 8 test groups of 32 channels. In each last bit iteration ETG is replaced by DTG, the cycle count is however the same so a simple multiplication by a factor of 8 is sufficient. The clock cycles operating at 1 MHz are normalized to 40 MHz before being taken into account for the total number of cycles.



(a) S-curves of each channel (b) Gaussian fit of the pedestal distribution

Figure 31: Pedestal alignment after calibration with firmware procedure. The calibration was run with a target threshold value 0x78 - $(120)_{10}$ and $N = 50$ triggers per iteration.

to absolutely make sure the register has been correctly configured. This is not done in the firmware calibration. Finally, when the firmware calibration is finalized the result is just that: a calibrated chip. The software procedure calibrates the chip and also immediately has a histogram of V_{plus} and V_{offset} available. So it is a bit unfair to compare software to firmware time performance.

As an interesting side note, it has been tried to drive the software procedure without read-backs, but it turns out calibrations are failing. The firmware procedure doesn't seem to be bothered by it.

As the tuning method is essentially the same, there is no doubt that the procedure will produce a sensible result. Comparing all offsets histogram with their deviations is not really insightful as the results are very similar, which is good of course. Nevertheless a single pedestal distribution with accompanying S-curves is presented in figure 31 to ease the mind of the sceptics.

The real contribution of the FPGA implementation is the time-performance of the procedure. A counter signal was added in various states counting the number of clock cycles for: the total procedure, the time spent waiting for I^{2}C, and the time spent waiting for data collection. Results are displayed in tables 7 and 6 for single and two-chip calibrations with various parameters.

The results are quite pleasing. They convincingly confirm the sequential behaviour of the calibration for multiple chips, and the optimistic estimation for $N = 50$ is close to the actual result. A tremendous advantage over the software procedure is the consistency of the time performance. Each calibration will take exactly so many clock cycles, independently of the computer load.

# triggers	I^2C	Data collection	Total	Time (s)
10	8514164	203760	8724883	0,2181
50	8514164	1018800	9542803	0,2386
300	8514164	6112800	14654803	0,3664

Table 6: Number of clock cycles spent waiting for I^2C transactions, collecting channel content, and the total amount of clock cycles to perform a full calibration. The last column is the time-equivalent of the total at 40MHz.

# triggers	I^2C	Data collection	Total	Time (s)
10	17028328	407520	17449763	0,4362
50	17028328	2037600	19085603	0,4771
300	17028328	12225600	29309603	0,7327

Table 7: Number of clock cycles spent waiting for I^2C transactions, collecting channel content, and the total amount of clock cycles to perform the calibration of a 2CBC2 hybrid. The last column is the time-equivalent of the total at 40MHz.

6 Outlooks

In the previous section, we have demonstrated the successful achievement of the goal of the thesis: a first efficient implementation of the CBC calibration procedure into firmware. While this is very encouraging towards this calibration and other procedures being run in parallel from firmware for the eventual full tracker, still some improvements are to be foreseen for this calibration procedure.

There are two approaches in improving the time efficiency of the procedure. The main loop of a single chip can be improved, and multiple chips can be (partly) calibrated at the same time.

Remember that the task of the main loop is to determine V_{plus} and V_{offet} values for which the occupancy of the channel measured on noise is (close to) 0.5. The number of iterations in the main loop can be reduced by ending the loop early when a value is reached in a user specified range around 0.5.

As pointed out in previous section, the current implementation of multi-chip calibrations is completely sequential. Parts of the calibration can however run in parallel for several chips, preventing unnecessary repetition of time expensive stages of the procedure. As a vast amount of chips will be calibrated at the same time, the benefit of parallelization can be tremendous.

While technically I^2C communicates with a single chip at a time, there is the option to broadcast the same data to all chips on a hybrid. This is especially useful for the initial configuration of the chips which should all start in the same state. A downside of broadcasting is that the response from the slaves is lost. It is best to implement read back in the procedure to make sure the broadcast was successful.

When requesting a trigger, data from all chips is collected. The current implementation of the procedure runs as follow: configure registers from chip $I \Rightarrow$ collect data $\Rightarrow$ reconfigure registers $\Rightarrow$ collect data $\Rightarrow \ldots \Rightarrow$ configure registers from chip $I+1 \Rightarrow \ldots$, repeating the data collection 8 times for every chip. Instead it would be more efficient to: configure registers from chip $I \Rightarrow$ configure registers from chip $I + 1 \Rightarrow$ collect data $\Rightarrow$ reconfigure registers chip $I \Rightarrow$ reconfigure registers chip $I + 1 \Rightarrow \ldots$.

References

[1] CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, 2012.

[2] ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters B*, 2012.

[3] The Phase-2 Upgrade of the CMS Tracker Technical Design Report, jul 2017.

[4] CMS Collaboration. The TriDAS project, Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger, 2002.

[5] Frank Hartmann. *Evolution of Silicon Sensor Technology in Particle Physics*. Springer, 2009.

[6] Davide Braga and Mark Prydderch. *CBC2 (CMS Binary Chip 2) User Guide 1.3*, nov 2013.

[7] I^2C protocol. `https://www.i2c-bus.org/`.

[8] Davide Braga. *Development of the Readout Electronics for the High Luminosity Upgrade of the CMS Outer Strip Tracker*. PhD thesis, Imperial College London, jan 2016.

[9] Xilinx. *7 Series FPGAs Configurable Logic Block - user guide*, sep 2016.

[10] μTCA standard. `https://www.picmg.org/openstandards/microtca/`.

[11] Georg Auzinger, Nikkie Deelen, Stefano Mersi, Sarah Seif El Nasr-Storey, and Giovanni Zevi Della Porta. CBC3 Single Module Simulation.